Normalized Gradient Vector Diffusion and Image Segmentation

Zeyun Yu and Chandrajit Bajaj

Department of Computer Science, University of Texas at Austin,
Austin, Texas 78712, USA
zeyun@cs.utexas.edu

Abstract. In this paper, we present an approach for image segmentation, based on the existing *Active Snake Model* and *Watershed-based Region Merging*. Our algorithm includes initial segmentation using *Normalized Gradient Vector Diffusion* (*NGVD*) and region merging based on *Region Adjacency Graph* (*RAG*). We use a set of heat diffusion equations to generate a vector field over the image domain, which provides us with a natural way to define *seeds* as well as an external force to attract the active snakes. Then an initial segmentation of the original image can be obtained by a similar idea as seen in active snake model. Finally an *RAG*-based region merging technique is used to find the true segmentation as desired. The experimental results show that our *NGVD*-based region merging algorithm overcomes some problems as seen in classic active snake model. We will also see that our *NGVD* has several advantages over the traditional gradient vector diffusion.

Keywords. Image segmentation, Gradient vector diffusion, Heat diffusion equation, Active snake model, Watershed method, Region merging.

1 Introduction

Since it was proposed by *M.Kass etc.* [1], active snake model has drawn a lot of attention from researchers in image-related fields. Due to its efficiency of converging to the desired features within an image by simply defining an energy function, it has found many applications, including edge detection [1], shape modeling [14], segmentation [15], and motion tracking [15, 16].

The traditional snake model is defined by an energy functional:

$$E = \int_0^1 \frac{1}{2}[\alpha\|\mathbf{x}'(s)\|^2 + \beta\|\mathbf{x}''(s)\|^2] + E_{ext}(\mathbf{x}(s))ds. \tag{1}$$

where $\mathbf{x}(s) = [(x(s), y(s)]$, and $s \in [0, 1]$. The first two terms within the above integral stand for the internal force that is determined by the geometric properties of the snakes, while the third term is thought of as the external force that is the issue mostly discussed in active snake models.

Generally there are several difficulties with this traditional model. First, the initial guess of the contour must be carefully chosen to be close to the true

boundaries. This is because the snake moves partially in the direction of external force $E_{ext}(\mathbf{x})$ which is based on the images gradient. Therefore, the external force exists only at those points close to the boundaries. Some approaches have been proposed to solve this problem, e.g., *multi-scale method* [8], where different Gaussian factor σ are used to create a scalable potential force to attract the snake to the desired boundaries; *distance potential force* [5], in which the distance map is produced first and then, based on the gradient of the distance map, a potential force is obtained to generate a large attraction range. Recently *Xu et.al.* proposed another method [2], called *gradient vector flow (GVF)*, to find the external force based on a group of heat diffusion equations.

The second problem with the traditional model is that it is difficult for the snake to move into the boundary concavities. If we consider the energy functional (1) again, we can see that the first two terms in the integral always make the snake as straight as possible. Thus, if the external force $E_{ext}(\mathbf{x})$ is not large enough to push the snake into the boundary concavities, the snake will always stop near the "entrance" of the concavities. Although many ways have been tried to solve this problem (see [9–12]), most of them did not give satisfying results. *Xu*'s GVF method and its generalized version [3] were originally proposed to rectify this problem but still did not work very well in the case of long and thin boundary concavities (see next section for analysis). Furthermore, boundary "gaps" or weak boundaries are always overwhelmed by the nearby strong boundaries.

The third problem with the traditional model is how to select the initial snakes. A good guess of the initial snakes makes a great impact on the final segmentation. Many of the existing methods, including *Xu*'s methods [2–4], choose the initial snakes by hand. Some other approaches find the initial snakes by "balloons" or by the locus of the zero-crossing of the Laplacian of the smoothed images (see [18] for a summary). In the present paper we will see a very natural way to find the initial snakes (or seeds).

To overcome these problems we proposed a new type of diffusion equations to generate the *gradient vector flow*. Our new diffusion equations are similar to those in [2] except that ours are based on the normalized gradient vector diffusion (NGVD). This slightly modified scheme does tremendously improve the vector diffusion behaviors, compared to other vector diffusion methods [2–4]. First, it is very easy to deal with the boundary concavities such that the snakes can easily move into the concavities. Second, weaker but obvious boundaries can be preserved even they are close to much stronger boundaries. Third, the snakes can correctly stop near the boundary "gaps". Based on the obtained vector field, we then propose an approach to generate the image segmentation in the following way. First, all the *source* points are identified over the image domain, then an initial segmentation can be easily obtained from the previously produced vector field. Finally an RAG-based approach was used to find the true segmentation of the original image. We will also see an interesting relationship between our NGVD-based approach and the classic watershed method and how our approach is better in some aspects than watershed method.

We organize the rest of this paper as follows: in next section we will review the work by *Xu etc.* [2, 3]. Then in section 3, our approach of generating vector field will be given and we will see the difference between our NGVD approach and *Xu*'s method. Sections 4 discusses how the gradient vector flow generated by our method can be applied to the image segmentation, including initial segmentation, region merging and the comparison between our approach and watershed method. And then we will present some segmentation results on different types of images in section 5. Finally in section 6, we give our conclusion.

2 Review of Gradient Vector Flow

In this section we give a brief description of this approach. We refer the readers to [2–4] for more details.

2.1 Edge Map

This method begins by defining an edge map $f(x, y)$ derived from the original image $I(x, y)$. The edge map should have the property that $f(x, y)$ is large near the image boundaries and small within the homogeneous regions. There are many ways to define an edge map. Commonly used are:

$$f(x, y) = -\|\nabla I(x, y)\|^2, \tag{2}$$

or

$$f(x, y) = -\|\nabla[G_\sigma(x, y) * I(x, y)]\|^2. \tag{3}$$

Then take the gradient of the edge map as the initial values of the vector flows, that is, $u^{(0)} = \partial f/\partial x, v^{(0)} = \partial f/\partial y$. In the following we will denote $\partial f/\partial x$ and $\partial f/\partial y$ as f_x and f_y, respectively.

From the definition, we know that the initial vectors determined by $\nabla f(x, y)$ always point toward the image boundary and have large magnitudes near the boundaries and low magnitudes in the homogeneous regions.

2.2 Gradient Vector Flow

In the work of *Xu et.al.*[2–4], the following energy functional was used to propagate the vectors from the boundaries to the inner parts of homogeneous regions:

$$\varepsilon = \int\int \mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + \|\nabla f\|^2\|\mathbf{v} - \nabla f\|^2 dxdy \tag{4}$$

where $\mathbf{v}(x, y) = [u(x, y), v(x, y)]$, and as we mentioned above, the initial value of $\mathbf{v}(x, y)$ is determined by $\nabla f(x, y)$. μ is a regularization parameter to be set on the basis of noise present in the image [2].

This variational formula consists of two terms. The first term, the sum of the squares of the partial derivatives of the vector field, makes the resulting

vector flow $\mathbf{v}(x, y)$ varying smoothly. The second term stands for the difference between the vector flow and its initial status. Thus minimizing this energy will force $\mathbf{v}(x, y)$ nearly equal to the gradient of the edge map where $\|\nabla f(x, y)\|$ is large.

It can be shown that equivalent diffusion equations can be derived from the minimization of the energy functional defined above (treating u and v as functions of time):

$$\begin{cases} \frac{\partial u}{\partial t} = \mu \nabla^2 u - (u - f_x)(f_x^2 + f_y^2) \\ \frac{\partial v}{\partial t} = \mu \nabla^2 v - (v - f_y)(f_x^2 + f_y^2) \end{cases} \tag{5}$$

These two equations can be separately solved by numerical method, resulting a vector flow over the entire image domain mostly with non-zero magnitudes if enough number of iterations are executed. And it was shown that an appropriate choice of time step $\triangle t$ guaranteed the convergence of GVF.

2.3 Generalized Gradient Vector Flow

As pointed out in [3], the GVF method described above has some difficulties when the boundary concavities are long and thin. To remedy this problem, *Xu etc.* proposed a method called *generalized gradient vector flow*(GGVF). The essential idea is to introduce two weighting functions for (5):

$$\begin{cases} \frac{\partial u}{\partial t} = g(\|\nabla f\|)\nabla^2 u - h(\|\nabla f\|)(u - f_x) \\ \frac{\partial v}{\partial t} = g(\|\nabla f\|)\nabla^2 v - h(\|\nabla f\|)(v - f_y) \end{cases} \tag{6}$$

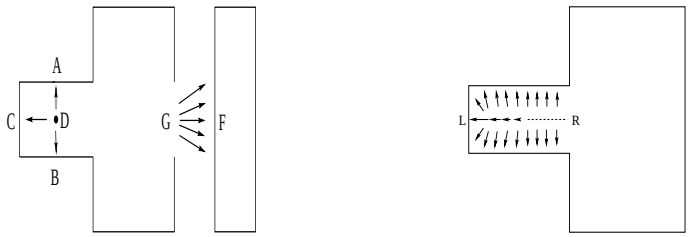
where

$$g(\|\nabla f\|) = e^{-(\|\nabla f\|/K)}, \qquad h(\|\nabla f\|) = 1 - g(\|\nabla f\|).$$

As claimed in [3], the problem with GVF [2] is caused by excessive smoothing of the field near the boundaries. Hence, the weighting functions $g(\|\nabla f\|)$ and $h(\|\nabla f\|)$ are introduced to decrease the smoothing effect near strong gradient. Consider a boundary concavity (see *fig.1(a)* ACB). As we know, the force coming from C is decreasing from left to right. Therefore, if D is far from C (assuming the concavity is arbitrarily long), the force from C will decrease to zero. On the other hand, the forces from A and B are relatively much stronger, resulting vectors pointing either up or down at the "entrance" of the concavity (see *fig.1(b)*). This problem prevents the snakes moving into the concavity. Although in the generalized GVF method the weighting functions $g(\|\nabla f\|)$ and $h(\|\nabla f\|)$ are used such that the force coming from C becomes stronger compared to that calculated in GVF method, but, as we can see in equations (5) and (6), the diffusion equations are isotropic, thus the forces from A and B also become stronger. The resulting vectors near the "entrance" of the concavity are still pointing either up or down. Remember that here we ignore the forces coming

from the right (where R is shown in *fig.1(b)*) to the concavity since the analysis is the same. Therefore the true problem does not occur at the "entrance" of the concavity, but in the central area of the concavity. The readers will see this fact soon in next section.

Another problem with GGVF (and GVF as well) is how to stop the curves from moving out of the boundaries "gap". If the boundary gap is isolated (like the examples in [2, 3]), these algorithms work correctly on the boundaries with gaps. However, if there is another boundary near this gap (as seen in *fig.1(a)* where G is shown), all the vectors around G resulting from equation (5) or (6) will point to F. This makes the curves move out of G and causes a problem. Similar problem happens when weak boundaries are close to relatively much stronger boundaries such that the weaker boundaries are overwhelmed after diffusion.



(a) Problems with traditional GVF (b) Vector diffusion by GVF

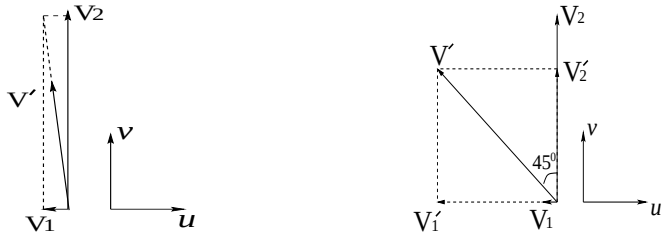
Fig. 1. Illustration of the problems seen in traditional GVF method

3 Normalized Gradient Vector Diffusion

In this section we will describe our normalized gradient vector diffusion, which overcomes the problems seen in traditional gradient vector diffusion. For simplicity we will call this method NGVD in the following.

3.1 Analysis: the Ideas

The traditional GVF and even the generalized GVF methods cannot efficiently solve the problems as we have seen above. The key observation behind these problems is that a weak vector (with almost zero magnitude) makes a very little impact on its neighbors that have much stronger magnitudes. However, if we normalize the vectors before we apply the diffusion equations, the results will be quite different. *Fig.2* gives an example of how these two different ways diffuse two vectors and have totally different results. For the normalized vector diffusion method, a very weak vector can tremendously affect a strong vector, both on its magnitude and on its orientation. This is the essential idea behind our new diffusion approach.



(a) Traditional GVF (b) Normalized GVD

Fig. 2. Two ways to diffuse vectors: the way on the left shows the traditional method that results in a vector almost pointing up, while the method on the right shows our normalized GVD in which the resulting vector is equally influenced by both V_1 and V_2 regardless their magnitudes.

3.2 Heat Equations and Algorithms

There are two different ways to implement the normalized vector diffusion: one is isotropic diffusion, similar to Xu's method, the other is anisotropic diffusion. Both ways produce almost the same vector field although their implementations are slightly different.

Isotropic Diffusion The heat equations of the isotropic vector diffusion are very similar to the equations used in the traditional GVF method as we saw in previous section. The only difference is that in our NGVD method we need to normalize all the vectors before we apply the diffusion operations. The heat equations can be written as:

$$\begin{cases} \frac{\partial u^*}{\partial t} = \mu \nabla^2 u^* - (u^* - f_x)(f_x^2 + f_y^2) \\ \frac{\partial v^*}{\partial t} = \mu \nabla^2 v^* - (v^* - f_y)(f_x^2 + f_y^2) \end{cases} \tag{7}$$

where (u^*, v^*) is the normalized vector on a pixel over the image domain and (f_x, f_y) stands for the initial vector on the same pixel. This equation can be further converted into an iterative numerical version by using simple central difference scheme (more details can be found in [2]). The overall algorithm is described as follows:

Initialize: Calculate $f(x, y)$ using (2) or (3). Let $u^{(0)} = f_x, v^{(0)} = f_y$.
Repeat: (for each iteration)
(a) Normalize all the non-zero vectors to unit vectors.
(b) Calculate the new value for u using the first equation in (7).
(c) Calculate the new value for v using the second equation in (7).

Remember that the number of iterations can be set manually or automatically. The latter one is achieved by checking the maximal change of the vectors

between two adjacent iterations. If the maximal change is smaller than a fixed threshold, then the diffusion terminates.

Anisotropic Diffusion The normalized gradient vector diffusion can also be simulated with anisotropic diffusion (see [13] for its definition). The heat equations can be written in anisotropic forms as follows:

$$\begin{cases} \frac{\partial u}{\partial t} = \mu \nabla (\frac{1}{\sqrt{u^2+v^2}} \nabla u) - (u - f_x)(f_x^2 + f_y^2) \\ \frac{\partial v}{\partial t} = \mu \nabla (\frac{1}{\sqrt{u^2+v^2}} \nabla v) - (v - f_y)(f_x^2 + f_y^2) \end{cases} \tag{8}$$

where (u, v) is the vector being diffused at a pixel over the image domain and (f_x, f_y) is the initial vector at the same pixel.

Using a similar scheme seen in [13], we can rewrite the above equation as follows:

$$\begin{cases} \frac{\triangle u_{i,j}}{\triangle t} = \frac{\mu}{4} \times (\sum_{(m,n) \in N(i,j)} \frac{u_{m,n}}{\sqrt{u_{m,n}^2 + v_{m,n}^2}} - 4u_{i,j}) - (u_{i,j} - f_x)(f_x^2 + f_y^2) \\ \frac{\triangle v_{i,j}}{\triangle t} = \frac{\mu}{4} \times (\sum_{(m,n) \in N(i,j)} \frac{v_{m,n}}{\sqrt{u_{m,n}^2 + v_{m,n}^2}} - 4v_{i,j}) - (v_{i,j} - f_y)(f_x^2 + f_y^2) \end{cases} \tag{9}$$

where (i, j) is the pixel being considered and (m, n) stands for the neighbors of (i, j) (either 4-neighbor or 8-neighbor scheme can be used here). Note that equation (9) is not exactly equivalent to equation (8) from the perspective of anisotropic diffusion defined in [13]. But the experiments showed that the slightly modified scheme in (9) gives more desirable results. The overall algorithm for the anisotropic diffusion version can be described as follows:

Initialize: Calculate $f(x, y)$ using (2) or (3). Let $u^{(0)} = f_x, v^{(0)} = f_y$.
Repeat: (for each iteration)
(a) Calculate the new value for u using the first equation in (9).
(b) Calculate the new value for v using the second equation in (9).

3.3 Comparison Between Different Diffusion Schemes

In the following we will compare our approach with the traditional GVF/GGVF methods [2–4, 12] on their abilities of dealing with boundary concavities and boundary gap. *Fig.*3 shows the testing image which we will work on. This image contains one boundary concavity and one boundary gap. The regions indicated by the two dashed rectangles will be enlarged in *fig.*4 and *fig.*5 such that the vector field can be more clearly displayed.

Fig.4(a) shows the enlarged vector field generated by the traditional gradient vector diffusion around the boundary concavity. We can see that the vectors in the concavity (especially those near the center) still point up or down as they

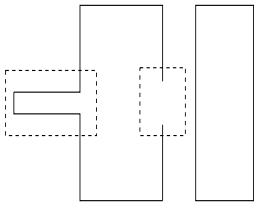
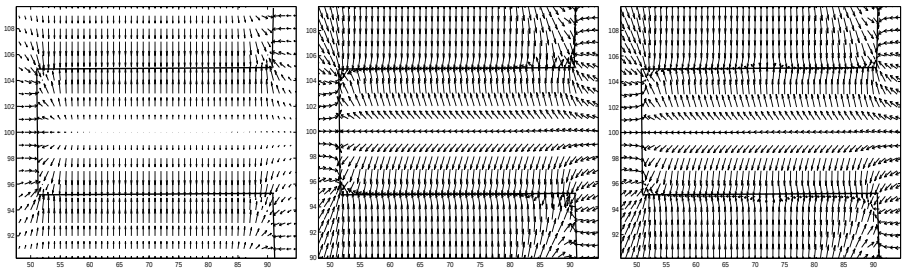


Fig. 3. The testing image used for the comparison: the boundary concavity and gap are indicated by dashed rectangles

did before the diffusion. Therefore, a snake will stop near the entrance of the concavity and cannot reach the bottom. However, by our normalized vector diffusion, the vectors in the concavity obviously changed their magnitudes and orientations and thus the snake can easily move into the bottom of the concavity (see *fig.4(b)* and *(c)*). The results by our normalized method are almost the same both for isotropic diffusion and for anisotropic diffusion.

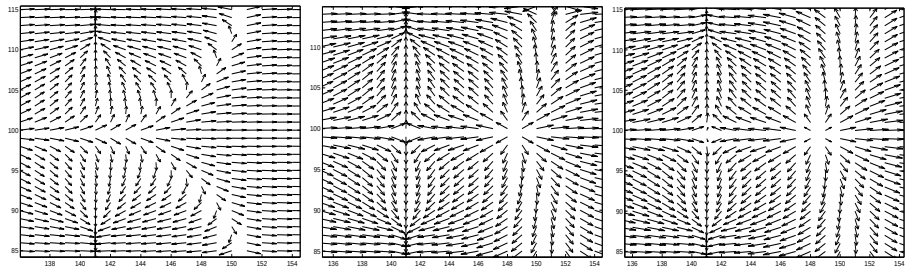


(a) By traditional GVF (b) By isotropic NGVD (c) By anisotropic NGVD

Fig. 4. The comparison between the traditional gradient vector diffusion and our normalized gradient vector diffusion for their behaviors on the boundary concavities

Fig.5 shows the results by our approach and traditional GVF near the boundary gap. As we can see from *fig.5(a)* traditional GVF has difficulty preventing the vectors on the boundary from being significantly influenced by the nearby boundaries and thus causes a problem such that the snake may move out of the boundary gap. However, our approach can avoid this problem. From *fig.5(b)* and *(c)* we can see that the vectors around the boundary gap point to the boundary from both sides.

The price we pay for the improvement of the performance in the above cases is that extra time is required to normalize the gradient vectors before or during each diffusion iteration. Furthermore, the traditional GVF method usually behaves better in noisy images than our NGVD. Based on these two considerations, a combined version of gradient vector diffusion can be used in the implementations to reduce time consumption as well as sensitivity to noises. The strategy is to



(a) By traditional GVF (b) By isotropic NGVD (c) By anisotropic NGVD

Fig. 5. The comparison between the traditional gradient vector diffusion and our normalized gradient vector diffusion for their behaviors on the boundary gap, which is located around the point (141, 100) in the left part of each figure

utilize the traditional GVF method in the first half (or more) of the entire diffusion process, while the normalized GVD method is used only in the second half (or less) of the diffusion process.

4 Image Segmentation

In this section we will briefly describe how to apply the *gradient vector field* to the image segmentation. As we have said before, the gradient vector field provides the active snakes an external force. However, in active snake model, we must answer the following questions: how to initialize the snakes? and what is the stopping criterion?

4.1 Seed Points

There are several ways to choose the initial snakes (or seed points): by hand, by "balloons", or by the locus of the zero-crossing of the Laplacian of the smoothed images (see [18] for a summary). In our case, we use *source* points (defined below) as our seeds. The *source* points can be automatically generated from the gradient vector field obtained from the diffusion process described in previous section.

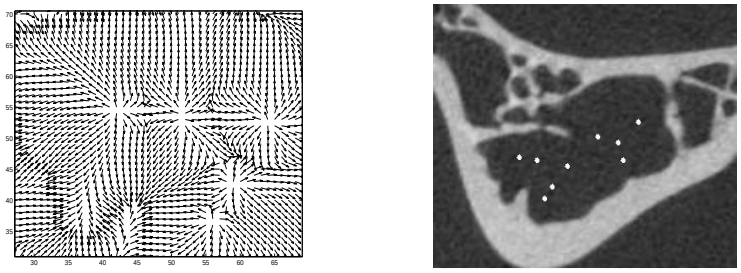
Definition: A point is called *source* if none of its neighbors points to it. In other words, a point A is a *source* if and only if, for any neighbor B of A:

$$\boldsymbol{v}_B \cdot \boldsymbol{BA} \leq 0,$$

where $\boldsymbol{v}_B$ is the gradient vector at B and $\boldsymbol{BA}$ is the vector from B to A.

A *source* point looks very like the origin of a spring where the water comes out. In *fig.6* we show an example of how a *source* looks like in our case of gradient vector flows. Some *source* points clearly appear in five different sites in the vector field shown in *fig.6(a)*. Usually there are more than one *source* even

in a homogeneous region, and the number of *sources* depends on two factors: (1) the geometric shape of the region's boundaries; (2) the number of iterations executed in the generation of the vector field. *fig.6(b)* illustrates the locus (the white small circles) of the *source* points distributed in a figure (only consider the central region).



(a) Example of source points (b) Seed points

Fig. 6. Take *source* points as our initial snakes (or seeds): the five *source* points shown on the left part of (*b*) correspond to the five *sources* of the vector field shown in (*a*)

4.2 Initial Segmentation

After we identify all seed points, initial snakes are generated by the following way: each isolated seed point is taken as an initial snake, and all connected seed points correspond to only one snake. Then we can let the initial snakes start to move. Their movements are directed by an external force that is determined by the previously generated vector field. The snakes stop moving by a very natural way, that is, if all the vectors on a snake are pointing to the opposite direction of the movement, then the snake stops. Finally when all the snakes stop, we obtain a partition of the original image such that each region corresponds to a snake. This is what we call initial segmentation, and next section will show some examples of initial segmentation on several different types of images. However, the initial segmentation does not give a satisfying result because there are too many small regions left. So we need to further refine the segmentation by merging the regions.

4.3 Region Merging

There have already been various approaches on region merging [19–21], most of which were based on the initial segmentation produced by the well-known *Watershed* method. Although our approach to generate the initial segmentation is different from *Watershed* method (the difference will be given in the following subsection), the basic idea for the region merging is the same: first, generate a *Region Adjacency Graph (RAG)* from the initial segmentation, and then merge

the most similar regions step by step until some criterion is satisfied. Different merging methods may have different ways to define the similarity. In our approach we will consider two factors to determine the similarity: one is the intensity difference between two adjacent regions, and the other is the average image gradient on the common boundaries between two adjacent regions. Two regions are said to be more similar if their intensity difference is smaller and/or the average image gradient on their common boundaries is higher. For these purposes we organize our data structures as follows:

```
struct Region:
{ int number;          /* number of points in this region */
  double intensity;    /* average intensity of this region */
  BOUND *boundary;     /* list of boundary points of this region */
  RGN *neighbor;       /* list of neighboring regions */
```

where the structure *BOUND* is defined by the *x-coordinate* and *y-coordinate* of the boundary point and a pointer to the next boundary point. The structure *RGN* contains the ID number of the neighboring region, the number of points on the common boundaries between the neighboring region and the region under consideration, the average image gradient on the common boundaries, the overall similarity between the neighboring region and the region under consideration, and finally a pointer to the next neighboring region.

4.4 Relationship with Watershed Method

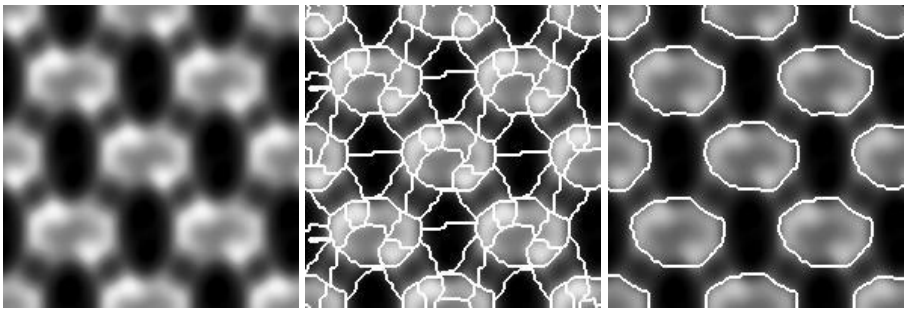
As mentioned above, *Watershed* is one of the well-known approaches to generate an initial segmentation (see [22]). This method usually begins with the image gradient map and takes the minimum of this map as the seeds. From this point *Watershed* method is very similar to our approach: both methods start from the image gradient map and the minimums in *Watershed* method look very like the *source* points of our approach. Furthermore, the geodesic distance transformation used in *Watershed* method can be thought of as a propagating process, which is analogous to the propagating process of gradient vectors in our approach. However, the geodesic distance transformation, like the *distance potential force* [5] we mentioned in the first section, makes the traditional *Watershed* method limited when dealing with the boundary concavities. Another advantage of our approach is its simplicity of generating the initial segmentation given a vector field over the image domain, because the vectors provide snakes the information of where to start or stop and how to move. However, it is not so straightforward in *Watershed* method to obtain the initial segmentation from a set of seeds.

5 Results

In the following we will give some examples of image segmentation by our region merging approach based on the normalized gradient vector diffusion (NGVD). *Fig. 7* shows an example of microscopy images, where the original image is shown

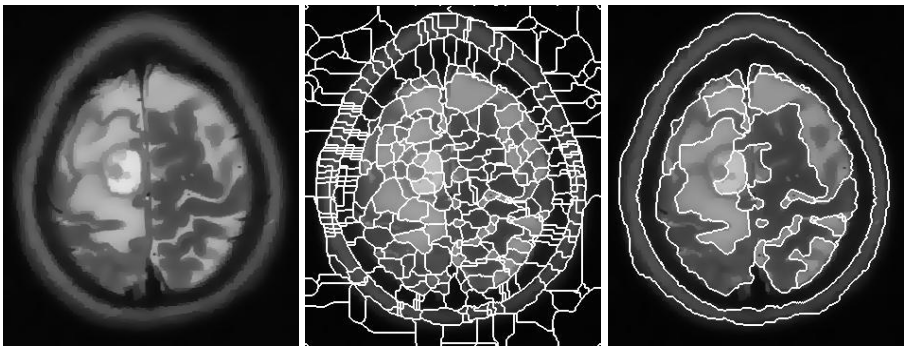
in *(a)*. After the initial segmentation there are totally 105 regions left (as seen in *(b)*). Finally 14 regions, including the background region, are remained after region merging. Although the original image is quite blurred, the cells are still correctly segmented. *Fig.8* shows an NMR image of brains. The initial segmentation generates 335 regions and after the region merging 25 regions are left.

In *fig.9* we show the results of segmentation on the same image using traditional GVF and our normalized gradient vector diffusion. It is very clear to see the different performance of these two methods on the boundary concavities shown in the left-top part of the images.



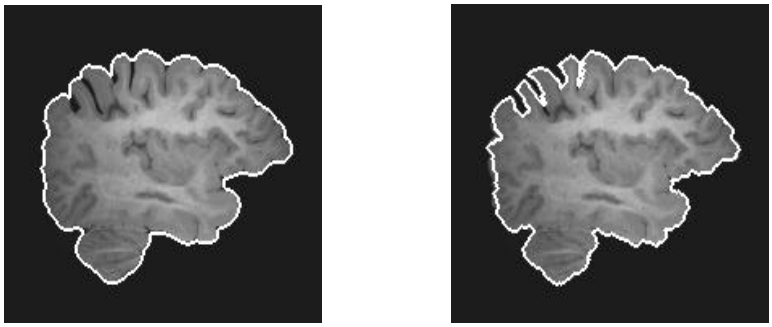
(a) The original image (b) Initial segmentation (c) After region merging

Fig. 7. Illustration of image segmentation by our NGVD-based region merging: there are 105 regions left after initial segmentation, and 14 regions left after merging



(a) The original image (b) Initial segmentation (c) After region merging

Fig. 8. Illustration of image segmentation by our NGVD-based region merging: there are 335 regions left after initial segmentation, and 25 regions left after merging



(a) By traditional GVF (b) By our NGVD

Fig. 9. The comparison between the traditional gradient vector diffusion and our normalized gradient vector diffusion for their behaviors on the image segmentation. Look closely at the difference between (a) and (b) in the left-top part of the images.

6 Conclusion

In this paper we present a new type of diffusion equations to generate *Gradient Vector Fields*. We normalize the vectors over the image domain before or during each diffusion iteration. The experiments show that our new approach not only can rectify the problems encountered in traditional snake models, but also can efficiently handle the cases of long-thin boundary concavities and boundary gaps as seen in traditional GVF method. Our approach can be applied to image segmentation with the help of region merging technique. The discussion also reveals an interesting relationship between our approach and the well-known *Watershed* method. Finally, it is quite straightforward to extend our method to 3D gradient vector diffusion and image segmentation.

References

1. Kass, M., Witkin, A., Terzopoulos, D.: Snake: active contour model. Int. J. Computer Vision. **1**(1987) 321-331
2. Xu, C., Prince, J.L.: Snakes, shapes, and gradient vector flow. IEEE Trans. Image Processing. **7(3)** (1998) 359-369
3. Xu, C., Prince, J.L.: Generalized gradient vector flow external force for active contour. Signal Processing - An International Journal. **71(2)** (1998) 131-139
4. Xu, C., Prince, J.L.: Gradient Vector Flow Deformable Models. Handbook of Medical Imaging, edited by Isaac Bankman, Academic Press. September, 2000
5. Cohen, L.D., Cohen, I.: Finite-element method for active contour models and balloons for 2D and 3D images. IEEE Trans. on Pattern Analysis and Machine Intelligence. **15(11)** (1993) 1131-1147
6. Terzopoulos, D., Witkin, A., Kass, M.: Constraints on deformable models: recovering 3D shape and non-rigid motion. Artificial Intelligence. **36(1)** (1988) 91-123
7. Malladi, R., Sethian, J.A., Vemuri, B.C.: Shape modeling with front propagation: A level set approach. IEEE Trans. on Pattern Analysis and Machine Intelligence. **17** (1995) 158-175

8. Leroy, B., Herliin, I., Cohen, L.D.: Multi-resolution algorithm for active contour models. In 12th Int. Conf. Analysis and Optimization of System. (1996) 58-65
9. Cohen, L.D.: On active contour models and balloons. CVGIP: Image Understanding. **53** (1991) 211-218
10. Davatzikos, C., Prince, J.L.: An active contour model for mapping the cortex. IEEE Trans. Medical Image. **14** (1995) 65-80
11. Abrantes, A.J., Marques, J.S.: A class of constrained clustering algorithm for object boundary extraction. IEEE Trans. Image Processing. **5** (1996) 1507-1521
12. Prince, J.L., Xu, C.: A new external force model for snakes. Proc. 1996 Image and Multidimensional Signal Processing Workshop. (1996) 30-31
13. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. IEEE Trans. on Pattern Analysis and Machine Intelligence. **12(7)** (1990) 629-639
14. McInerney, T., Terzopoulos, D.: A dynamic finite element surface model for segmentation and tracking in multidimensional medical images with application to cardiac 4D image analysis. Comput. Med. Imag. Graph.. **19** (1995) 69-83
15. Leymarie, F., Levine, M.D.: Tracking deformable objects in the plane using an active contour model. IEEE Trans. on Pattern Analysis and Machine Intelligence. **15** (1993) 617-634
16. Terzopoulos, D., Szeliski, R.: Tracking with Kalman snakes. in Active Vision, A.Blake and A.Yuille, Eds. Cambridge, MA: MIT Press, (1992) 3-20
17. Malladi, R., Sethian, J.A.: A real-time algorithm for medical shape recovery. in Proceedings of International Conference on Computer Vision. Mumbai, India. (1998) 304-310
18. Shah, J.: A common framework for curve evolution, segmentation and anisotropic diffusion. in Proceedings of International Conference on Computer Vision and Pattern Recognition: CVPR'96. June (1996) 136-142
19. Weickert, J.: Fast segmentation methods based on partial differential equations and the watershed transformation. P. Levi, R.-J.Ahlers, F. May, M. Schanz (Eds.), Mustererkennung 1998, Springer, Berlin. (1998) 93-100
20. Haris, K., Efstratiadis, S.N., Maglaveras, N., Katsaggelos, A.K.: Hybrid image segmentation using watersheds and fast region merging. IEEE Trans. on Image Processing. **7(12)** (1998) 1684-1699
21. Moscheni, F., Bhattacharjee, S., Kunt, M.: Spatiotemporal segmentation based on region merging. IEEE Trans. on Pattern Analysis and Machine Intelligence. **20(9)** (1998) 897-915
22. Vincent, L., Soille, P.: Watersheds in digital spaces: an efficient algorithm based on immersion simulations. IEEE Trans. on Pattern Analysis and Machine Intelligence. **13(6)** (1991) 583-598