IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 6, June 2015.
www.ijiset.com
ISSN 2348 – 7968

A Novel Geometric Snake Model for Medical Image Segmentation

Soppannappa[1], Virupakshappa[2]

[1]P.G.Student, Department of Computer Science & Engineering,
Appa Institute of Engineering & Technology, Kalaburgi, Karnataka, India.
[2]Assosciate Professor, Department of Computer Science & Engineering,
Appa Institute of Engineering & Technology, Kalaburgi, Karnataka, India

***Abstract*—** In this work, we employ the new geometric active Contour models formulated in [25] and [26] for edge detection and segmentation of magnetic resonance imaging (MRI), computed tomography (CT), and ultrasound medical imagery. Our method is based on defining feature-based metrics on a given image which in turn leads to a novel snake paradigm in which the feature of interest may be considered to lie at the bottom of a potential well. Thus, the snake is attracted very quickly and efficiently to the desired feature.

***Keywords*— Active contours, active vision, edge detection, gradient flows, segmentation, snakes.**

Introduction

Medical image segmentation algorithms often face difficult problems such as poor image contrast, noise, and missing or diffuse boundaries. Medical image segmentation algorithms face difficult problems such as poor image contrast, noise, and missing or diffuse boundaries. For example, consider tissue boundaries in medical images consider to be spread (resultant of patient movements), missing (because of low SNR of the acquisition-apparatus), or non-existence (due to blended with similar surrounding tissues).Under such situations, without a prior-model to constrain the operation of segmentation, most of the algorithms (including known intensity- and curve-based methods) fail-mostly because of the under-determined nature of the image segmentation process.
Similar difficulties arise with other digital imaging applications as well and they also delay the segmentation of the given input image. Here these image segmentation problems require the incorporation of as much prior data as possible to support the segmentation algorithms extract the feature of tissue of interest. We implement such an algorithm in this paper. In particular, we obtain a model-based, implicit parametric depiction of the segmenting curve and calculate the parameters of such representation via gradient descent to reduce energy functional for medical image segmentation process.

A Relationship to Previous Work

The proposed work shares common view of Eigen-vectors and that shows variations from the mean shape along with a large number of specific model-based image segmentation algorithms in the classic literature.
In the previous work the author employed an “average shape” to serve as basis of the shape earlier name in their geometric active contour technique and also previously developed a parametric point distribution model for drawing the segmenting with the help of linear combinations of segmented images.

Point distribution model is having shape and pose parameters, which are used to determine or specify point to strong image gradients. In the later work the authors (Entland and Sclaroff) explained a variant of above mentioned approach. And authors Staib and Duncan proposed a parametric point model relative to an elliptic Fourier segmentation of the specified landmark-points. Here the features or values of the curve are calculated to optimize the values or match between segmenting curve and the gradient of the corresponding image. In the later work the author by the name Chakrabortyet al. extended this method to a hybrid segmentation model that integrates both gradient and region-homogeneity information of image. And recently, the authors Wang and Staib implemented a statistical point model for segmenting curve object by applying principal component analysis (PCA) method to the covariance matrices that collect the statistical variations of the specified landmark points. They formulated or calculated the edge-detection and correspondence-determination issue in a maximum a posteriori Bayesian-framework.
Pose and shape parameters are calculated using image gradient method within the framework that describe the segmenting curve.
The author Leventonet al. worked on a technique called a less restrictive model-based segment. He combined shape data as a prior model to limit the flow of the geodesicactive contour. Their initial parametric shape model is obtained by performing PCA on a set of signed distance-maps of the training-shape. The segmenting curve results according to following competing forces as:
1) General gradient force of the image, and
2) And specific force exerted by the estimated shape
Here image gradients and current position of curve are used to calculate parameters of the shape.
The working of project is related closely to region-based active contour models. Generally these region-based active

ISSN 2348 – 7968

contour techniques support a number of pre-processing properties over gradient-based techniques for image segmentation, containing greater robustness-to-noise (by nullify derivatives of image intensity) and starting contour placement.
Our unique general editing techniques are founded upon the embedding scalar field associated with any area specified as the point-set surface. We used an embedded scalar field incorporated with any region with point-set surface to found novel editing techniques. In comparison with the real, specific point-based surface representation, defined scalar field-driven implicit representation and normal level sets have been proven as a very strong approach. This approach free of parameterization artifacts, and also handle arbitrary topology specified and complexity geometry easily.
During the phase of shape sculpting process the implicit representation process allow the proper topological change without much ambiguity.

Literature Survey

2.1. *Original Snake Model by Kass*

The concept of active contours was introduced by Kass, in the seminal paper "Snakes: Active Contour Models" [Kass- 88]. The paper was extremely influential and has since then been a major topic for research As we have already discussed a snake is a parametric curve which tries to move into a position where its energy is minimized. Kass et al. introduced the following energy functional for calculating the snake energy.

$$E_{snake} = E_{internal} + E_{external} + E_{constraint}$$

The snake energy consists of three terms. The first term E_{int} represents the internal energy of the snake while the second term Eimg denotes the image forces, the last term Econ gives rise to external constraint forces. The sum of the image forces E_{img} and the external constraint forces E_{con} is also simply known as the external snake forces, denoted by E_{ext}.

Internal Energy (E_{int}) depends on the intrinsic properties of the curve and is the sum of elastic energy and bending energy

$$E_{int} = E_{elastic} + E_{bending} = \int_s \frac{1}{2}(\alpha |v_s|^2 + \beta |v_{ss}|^2) ds$$

External energy (E_{ext}) of the contour is derived from the image so that it takes on its smaller values at the function of interest such as boundaries. Define a function $E_{image(x,y)}$ so that it takes on its smaller values at the features of interest, such as boundaries

$$E_{ext} = \int_s E_{image}(v(s)) ds$$

3. Gradient Vector Flow Model (GVF)

As discussed earlier one of the major problems faced during the implementation of active contour model was the poor convergence of this snake, this is because the forces point horizontally in opposite direction. Another weakness of the traditional snake model is that it has a limited capture range as seen in figure 1, this can be explained by the simple theory that the magnitudes of the external forces die out quite rapidly away from the object boundary. The boundary localization will become less accurate and distinct. To overcome this problem we introduce the GVF snake model.
The gradient vector flow snake is used in order to increase the capture range and improve the snakes ability to move into boundary concavities. The capture range of the original snake is generally limited to the vicinity of the desired contour. Furthermore the original snake has problems with moving into concave regions e.g. moving into the concave region of an U-shaped object, the gradient vector flow snake handles these problems by introducing a new external force. By minimizing an energy function we can derive a new vector field by using this external. We call this vector field as gradient vector flow fields

Gradient Vector Flow: - The GVF field is defined to be a vector field

$$\mathbf{V(x,y)} = (u(x,y), v(x,y))$$

Force equation for GVF snake is,

$$\alpha v_{ss} - \beta v_{ssss} + V = 0$$

V(x,y) is defined such that it minimizes the energy functional,

$$E = \iint \mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla f|^2 |V - \nabla f|^2 \, dxdy$$

f(x,y) is the edge map of the image.

The above equations are solved iteratively using time derivative of u and v. These equations provide further intuition behind the GVF formulation. We note that in the homogenous region the second term in both regions is zero because the gradient of f(x, y) is zero.

After the application of the GVF snake model we can see that the snake can move towards object that are too far away from the boundary. Secondly, that now snakes can move into boundary concavities. Hence we can conclude that both the problems of the traditional snake models have been successfully resolved.

ISSN 2348 – 7968

4. Greedy Snake Algorithm

A greedy algorithm makes locally optimal choices, hoping that the final solution will be globally optimum. It is a feature extraction technique widely used in image segmentation. It works like an elastic band being stretched around and object and then being released. Initial points defined around feature to be extracted explicitly defined and the Pre-defined number of points are generated.
Points are calculated by an Iterative Process**:**

- Energy function for each point in the local neighborhood is calculated
- The point is moved to the next point with lowest energy function
- This process is repeated for every point
- Iteration is done until termination condition met
- Defined number of iterations
- Stability of the position of the points

The final step in the iteration of the greedy snake algorithm consists of checking whether the number of points moved in the iteration is below the threshold. This is used as a stopping criterion as the snake is presumed to have reached minimum energy when most of the control points have stopped moving.

Proposed Work

Image Snakes

An image snake is an active contour on an image which moves by minimizing an energy functional *Esnake*. The functional *Esnake* contains internal and external energy terms. The internal energy *Espline* concerns with the smoothness of a snake and minimizing *Espline* makes a snake act like a spline curve. The external energy *Eimage* is related to image features and has local minimums at the features. Hence, a snake obtained by minimizing *Esnake* is a smooth curve that passes through a feature in an image. In this section, we summarize the energy minimization process of a snake, which will be referred to in later sections. The details of the process can be found in reference. The position of a snake is represented in a parametric form,

$$v(s) = (x(s); y(s)),$$

where $s \in [0;1]$.

Then, the energy functional *Esnake*(*v*) can be written as

Esnake(*v*) = Z 10_*Espline*(*v*)+*Eimage*(*v*)_*ds*:

The internal spline energy *Espline*is defined by

Espline= (α(*s*)k*v*′(*s*)k2 +β(*s*)k*v*′′(*s*)k2)=2;

which consists of the first- and second-order derivatives of *v*(*s*) with weights α(*s*) and β(*s*). The external feature energy *Eimage* is determined by the types of features to be detected. For example, *Eimage*= □j∇*I*(*x*; *y*)j2 can be used to detect edges in an image, where ∇*I*(*x*; *y*) is the image gradient at a pixel (*x*; *y*). In implementation, a snake *v*(*s*) is represented by a sequence of *n* sample points; that is, *v*(*i*) = (*x*(*i*); *y*(*i*)) for *i* = 1; : : :;*n*. Then, the energy *Esnake* can be minimized by solving a matrix equation;

$$\mathbf{Ax}+\mathbf{fx}(\mathbf{x};\mathbf{y}) = 0$$
$$\mathbf{Ay}+\mathbf{fy}(\mathbf{x};\mathbf{y}) = 0 \quad (3)$$

In Eq. (3), the matrix **A** comes from the finite difference method that approximates the derivatives of *v*(*s*). The column vectors **x** and **y** consist of *x* and *y* positions of the snake points, respectively. The column vectors **fx**(**x**;**y**) and **fy**(**x**;**y**) represent partial derivatives of *Eimage* with respect to **x** and **y**, respectively. The matrix equation in Eq. (3) cannot be directly solved for **x** and **y** because **fx** and **fy** depend on **x** and **y**. Hence, an iterative process is used in which **x***t* and **y***t* converge to the solution of Eq. (3) as time *t* progresses. Assuming that **fx** and **fy** are constant within a unit time interval γ, **x***t* and **y***t* can be obtained from **x***t*□1 and **y***t*□1 by solving two linearequations;

$$\mathbf{x}t= (\mathbf{A}+\gamma\mathbf{I})□1(\gamma\mathbf{x}t□1□\mathbf{f}x(\mathbf{x}t□1;\mathbf{y}t□1))$$
$$\mathbf{y}t= (\mathbf{A}+\gamma\mathbf{I})□1(\gamma\mathbf{y}t□1□\mathbf{f}y(\mathbf{x}t□1;\mathbf{y}t□1)) \quad (4)$$

In summary, after the initial position of a snake is specified by the user, the position is incrementally updated by repeatedly solving the linear equations in Eq. (4). In each iteration, the snake approaches a nearby feature by moving inthe direction of (□**fx**;**fy**), which reduces the external feature energy *Eimage*. In the movement, the shape of the snake is kept smooth by solving linear equations that contain the matrix **A** derived from the internal spline energy *Espline*.

4. Geometric Snakes

4.1. Representation

A geometric snake for a triangular mesh is represented by*v*(*s*) = (*x*(*s*); *y*(*s*); *z*(*s*)), where *s* is the parameter such that *s* 2 [0;1]. Similar to Eq. (1) for an image snake, the energy functional of a geometric snake consists of the internal spline energy *Espline* and the external feature energy *Emesh*;

Esnake(*v*) = Z 10_*Espline*(*v*)+*Emesh*(*v*)_*ds* (5)

In Eq. (5), the spline energy *Espline* is concerned with the smoothness of a geometric snake and can be computed in the same way as that of an image snake with Eq. (2). The feature energy *Emesh* is determined by the definition of a feature on a

ISSN 2348 – 7968

mesh and should have local minimums at the features. In this paper, we use the normal variations of the neighbour faces to determine the feature energy at a vertex of a mesh. The feature energy at the internal points of a face is computed by linear interpolation. We explain the details of the feature energy *Emesh* in Section 6.In implementation, similar to an image snake, we represent a geometric snake $v(s)$ by a sequence of n sample points $v(i)$ on the mesh. With this representation, the line segment connecting two sample points may not lie on the mesh surface. For simplicity, in the energy minimization process, we only consider the sample points that lie on the mesh surface. When the final position of a geometric snake is determined, we project the line segments between sample points to obtain a piecewise linear curve on the mesh surface.

4.2. Movement

After its initial position on a mesh is specified by a user, a geometric snake can detect a nearby feature by minimizing Eq. (5).For the minimization process, we can use the same numerical technique as for an image snake, summarized in Section 3. That is, the matrix equations in Eq. (3) is obtained for $v(s) = (x(s); y(s); z(s))$ and the position is incrementally updated by repeatedly solving the linear equations in Eq. (4).In this approach, the updated position of a geometric snake will be in the 3D space near the old position and is not guaranteed to lie on the surface of the mesh. However, since the feature energy *Emesh* is defined only on the mesh surface, it is impossible to solve Eq. (4) for the next step if the current position of a geometric snake is not on the mesh surface. A simple solution to resolve this problem is to project a geometric snake onto the surface of a mesh every time when its position is updated by solving Eq. (4). We implemented and tested this approach, but the approach was found to be computationally expensive because the projection must be performed at every update. This drawback may prohibit a geometric snake from having a speed fast enough to be used as an interactive tool. In this paper, to constrain a geometric snake onto the surface of a mesh, we use mesh parameterization that embeds a part of the mesh onto a 2D plane. With the parameterization, a geometric snake

$$v(s) = (x(s); y(s);$$

$z(s))$ is mapped to a curve $v\_(s) = (x\_(s); y\_(s))$ in the plane. Then we can use the same equations in Section 3 to compute the position updates of the curve $v\_(s)$ in the plane. Since the feature energy *Emesh* is a scalar field on the surface of a mesh, it can be easily mapped onto the embedding plane. The updated position of the geometric snake $v(s)$ is obtained by mapping the curve $v\_(s)$ back onto the mesh surface. Smoothness of $v(s)$ can be maintained by the spline energy *Espline* applied to $v\_(s)$.

If we embed the whole mesh onto a 2D plane, a single parameterization is sufficient regardless of the updated positions of the curve $v\_(s)$ in the energy minimization process. However, this global parameterization requires more computation than the local one used in this paper. More importantly, when a large mesh is parameterized, a severe distortion may happen in the embedding. Since the smoothness of a geometric snake $v(s)$ comes from that of the curve $v\_(s)$,the distortion should be minimized in the embedding. By using local parameterization, we can avoid large distortion and preserve the smoothness of a geometric snake.

4.3. Overall process

The overall process for feature detection on a mesh with a geometric snake proposed in this paper is as follows. Fig.1 summarizes the process.

1. The initial position of a geometric snake $v(s)$ is interactively specified by the
2. The local region containing $v(s)$ is determined on the mesh .
3. The local region is embedded onto a 2D plane while minimizing the distortion.
4. Snake $v(s)$ is converted to a curve $v\_(s)$ on the plane, Specify an initial snake position.
5. Derive the feature energy on the 2D curve $v^*(s)$
6. Update the position of the2D curve $v^*(s)$
7. Convert the snake to a 2Dcurve $v^*(s)$

1. Parameterize the local region

2. Interactively edit the snake $v(s)$ Current snake position $v(s)v(s)$

3. Update the position of the snake by using $v^*(s)$

4. Obtain the local region containing the snake $v(s)$

5. The external feature energy *Emesh* is derived at the points on the curve $v\_(s)$.

6. The updated position of $v\_(s)$ in the plane is determined by solving the linear equations in Eq. (4). If the position update needs to be repeated, go to step 5.

7. The final position of the curve $v\_(s)$ is mapped onto the mesh surface to update the current position of snake $v(s)$.

8. If the updated current position of snake $v(s)$ is not satisfactory, the user can interactively edit the parts of $v(s)$. If the user wishes, proceed to step 2.

In the process, the position update of the curve $v\_(s)$ by steps 5 and 6 can be iterated in the fixed number of times or until a part of $v\_(s)$ moves out of the embedded local region of the mesh. Note that the local parameterization is performed only once in step 3 for the iteration. If the updated position of the snake $v(s)$ obtained from the result of the iteration does not

ISSN 2348 – 7968

capture the desired feature yet, the user can reapply the process to *v*(*s*) starting from step 2.

5. Movements of Geometric Snakes

In this section, we present the component techniques that are used to move a geometric snake on the surface of a mesh. We also explain the user interaction techniques that can guide a geometric snake to capture a desired feature.

5.1. Partitioning of a snake

In general, the local surrounding region of a geometric snake is much smaller than the entire surface of a mesh. However, if a snake widely stretches over the surface (see Figs. 10(b)and 10(c)), the surrounding region to be parameterized becomes large, which increases the computation time. Moreover, in the case of a closed snake, such as shown in Fig.10(b), the surrounding region contains a hole, which complicates the parameterization process. To reduce the size and avoid a hole of a parameterized region, we partition a geometric snake into two or more curve segments. Then, the steps 2 through 7 of the overall process in Fig. 1 are applied
To each curve segment as if it is a single geometric snake. Partitioning a geometric snake into curve segments may introduce a discontinuity into the shape of the snake. Without loss of generality, we assume that a snake is partitioned into two segments *vA*(*s*) and *vB*(*s*) (see Fig. 2). The snake may become discontinuous at the border point *pj* when the positions of *vA*(*s*) and *vB*(*s*) are independently updated. To overcome this problem, we recompute the snake position around *pj*by using a curve segment *vC*(*s*) that partially overlaps with *vA*(*s*) and *vB*(*s*). In other words, we first update the positions of *vA*(*s*) and *vB*(*s*) while fixing the borderpoint *pj*. Then, the position of *vC*(*s*) is updated with its endpoints *pi* and *pk*fixed at the previously computed positionsin *vA*(*s*) and *vB*(*s*). This adjustment of the snake positionaround *pj*makes the two segments *vA*(*s*) and *vB*(*s*) be connectedsmoothly.
vAvBvC pipjpk

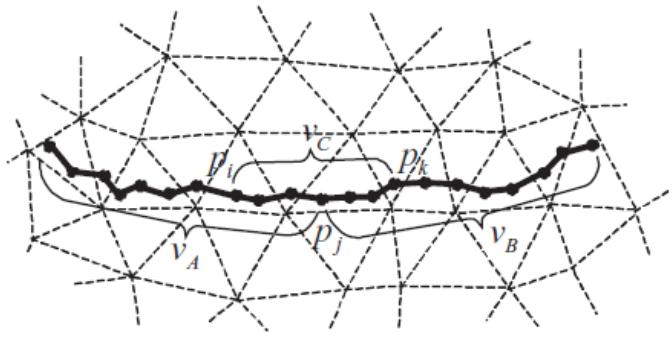


Figure 2: *Partitioning of a geometric snake*

It may be possible to partition a geometric snake into curve segments so that distortions are minimized in parameterizing the surrounding regions of the segments. In most cases, however, since the shape of a snake is smooth due to the internal energy *Espline*, a simple partitioning may not incur severe distortions in the embedding of a surrounding region. In this paper, we partition a snake into curve segments that contain the same number of snake sample points. If the current shape of a snake contains sharp corner points with small incident angles, the snake is partitioned to place the corner points at the middles of curve segments. Rather than arbitrarily placing the corner points, for example, at the borders of curve segments, this approach enables the shape of a snake to rapidly smoothen around the corner points in a position update. A sharp corner point may happen in the initial positioning and interactive editing of a geometric snake. Usually, a snake contains either few, or most often, no sharp corner points, and two or at most several curve segments are obtained from the partitioning.

5.2. Local surrounding region of a snake

After partitioning a snake, we determine the local surrounding region on the mesh for each segment, which will be embedded onto a 2D plane. The selection of a too small region decreases the number of position updates possible with a single parameterization. If a large region is selected, it will increase the computation time and distortions in the embedding. To obtain the surrounding region with a reasonable size, we first obtain the face sequence *F* on the mesh on which the line segments between snake sample points are projected. Then, the surrounding region is determined as the set of faces which contains the face sequence *F* and the neighbour faces sharing a vertex with a face in *F* (see Fig. 3). Finally, we fill the holes that may exist in the resulting region.

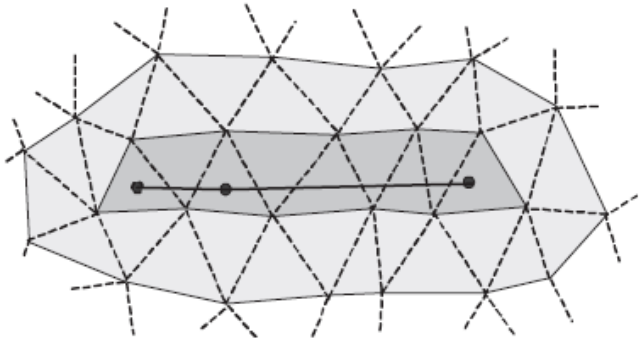
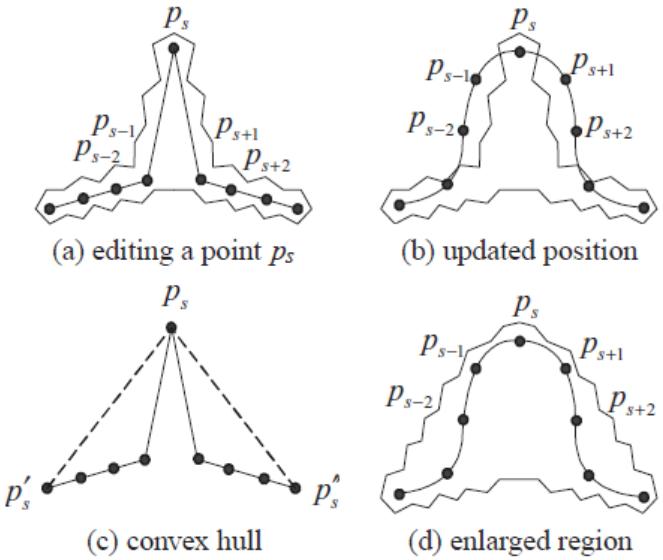


Figure 3: *Surrounding region of a geometric snake*

A geometric snake is initially placed near a feature and its position is incrementally updated in the energy minimization

ISSN 2348 – 7968

process. Hence, with the proposed method to select the local surrounding region, several position updates are possiblewith a single parameterization, as demonstrated in Section7. Also, the selected region contains a small number of triangles and can be parameterized quickly. In interactive editing of a geometric snake which will be explained in Section 5.5, the user can move a snake sample point *ps* far from the current position.



See Fig. 4(a) for an example. Then, the parts of the snake near the point *ps* will be rapidly changed to maintain the smoothness in the following position update. In this case, the updated position of the snake may not belong to the local region determined from the current position, as shown in Fig. 4(b). To avoid such a case, we first map the current snake curve to a 2D plane and find the two vertices *p*0*s* and *p*00 *s* adjacent to *ps* in the 2D convex hull of the snake. Then, the local region is enlarged to include the projections of two 3D line segments, from *pstop*0*s* and *p*00*s* , onto the mesh surface.

5.3. Parameterization of a local region

Since geometric snakes are an interactive tool to detect features, the parameterization of the selected local region should be performed quickly. In addition, the distortions in the parameterization should be minimized so that the positions of a geometric snake *v*(*s*) and the 2D curve *v*_(*s*) are properly mapped to each other. In this paper, for parameterization, we use the convex combination approach extended with virtual boundaries 18.The convex combination approach 8 obtains the parameterization of a mesh by solving a linear system. The approach runs fast and generates an one-to-one embedding. However, the approach requires the boundary of a mesh to be fixed onto a convex polygon, which causes high distortion near the boundary. The extension 18 of the approach used in this paper reduces the distortions by using virtual vertices attached to the real boundary.

Fig. 5(a) shows the surrounding region of a curve segment from a snake that is placed near an ear. Fig. 5(b) shows the embedding result of the region generated by the convex combination approach with a virtual boundary.(a) surrounding region (b) embedding of the region

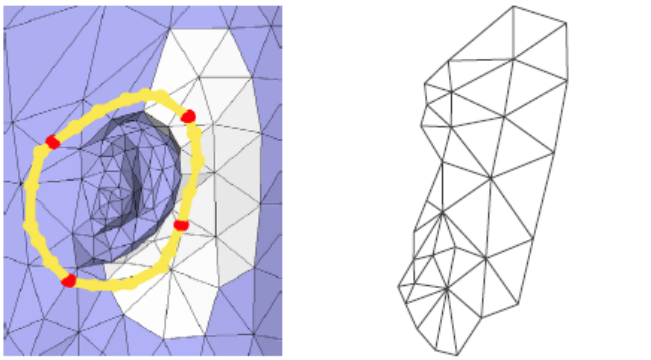


Figure 5: *Parameterization of a surrounding region*

5.4. Update of the snake position

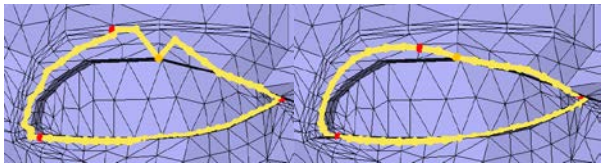
After the surrounding region is embedded onto a 2D plane,we convert the snake *v*(*s*) to a curve *v*_(*s*) in the plane. Asnake point inside a 3D triangle is mapped to a point in the corresponding 2D triangle by using barycentric coordinates. Then, the position of the curve *v*_(*s*) is incrementally updated by iteratively solving the linear equations in Eq. (4).The iteration continues until the curve *v*_(*s*) moves out of the embedded region or the preset number of iterations are performed. The final position of the curve *v*_(*s*) is mapped back onto the mesh surface to determine the updated position of the snake *v*(*s*), again by using barycentric coordinates.

5.5. User Interaction

Our geometric snake model provides three user interaction techniques: snake initialization, point editing, and point fixing. The snake initialization is used to specify the initial position of a snake. When the user selects a sequence of vertices on a mesh near a desired feature, a closed or open curve is automatically built by connecting adjacent vertices in the sequence with the shortest paths through the edges of the mesh. With point editing, the user can guide a point of a snake to a desired feature. For example, in Fig. 6, the user wants to detect the eyelid using the snake, but some snake points have been attracted to the eyebrow. Then, the user can edit a snake point to change its position from the eyebrow to the eyelid, as shown in Fig. 6(a). After the editing, the snake points around the edited point moves together toward the eyelid due to the spline energy *Espline*. Fig. 6(b) shows the resulting position from which the snake can finally capture the eyelid. The point fixing allows a point of a snake to be fixed ata specific position on the mesh in the energy minimization process. For example, in Fig. 6, the edited snake point is fixed at the

ISSN 2348 – 7968

moved position after editing. The point fixing is useful to protect snake points that already capture parts of a desired feature from position changes in minimizing the spline energy *Espline*.



(a) editing a point (b) updated position

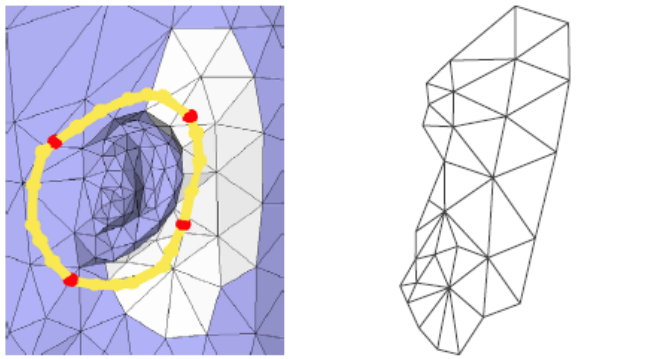
6: *Editing a snake*

See Fig. 4(a) for an example. Then, the parts of the snake near the point *ps* will be rapidly changed to maintain the smoothness in the following position update. In this case, the updated position of the snake may not belong to the local region determined from the current position, as shown in Fig. 4(b). To avoid such a case, we first map the current snake curve to a 2D plane and find the two vertices *p*0*s* and *p*00 *s* adjacent to *ps* in the 2D convex hull of the snake. Then, the local region is enlarged to include the projections of two 3D line segments, from *ps*to*p*0*s* and *p*00*s* , onto the mesh surface.

5.3. Parameterization of a local region

Since geometric snakes are an interactive tool to detect features, the parameterization of the selected local region should be performed quickly. In addition, the distortions in the parameterization should be minimized so that the positions of a geometric snake *v*(*s*) and the 2D curve *v*_(*s*) areproperly mapped to each other. In this paper, for parameterization, we use the convex combination approach extended with virtual boundaries 18.The convex combination approach 8 obtains the parameterization of a mesh by solving a linear system. The approach runs fast and generates an one-to-one embedding. However, the approach requires the boundary of a mesh to be fixed onto a convex polygon, which causes high distortion near the boundary. The extension 18 of the approach used in this paper reduces the distortions by using virtual vertices attached to the real boundary.

Fig. 5(a) shows the surrounding region of a curve segment from a snake that is placed near an ear. Fig. 5(b) shows the embedding result of the region generated by the convex combination approach with a virtual boundary.(a) surrounding region (b) embedding of the region



(a) surrounding region (b) embedding of the region

Figure 5: *Parameterization of a surrounding region*

6. Feature Energy for Geometric Snakes

6.1. Feature energy definition

To define external feature energy *Emesh* for a mesh, we must have a numerical representation of features which has a local minimum at a feature. In image snakes, for example, image gradients at pixels are used to define the features onan image 14. In the same manner, we can consider the features on a mesh as the points on the mesh where the mesh property changes drastically. For example, peaked corners, sharp edges, and color discontinuities can be used for mesh features. In this paper, we use normal variations of the neighbour faces to determine the feature energy at a vertex of a mesh. To compute the normal variation, we adopt the opening angleof the normal cones at a vertex 16. Then, the feature energy *Emesh* at a vertex *v* is defined by

$$Emesh(v) = \min f(nTvnf) \tag{6}$$

In Eq. (6), *nv* is the normal at a vertex *v* and *nf* is the normal of a face *f* adjacent to the vertex *v*. With this definition, the feature energy at a vertex *v* has values between 0 and 1.The feature energy at points inside a face is determined by linearly interpolating the energy at vertices of the face. In addition to normal variation, we experimented with thediscrete curvature norm 15 and the quadric error metric 9 to define mesh features. With the discrete curvature norm, we can consider a feature as a point at which the sum of principal curvatures κ1 and κ2 is large. By using the quadric error metric, we can define the feature energy at a vertex *v* as the negation of the maximum of the edge collapse errors for the edges adjacent to *v*. As shown in Fig. 7, all three definitions of feature energy give similar results and have local minimum sat features such as edges and corners. In this paper, we chose the normal variation for feature definition because of its simplicity.

6.2. Feature energy on a face

Before starting a feature detection process, we compute the feature energy values for the vertices of a mesh by using Eq.
(a) fan disk model (b) normal variation

ISSN 2348 – 7968

(c) Discrete curvature (d) quadric error

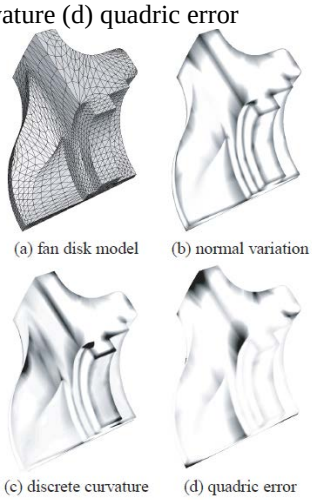


Figure 7: *Comparison of feature energy definitions*

(6). The feature energy inside a face is determined by linear interpolation, as mentioned before. However, to update the position of a snake, we do not need the feature energy itself but the derivatives of feature energy, that is, **fx** and **fy** in Eq. (4). Inside a face, the derivatives **fx** and **fy** are constant because the feature energy linearly varies. Therefore, in the energy minimization process to update the position of the curve $v\_(s)$, the derivatives **fx** and **fy** at a point p on $v\_(s)$ can be obtained as follows:

1. The feature energy values of the vertices in the surrounding region are mapped onto the embedding plane.
2. The derivatives **fx** and **fy** are computed and saved for each face in the plane as a pre-processing step.
3. During the energy minimization process, the values saved for the face containing the point p are used for the derivatives **fx** and **fy** at p. With this approach, the derivatives **fx**and **fy** need not be dynamically computed in the energy minimization process, which reduces the computational overhead.

6.3. Feature energy smoothing

In order to enable a distant feature to attract a snake and to reduce the effects of noises, image snake models use low pass filters that blur feature energy values at image pixels14; 10. Similarly, we can smooth the feature energy by weighted averaging the energy value of a vertex with those of neighbor vertices. With smoothing, a geometric snake can be less affected by noises and capture a strong feature that is distant from the initial position.

APPLICATIONS TO MEDICAL IMAGERY

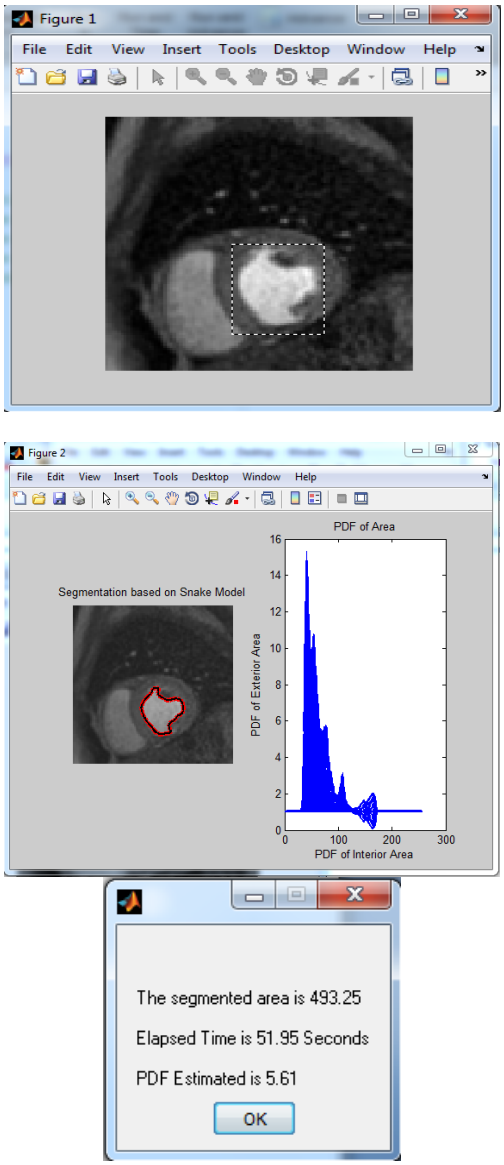
We will now apply the active contour model derived above to some medical imagery. The numerical methods we have used come from the level set evolution techniques developed by Osher–Sethian [37], [38], [45], [46], [47], and Malladi*et al*. [32]. To speed up the evolutions, we have used the local versions of these algorithms precisely as described in [1]. The equations described in this paper have been coded for thecase of active contours on 2-D images. In Section A of the Appendix, we make some remarks about this method. or 2-D active contours, the evolution equation as derived above is [(8)]div where is a constant inflation force and div is the curvature of the level sets of . This equation describes a propagating front, and we are interested in its propagation in the plane of an image. It is known that a propagating front may not remain smooth at all times (for example, it may cross itself). For evolution beyond the discontinuities the solutions are required to satisfy an entropy condition to ensure that the front remains physically meaningful at all times. The discrete approximations to the spatial derivatives are thus, derived from the entropy condition. Osher and Sethian [38] have given such entropy satisfying schemes and these have been used successfully in shape modelling [32]. We can regard a decomposition of our speed where is regarded as the constant passive advection term and the curvature is the diffusive term of the speed function. The inflation part in (8), i.e., is approximated using upwind schemes. The diffusive part, i.e., isapproximated using usual central differences [32].There is several stability considerations for the choice of the step sizes. In [32], it is noted that for the evolution equation used in that work the requirement is .Fig. 3. Contour extraction from MRI heart image via snake.

Therefore, if small spatial step sizes are used, it forces a small time step and the resulting evolution can be very slow. One possibility for speeding up the evolution is to use a larger inflationary force and move the front faster (recall the advection term causes a constant contraction/expansion of the front). However, in our experience with using the approach in [32] this results in large motion of the front causing “overshooting” of the edge of the feature of interest in the image, because might not be rigorously zero on the desired contour. This problem is resolved by the evolution in (8) in which has behaviour similar to a doublet near an edge. Thus, it exerts a “stronger” stopping effect and arrests the evolution of the contour close to an edge. In our simulations, we have observed that this arresting behaviour of the term allows use of large inflationary forces, resulting in features being extracted in relatively fewer time steps.

RESULTS

After implementing the proposed system on Matlab platform the results obtained are as follows:

ISSN 2348 – 7968



CONCLUSION & FUTURE SCOPE

We have presented a new segmentation framework based on the level set for Medical Image segmentation. The framework incorporates not only prior shape knowledge through the KDE method, but also local geometrical features through Mean Curvature Energy model, into the level set segmentation. Experimental results on MRI, CT and Ultrasound images, validated with ground truth, demonstrate the effectiveness of proposed framework. The framework has achieved much higher segmentation accuracy than existing methods, such as Chan–Vese and Caselles, region growing, and graph cut. And we achieve the better Area Estimation with reduced time by interfacing KDE with Mean Curvature. Ongoing research includes integrating the segmentation framework into a system for detection and Classification of some diseases using Medical images like Tumor, Spinal Fraction, Stroke, etc., .

REFERENCES

1. D. Adalsteinsson and J. Sethian, “A fast level set method for propagating interfaces,” J. Computat. Phys., vol. 118, pp. 269–277, 1995.
2. _L. Alavareez, F. Guichareed, P. L. Lions, and J. M. Moreelli, “Axiomes et equations fondamentales du traitement d‘images,” C. R. Acad. Sci., vol. 315, pp. 135–138, 1992.
3. “Axioms and improved fundamental equations of digital image processing,” Archie.Rationnalle Mech., Anal., vol. 123, 1993.
4. “Axiomatization et nouveaux operateurs de la morphologie mathematique,” C. R. Acad. Sci., vol. 315, pp. 265–268, 1992.
5. L. Allavarreze, P. L. Lionsi, and J. M. Morel, “Image selective smoothing and edge detection with a general nonlinear diffusion,” SIAM J. Numer. Anal., vol.29, pp. 845–866, 1992.
6. L. Alvarez and J. M. Morel, “Formalization and computational aspects of image analysis,” Dept. Inform. Syst., Univ. de las Palmas de Gran Canaria, Rep. 0493, 1993.
7. A. Blake and A. Yuille, Active Vision,. Cambridge, MA: MIT, 1992.
8. V. Caselles, F. Catte, T. Coll, and F. Dibos, “A geometric model for active contours in image processing,” Numerische Mathematik, vol. 66, pp. 1–31, 1993.
9. V. Caselles, R. Kimmel, and G. Sapiro, “Geodesic snakes,” in Proc. ICCV, June 1995.
10. V. Caselles and C. Sbert, “What is the best causal scale-space for 3-D images?,” Dept. Math. Comput. Sci., Univ. Illes Balears, Palma de Mallorca, Spain, Tech. Rep., Mar. 1994.
11. B. Chow, “Deforming along with convex hyper-surfaces by the accurate nth root of the Gaussian curvature,” J. Differential Geometry, vol. 22, pp. 117–138,1985.
12. L. D. Cohen, “On active contour models and balloons,” CVGIP: Image Understanding, vol. 53, pp. 211–218, 1991.
13. I. Cohen and L. D. Cohen, “Using deformable surfaces to segment 3-D images and infer differential structure,” CVGIP: Image Understanding,vol. 56, pp. 242–263, 1992.
14. M. P. Do Carmo, Differential Geometry of Curves and Surfaces. Englewood Cliffs, NJ: Prentice-Hall, 1976.
15. Riemannian Geometry. Englewood Cliffs, NJ: Prentice-Hall, 1992.

ISSN 2348 – 7968

16. C. L. Epstein and M. Gage, "The curve shortening flow," in *Wave Motion: Theory, Modeling, and Computation*, A. Chorin and A. Majda, Eds. New York: Springer-Verlag, 1987.
17. M. Gage and R. S. Hamilton, "The heat equation shrinking convex plane curves," *J. Differential Geometry*, vol. 23, pp. 69–96, 1986.
18. I. M. Gelfand and S. V. Fomin, *Calculus of Variations*. Englewood Cliffs, NJ: Prentice-Hall, 1963.
19. M. Grayson, "The heat equation shrinks embedded plane curves to round points," *J. Differential Geometry*, vol. 26, pp. 285–314, 1987.
20. A. Gupta, L. von Kurowski, A. Singh, D. Geiger, C. Liang, M. Chiu, P. Adler, M. Haacke, and D. Wilson, "Cardiac MRI analysis: Segmentation of myocardial boundaries using deformable models," Siemens Corp. Res., Princeton, NJ, Tech. Rep., 1995.
21. S. R. Gunn and M. S. Nixon, "A dual active contour model," *BMVC'94*, Sept. 1994, pp. 305–314.
22. G. Huisken, "Flow by mean curvature of convex surfaces into spheres," *J. Differential Geometry*, vol. 20, pp. 237–266, 1984.
23. M. Crandall, H. Ishii, and P. L. Lions, "User's guide to viscosity solutions of second-order partial differential equations," *Bull. Amer.Math. Soc.*, vol. 27, pp. 1–67, 1992.
24. M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Comput. Vision*, vol. 1, pp. 321–331. 1987.
25. S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A. Yezzi, "Gradient flows and geometric active contours," in *Proc. ICCV*, June 1995, Sept. 1994.
26. "Conformal curvature flows: From phase transitions to active vision," *Arch. Rational Mech. Anal.*, vol. 134, pp. 275–301, 1996.
27. B. B. Kimia, A. Tannenbaum, and S. W. Zucker, "Toward a computational theory of shape: An overview," in *Lecture Notes in Computer Science*, vol. 427. New York: Springer-Verlag, 1990, pp. 402–407.
28. B. B. Kimia, A. Tannenbaum, and S. W. Zucker, "Shapes, shocks, and deformations—I," *Int. J. Comput. Vision*, 1995.
29. B. B. Kimia, A. Tannenbaum, and S. W. Zucker, "On the evolution of curves via a function of curvature—I: The classical case," *J. Math. Anal., Applicat.*, vol. 163, pp. 438–458, 1992.
30. R. J. LeVeque, *Numerical Methods for Conservation Laws*. Boston: Birkh¨auser, 1992.
31. P. L. Lions, *Generalized Solutions of Hamilton–Jacobi Equations*. Boston: Pitman, 1982.
32. R. Malladi, J. Sethian, and B. Vemuri, "Shape modeling with front propagation: A level set approach," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, pp. 158–175, 1995.
33. F. Morgan, *Riemannian Geometry*. Boston: John and Bartlett, Boston, 1993.
34. T. McInerney and D. Terzopoulos, "Topologically adaptable snakes," in *Proc. ICCV'95*, June 1995, pp. 840–845.
35. P. Neskovic and B. Kimia, "Three-dimensional shape representation from curvature-dependent deformations," LEMS, Brown Univ, Tech. Rep. 128, 1994.
36. P. Olver, G. Sapiro, and A. Tannenbaum, "Geometric invariant evolution of surfaces and volumetric smoothing," *SIAM J. Math. Anal.*, submitted for publication.
37. S. Osher, "Riemann solvers, the entropy condition, and difference approximations," *SIAM J. Numer. Anal.*, vol. 21, pp. 217–235, 1984.
38. S. J. Osher and J. A. Sethian, "Fronts propagation with curvature dependent speed: Algorithms based on Hamilton–Jacobi formulations," *J. Computat. Phys.*, vol. 79, pp. 12–49, 1988.
39. S. Osher and L. I. Rudin, "Feature-oriented image enhancement using shock filters," *SIAM J. Numer. Anal.*, vol. 27, pp. 919–940, 1990.
40. P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 629–639, 1990.