University of Alberta

Skeletonization and Segmentation Algorithms for Object Representation and Analysis

by

Tao Wang

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

Spring 2010
Edmonton, Alberta

Examining Committee

Anup Basu, Computing Science

Irene Cheng, Computing Science

Walter Bischof, Computing Science

Pierre Boulanger, Computing Science

Carlos Flores-Mir, Dentistry

Ghassan Hamarneh, Computer Science, Simon Fraser University

To my wife, Xuefen Chen, and our families.

Abstract

Skeletonization and segmentation are two important techniques for object representation and analysis. Skeletonization algorithm extracts the “centre-lines” of an object and uses them to efficiently represent the object. It has many applications in various areas, such as computer-aided design, computer-aided engineering, and virtual reality. Segmentation algorithm locates the target object or Region Of Interest (ROI) from images. It has been widely applied to medical image analysis and many other areas. This thesis presents two studies in skeletonization and two studies in segmentation that advanced the state-of-the-art research. The first skeletonization study suggests an improvement of an existing algorithm for connectivity preservation, which is one of the fundamental requirements for skeletonization algorithms. The second skeletonization study proposes a method to generate curve skeletons with unit-width, which is required by many applications. The first segmentation study presents a new approach named Flexible Vector Flow (FVF) to address a few problems of other active contour models such as insufficient capture range and poor convergence for concavities. This approach was applied to brain tumor segmentation in two dimensional (2D) space. The second segmentation study extends the 2D FVF algorithm to three-dimension (3D) and utilizes it to automatically segment brain tumors in 3D.

ACKNOWLEDGMENT

I would like to thank my wife Xuefen Chen and my son Sky Wang for their great endurances during my PhD studies. I would thank my father Dexun Wang and mother Sanyu You for their nurturance and long time support. I also thank my father-in-law Songan Chen and mother-in-law Liandi Xu for their support.

I would like to thank my supervisor Dr. Anup Basu and co-supervisor Dr. Irene Cheng for their attentive guidance and kind help during my PhD studies. They always encouraged me to explore my potentials and try new approaches to advance the state-of-the-art research.

I would also like to thank my committee members (in no particular order), Drs. Ghassan Hamarneh, Walter Bischof, Pierre Boulanger and Carlos Flores Mir for their careful examination and valuable suggestions on my thesis.

I also thank all my teachers, colleagues and friends, especially Tianhao Qiu, Dr. Lihang Ying, Dr. Meghna Singh, Rui Shen, Feng Chen, Dr. Zhipeng Cai, Jiyang Chen, George Qiaohao Zhu, Dr. Gang Wu, Nicholas Boers, Dr. Soudong Zou, Dr. Baochun Bai, Dr. Cheng Lei, Dr. Jun Zhou, Dr. Sa Li, Yongjie Liu, Yang Zhao, Benjamin Chu, Monika Owczarek, Steve Jaswal, Alexey Badalov, Hossein Azari, Ivan Filippov, Matthew Wearmouth, Nathaniel Rossol, Parisa Naeimi, Saul Rodriguez, Tao Xu, Victor Jesus Lopez, Dr. Steven Miller, Dr. Shoo Lee, Dr. Paul Major, Dr. Manuel O Lagravere, Catherine Descheneau, Edith Drummond and Dr. Russell Greiner, for their help.

Last but not the least, I would like to thank AHFMR-HBI, iCORE, the Department of Computing Science, the University of Alberta, for financial support and providing me the excellent environment for PhD studies.

TABLE OF CONTENTS

List of Tables

List of Figures

Cornea proposed a potential field based algorithm to generate curve skeletons [2]. The idea is to extract some critical points in a force field to generate the skeleton. This algorithm has three steps. First is to compute the vector field on a 3D model. Second is to locate the critical points in the vector field, and

Chapter 1 Introduction

1.1 Motivations and Contributions

This thesis presents two studies in skeletonization and two studies in segmentation that advanced the state-of-the-art research. The motivations behind the thesis and the contributions of the thesis are introduced in this section. This thesis is based on two refereed journal papers [14, 19], four refereed conference papers [18, 21-22, 24], one refereed conference poster [23], and one submitted paper [20].

Three-dimensional (3D) models are extensively used in many areas, including medical image processing and visualization, computer-aided design, computer-aided engineering, and virtual reality. In some real-world applications, the input data is very dense and may require extensive computational resources. Efficient representation of a 3D model is therefore crucial for many applications to achieve satisfactory quality of service. An effective approach to address this issue is to consider skeletonization. The 3D skeleton extracted by a skeletonization algorithm is a compact representation of a 3D model. 3D skeletons can be used in many applications [1-2] such as 3D pattern matching, 3D recognition and 3D database retrieval. Connectivity preservation is one of the most desired properties for a skeletonization algorithm. It requires that the skeletons must be connected for connected object. Unfortunately, many skeletonization algorithms [10, 17] generate disconnected skeletons. Figure 1.1 displays some examples of disconnected skeletons.

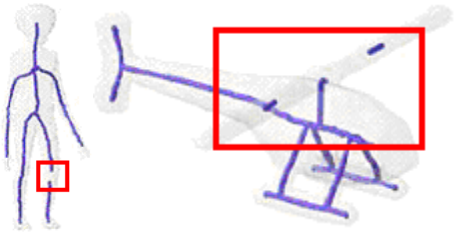


Figure 1.1: Models with disconnected skeletons (images adapted from [17])

In the first skeletonization study, a correction [14] of a 3D skeletonization algorithm [10] is presented. The algorithm [10] was one of the first, if not the

only, fully parallel 3D skeletonization algorithms in the literature and has been applied in many fields such as medical image processing [11] and 3D reconstruction [12]. However, this algorithm fails to preserve connectivity. Lohou discovered this problem and gave a counter example in [13]. Some other researchers, such as Chaturvedi [11] who applied this algorithm and found it disconnected small segments, but did not suggest how to fix this problem. Our study reveals the reason for the problem and gives a solution to it. This study solves a long-time pending problem for the skeletonization research community. We had applied the new algorithm to generate skeletons for automatic estimation of 3D transformation for object alignment [21-22].

In addition to connectivity preservation, many applications, *e.g.*, 3D object similarity match and retrieval [17] require unit-width curve skeletons (*i.e.*, the skeleton is only one-voxel thick and has no crowded regions). However, many 3D skeletonization algorithms [10, 14-16] fail to generate unit-width curve skeletons. The second skeletonization study [18] presents a so-called Valence Normalized Spatial Median (VNSM) algorithm, which eliminates crowded regions and ensures that the output skeleton is unit-width. Figure 1.2 (Left) shows an example of non-unit-width skeleton with crowded regions with crowded points and Figure 1.2 (Right) shows the unit-width curve skeleton generated by the proposed method. The proposed method can serve as a "post-processer" for other skeletonization algorithms to obtain unit-width curve skeleton, which can be used in a variety of applications mentioned above. Recently, we encoded unit-width curve skeletons to generate chain expressions for measuring 3D shape dissimilarity [23, 24].

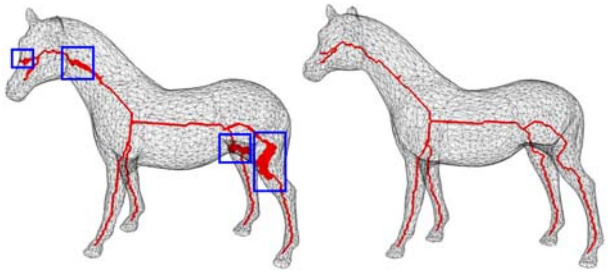
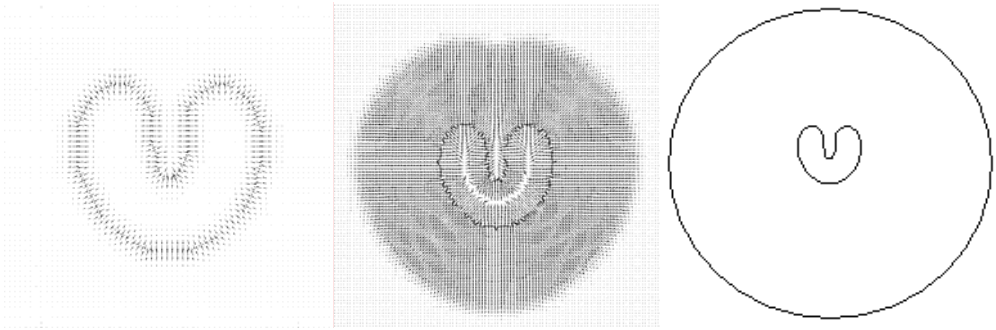


Figure 1.2: (Left) a horse model with non-unit-width skeleton. (Right) the horse model with unit-width curve skeleton.

This thesis addresses another important and interesting problem: segmentation. The goal of segmentation is to locate the target object, *i.e.*, the Region Of Interest (ROI) in 2D or the Volume Of Interest (VOI) in 3D. Active contour models or snakes [3-7] have been widely adopted as effective tools for segmentation [8-9]. Active contour based segmentation algorithms have many applications such as medical image processing and analysis. Our application is brain tumor segmentation in Magnetic Resonance (MR) images.

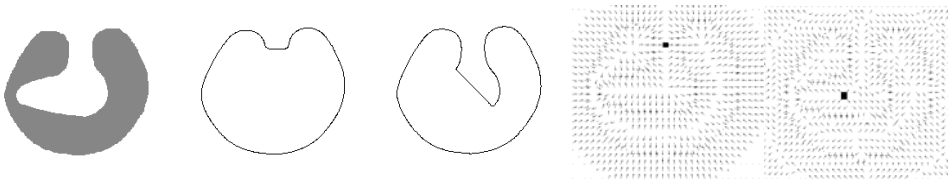
There are two limitations with the existing active contour models: limited capture range and inability to handle acute concave shapes. Capture range is the region that the external forces of the active contour are strong enough to drive contour evolution. The external forces of the traditional [3] and Gradient Vector Flow (GVF) [4] snakes are represented as small arrows in Figure 1.3 (a) and (b). The length of an arrow represents the magnitude of an external force at that location. In Figure 1.3, the capture range is the region with dense arrows (external forces) that are strong enough to drive the contour evolution. We can see that the capture range of the traditional snake is a very limited region around the object boundary. GVF diffuses the external forces from the object boundary to its surroundings to obtain a larger capture range. However, the capture range of GVF is still not the entire image. If the initialization is out of the capture range, the active contour will not evolve (Figure 1.3 (c)).



(a) (b) (c)

Figure 1.3: The limited capture range of (a) a traditional parametric snake and (b) a GVF parametric snake. If the initialization (outer circle in (c)) is outside the capture range, convergence does not occur.

Second, some active contour models, e.g. GVF and Boundary Vector Flow (BVF) [5], are unable to extract acute concave shapes (Figure 1.4 (b) and (c)). We observe that a number of active contour models (traditional, GVF and BVF) [3-5] are unable to extract acute concavities because their external force fields are static. There could be saddle points or stationary points [6] where the composition of external forces is zero (Figure 1.4 (d) and (e)) in static force fields. Therefore, the contours will get stuck at those locations and equilibrium will be achieved too early [6].



(a) (b) (c) (d) (e)

Figure 1.4: (a) An acute concave shape. (b) GVF and (c) BVF are not able to capture the acute concave shape. A saddle point in GVF is shown in (d) and a stationary point in BVF is shown in (e).

The first segmentation study [19] presents a semi-automatic approach called Flexible Vector Flow (FVF) active contour model to address problems of insufficient capture range and poor convergence for concavities. FVF was validated on a few datasets and applied to segment brain tumor in 2D.

One drawback of our 2D FVF algorithm [19] was that an initial contour must be given to start the vector flow evolution. In the second segmentation study [20], the FVF algorithm was extended to 3D and a Gaussian Bayesian Classifier was used to provide an initial position of a brain tumor to the 3D FVF algorithm to make the brain tumor segmentation process fully automatic. We compared our segmentation results with ground-truth segmentations (segmented by human experts) and found satisfactory accuracy in some test cases. However, in other test cases, poor accuracy has been observed. Therefore, this method must be further improved and validated before being applied to clinical trials. Figure 1.5 shows

one of the test brain MR images (1st column), the ground truth segmentations (2nd column) and the brain tumor extracted by the proposed method (3rd column).

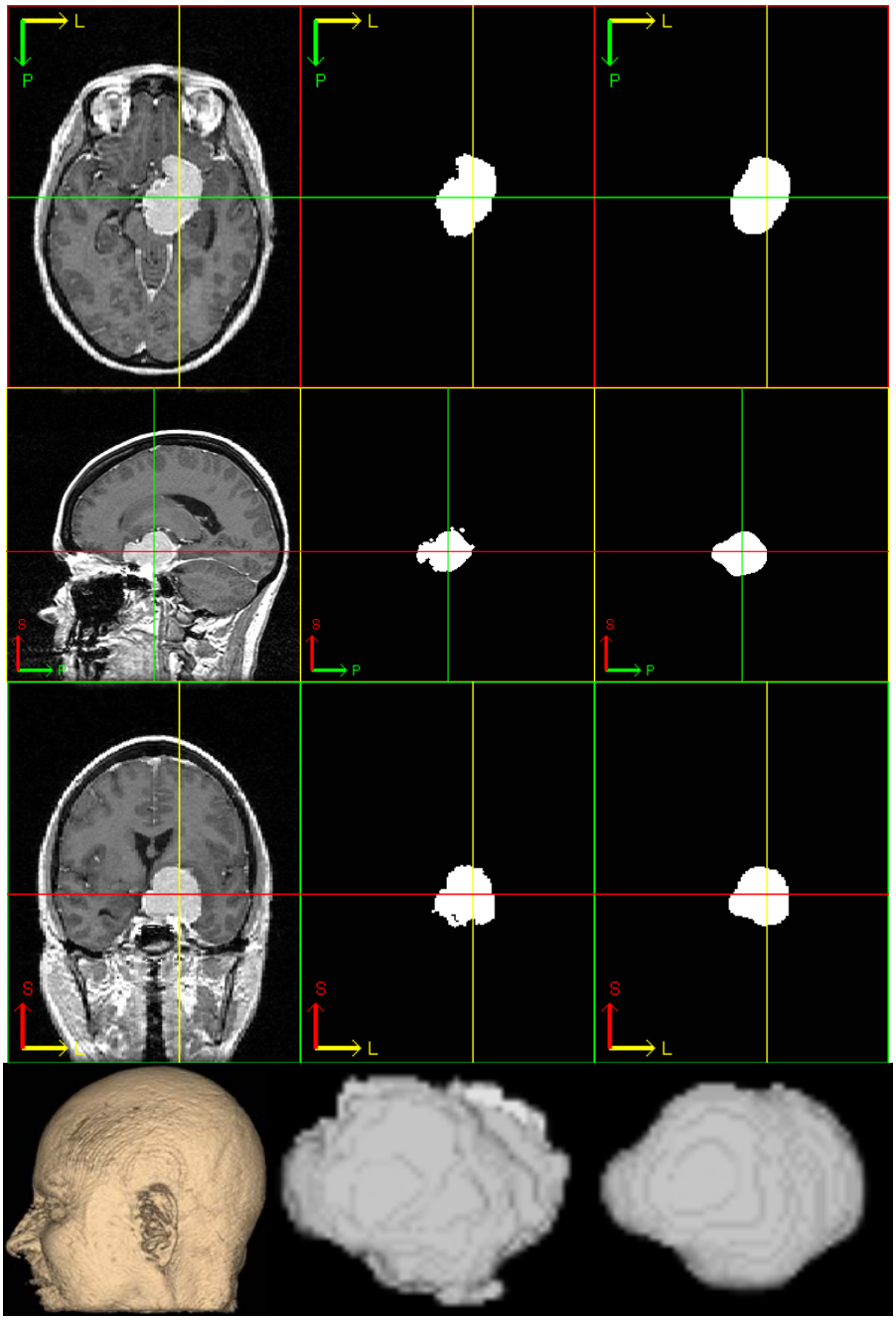


Figure 1.5: A demonstration of 3D brain tumor segmentation. 1st column: brain MR image, 2nd column: ground truth, 3rd column: brain tumor extracted by the proposed method. 1st row: axial view, 2nd row: sagittal view, 3rd row: coronal view, 4th row: volume rendering.

1.2 Thesis Organization

The remainder of this thesis is organized as follows. In the next chapter, the first skeletonization study on connectivity preservation is introduced. Then the second skeletonization study on unit-width curve skeleton is presented in Chapter 3. In Chapter 4, the first segmentation study based on Flexible Vector Flow is introduced, followed by the second segmentation study in Chapter 5. Discussions and conclusions are presented in Chapter 6.

Bibliography

[1] P. J. Besl and R. C. Jain. Three-dimensional object recognition. *ACM Computing Surveys*, 17 (1): pp 75-145, 1985.

[2] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin and D. Jacobs. A Search Engine for 3D Models, *ACM Trans. on Graphics*, 22(1): pp 83-105, 2003.

[3] M. Kass, A. Witkin, and D. Terzopoulus, "Snakes: Active contour model", Intl. J. of Computer Vision, vol. 1(4), pp. 321-331, 1988.

[4] C. Xu and J. L. Prince, "Snakes, shapes, and gradient vector flow", IEEE Trans. on Image Processing, pp. 359-369, 1998.

[5] K.W. Sum and P. Y. S. Cheung, "Boundary vector field for parametric active contours", Pattern Recognition, pp. 1635-1645, 2007.

[6] X. Xie and M. Mirmehdi, "MAC: Magnetostatic Active Contour Model", IEEE Trans. PAMI, vol. 30(4), pp. 632-645, 2008.

[7] O. Juan, R. Keriven, and G. Postelnicu, "Stochastic motion and the level set method in computer vision: Stochastic active contours", Intl. J. of Computer Vision, vol. 69(1), pp. 7–25, 2006.

[8] I. Dagher and K. E. Tom, Water, "Balloons: A hybrid watershed Balloon Snake segmentation", Image and Vision Computing, vol. 26 (7), pp. 905-912, 2008.

[9] S. W. Yoon, C. Lee, J. K. Kim and M. Lee, "Wavelet-based multi-resolution deformation for medical endoscopic image segmentation", J. of Medical Systems, vol. 32 (3), pp. 207-214, 2008.

[10] C. M. Ma, M. Sonka, A fully parallel 3D thinning algorithm and its applications, Computer Vision and Image Understanding, 64 (3), November 1996, pp. 420-433.

[11] A. Chaturvedi, Z. Lee, Three-dimensional segmentation and skeletonization to build an airway tree data structure for small animals, 50 (7), April 2005, Physics in Medicine and Biology, pp. 1405-1419.

[12] M.S. Talukdar, O. Torsaeter, M.A. Ioannidis, J.J. Howard, Stochastic reconstruction, 3D characterization and network modeling of chalk, Journal of Petroleum Science and Engineering, 35 (1-2), July 2002, pp. 1-21.

[13] C. Lohou, Contribution à l'analyse topologique des images (Ph.D. thesis), UNIVERSITÉ DE MARNE-LA-VALLÉE, 2001.

[14] **T. Wang** and A. Basu. A note on "A fully parallel 3D thinning algorithm and its applications", Pattern Recognition Letters, 28(4): 501-506, 2007.

[15] K. Palagyi and A. Kuba. A 3D 6-subiteration thinning algorithm for extracting medial lines, Pattern Recognition Letters, 19 (7): pp 613-627, 1998.

[16] C. Lohoua and G. Bertrand. A 3D 6-subiteration curve thinning algorithm based on P-simple points, Discrete Applied Mathematics, Vol. 151, pp 198–228, 2005.

[17] N. D. Cornea. Curve-Skeletons: Properties, Computation And Applications, Ph.D. Thesis, The State University of New Jersey, May 2007.

[18] **T. Wang** and I. Cheng, Generation of Unit-width curve skeletons based on Valence Driven Spatial Median (VDSM), International Symposium on Visual Computing, LNCS 5358, pages 1061-1070, 2008.

[19] **T. Wang**, I. Cheng and A. Basu, Fluid Vector Flow and Applications in Brain Tumor Segmentation, IEEE Transactions on Biomedical Engineering, Vol.56(3), pages 781-789, 2009.

[20] **T. Wang**, I. Cheng and A. Basu, Automatic Brain Tumor Segmentation with Normalized Gaussian Bayesian Classifier and Fluid Vector Flow, submitted for publication.

[21] **T. Wang** and A. Basu, Iterative Estimation of 3D Transformations for Object Alignment, International Symposium on Visual Computing, LNCS 4291, pages 212-221, 2006.

[22] **T. Wang** and A. Basu, Automatic Estimation of 3D Transformations using Skeletons for Object Alignment, IAPR/IEEE International Conference on Pattern Recognition, pages 51-54, 2006.

[23] V. Lopez, I. Cheng, E. Bribiesca, **T. Wang** and A. Basu, Twist-and-Stretch: A Shape Dissimilarity Measure based on 3D Chain Codes, ACM SIGGRAPH Asia Research Poster, 2008.

[24] **T. Wang**, I. Cheng, V. Lopez, E. Bribiesca and A. Basu, Valence Normalized Spatial Median for Skeletonization and Matching, Search in 3D and Video workshop (S3DV), in conjunction with IEEE International Conference on Computer Vision (ICCV) 2009.

Chapter 2 Connectivity Preservation in 3D Skeletonization

2.1 Introduction

We live in a three-dimensional (3D) world. In many application areas, such as medical image processing and visualization, computer-aided design, computer-aided engineering, and virtual reality, we need to build and use 3D models. However, in many real-world applications, the input data is very dense. For instance, the Visible Man Bone triangle mesh model with 4,715,110 faces at Georgia Tech is 190MB in Open Inventor format. The volumetric models may have even larger size. For example, the size of a 1,024×1,024×1,024 binary 3D image is 1GB. Moreover, there is no reason to believe that the sizes of 3D models will stop increasing in the near future. Therefore, without efficient representation of a 3D model, some applications are considered to be impossible or very difficult.

The 3D skeleton extracted by a skeletonization algorithm is a compact representation of a 3D model. 3D skeletons can be used in many applications [1-2] such as 3D pattern matching, 3D recognition and 3D database retrieval. Skeletonization is also known as skeletonizing or topological skeleton generation.

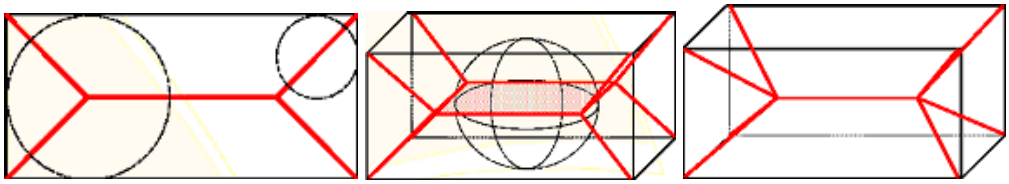
The skeleton can be defined via the Medial Axis Transformation (MAT) [3]. The MAT can be computed by "prairie fire" propagation. Consider an object as a prairie of uniform and dry grass. Suppose that a fire is lit along its border. All fire fronts advance into the object at the same speed. The MAT of the region is the set of points reached by more than one fire front at the same time.

Definition 2.1 The medial axis (or skeleton) of an *n* dimension manifold is the closure of the centers of all inscribed maximal hyper-spheres tangent to the manifold in at least two distinct locations.

In 2D space, a medial axis consists of the loci of the centers of all inscribed maximal circles of the 2D model - in other words, where these circles share at

least two points with the boundary of the model. Figure 2.1 (a) shows a rectangle and its skeleton.

In 3D space, a medial axis consists of the loci of the centers of all inscribed maximal spheres of the 3D model, where these spheres share at least two points with the boundary of the model. A skeleton in 3D consists of a set of 2D surfaces and/or 1D line segments. In many applications, a skeleton with only 1D line segments is more desirable. Figure 2.1 (b) shows a 3D box and its skeleton with 2D surfaces and 1D line segments. Figure 2.1 (c) shows the same 3D box and its skeleton with 1D line segments only.



(a) (b) (c)

Figure 2.1: (a) A rectangle and its skeleton in 2D (consisting of 5 line segments), shown with representative maximal circles and contact points; (b) 3D box and its skeleton with 2D surfaces and 1D line segments. (c) 3D box and its skeleton with 1D line segments only. (Images courtesy of Cornea [61].)

3D skeletons can be used in many applications such as 3D pattern matching, 3D recognition, 3D database retrieval, and dimensional reduction of complicated 3D models. There are four kinds of skeletonization techniques in the literature: thinning based algorithms [4-13, 15, 24-27, 29, 30], general field based algorithms [31, 36-47], Voronoi diagram based algorithm [48-52], and shock graph based algorithm [69-74, 76-78, 81].

The remainder of this chapter is organized as follows. In Section 2.2, the thinning based 3D skeletonization algorithms are introduced. Then the general field based 3D skeletonization algorithms are discussed in Section 2.3. In Section 2.4, the Voronoi diagram based 3D skeletonization algorithms are introduced, followed by the shock graph based 3D skeletonization algorithms in Section 2.5. In Section 2.6, the Ma and Sonka's algorithm [9] is briefly introduced, its problematic part is analyzed and the modification is presented, before the work is

concluded in Section 2.7. The modified algorithm was published in [109]. We had used the skeletons generated by the modified algorithm for automatic estimation of 3D transformation for object alignment [110-111].

2.2 Thinning based 3D skeletonization algorithms

A *3D thinning* algorithm applies in a local neighborhood of an object point and iteratively removes object points that satisfy some pre-defined masks to generate skeletons in a *3D binary image*. A 3D binary image is a mapping that assigns the value of 0 or 1 to each point in the discrete 3D space. Points having the value of 1 are called *black* (*object*) points, while 0's are called *white* (*background*) ones. Black points form objects of the binary image. The thinning operation iteratively deletes or removes some object points (that is, changes some black points to white) until some restrictions prevent further operation. Note that the white points will never be changed to black ones. Most of the existing thinning algorithms are parallel, since the medial axis transform (MAT) can be defined as fire front propagation, which is by nature parallel [3]. There are three categories of parallel thinning algorithms in the literature: fully parallel thinning algorithms [9, 24, 25], sub-iteration parallel thinning algorithms [4-6, 10-12, 26, 27, 29, 30], and sub-field parallel thinning algorithms [7, 8, 13, 15].

2.2.1 Basic concepts

Some terms and notations are introduced here:

Let p and q be two different points with coordinates (p_x, p_y, p_z) and (q_x , q_y, q_z), respectively, in a 3D binary image P. The Euclidean distance between p and q is defined as:

$$dis = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2 + (p_z - q_z)^2}$$

Then p and q are:

6-adjacent if $dis \le 1$. *18-adjacent* if $dis \le \sqrt{2}$. *26-adjacent* if $dis \le \sqrt{3}$. Let us denote the set of points *k-adjacent* to point p by $N_k(p)$, where k = 6, 18, 26, (see Figure 2.2). $N_k(p)$ is also called p's *k-neighborhood*. Let p be a point in a 3D binary image. Then, $e(p)$, $w(p)$, $n(p)$, $s(p)$, *u(p), and d(p)* are *6-neighbors* of *p,*

which represent 6 directions of *east*, *west*, *north*, *south*, *up*, and *down*, respectively. The *18-neighbors* of p (but not in p's *6-neighborhood*) are *nu*(p), *nd*(p), *ne*(p), *nw*(p), *su*(p), *sd*(p), *se*(p), *sw*(p), *wu*(p), *wd*(p), *eu*(p), and *ed*(p), which represent 12 directions of *north-up, north-down, north-east, north-west, south-up, south-down, south-east, south-west, west-up, west-down, east-up,* and *east-down*, respectively.

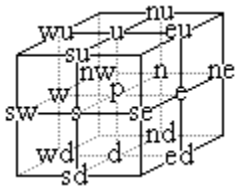


Figure 2.2: The adjacencies in a 3D binary image. Points in $N_6(p)$ are marked *u, n, e, s, w,* and *d*. Points in $N_{18}(p)$ but not in $N_6(p)$ are marked *nu, nd, ne, nw, su, sd, se, sw, wu, wd, eu,* and *ed.* The unmarked points are in $N_{26}(p)$ but not in $N_{18}(p)$.

Let S be a set of points of a 3D binary image P. A *k-path* in S is a sequence of distinct points with coordinates (x_0, y_0, z_0), (x_1, y_1, z_1), …, (x_n, y_n, z_n), where every two subsequent points (x_i, y_i, z_i) and $(x_{i+1}, y_{i+1}, z_{i+1})$ are *k-adjacent*. Two object points are said to be *26-connected* if there exists a 26-path between them consisting entirely of object points in S. An *object component* in S is a maximal 26-connected subset of object points of S. A *background component* in S is a maximal 6-connected subset of background points of S.

A *unit edge* is a pair of two adjacent points. A *unit square* is a set of four points of a 1×1 square. A *unit cube* is a set of eight points of a 1×1×1 cube.

A *boundary point* is an object point that has at least one background point in its 6-neighborhood. Thinning algorithms are designed to delete all the boundary points that satisfy certain pre-defined conditions.

Some thinning algorithms generate *surface skeletons* (shown in Figure 2.1 (b)) and some algorithms generate *curve skeleton* (shown in Figure 2.1 (c)). For surface skeleton thinning algorithms, *surface-end points* must be kept. A

boundary point *p* is a surface-end point if it is 6-adjacent to at least one opposite (U and D, or N and S, or E and W) pair of background points. For curve skeleton thinning algorithms, *curve-end points* must be kept. A boundary point *p* is a curve-end point if it is 6-adjacent to exactly at least one object point.

A 3D thinning algorithm should preserve the topology of a 3D image. To preserve topology, the number of *connected components*, the number of *cavities* and the number of *holes* should be preserved. However, it is very difficult to calculate these three numbers. Malandain and Bertrand suggested [62] and then proved [63] that to preserve topology only *simple* points can be deleted. This result is stated as the Theorem 2.1.

Definition 2.2 An object point is *simple* if it is 26-adjacent to only one object component in its 26-neighborhood and 6-adjacent to only one background component in its 18-neighborhood.

Theorem 2.1 Deletion of a simple point preserves topology.

Theorem 2.1 is sufficient to test the deletion of a single point preserve topology or not. Ma [14] proposed the theorem to test if a 3D thinning algorithm can preserve the connectivity or not based on Theorem 2.1. It is stated as Theorem 2.2.

Definition 2.3 Denote *X* as a set of object points in a 3D image *P*. If *X* can be ordered as a sequence in which every point is simple after all previous points in the sequence are deleted, then the set *X* is *simple*.

Theorem 2.2 A 3D thinning algorithm preserves connectivity of a 3D image *P* if the following conditions are satisfied:

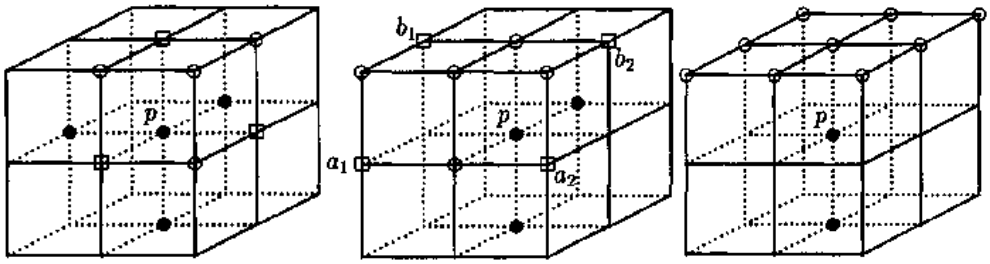
- Any subset of *P* that is contained in a unit square and deleted by the algorithm is simple.
- No object component contained in a unit cube can be deleted completely.

2.2.2 Fully parallel thinning algorithms

A fully parallel thinning algorithm applies simultaneously to all object points on the boundary. It processes all input points in a loop with the same set of deleting masks. Ma *et al.* proposed two fully parallel thinning algorithms [9, 24] for

extracting medial surfaces and medial lines. Manzanera *et al*. [25] presented an algorithm for extracting medial surfaces.

Ma [24] defined three deleting templates (shown in Figure 2.3). In Figure 2.3 and Figure 2.5 (see below), a "•" is used to denote an object point while a "◦" is used to denote a background point. An unmarked point is a "don't care" point, which can represent either an object point or a background point. In Figure 2.3(a), at least one point marked as □ is an object point. In Figure 2.3(b) at least one point in $\{a_1, b_1\}$ and at least one point in $\{a_2, b_2\}$ are background points. The algorithm simultaneously deletes every object point that satisfies any of the three deleting templates or any rotations/reflections of the deleting templates. Some results are shown in Figure 2.4. This algorithm is able to extract medial surfaces from 3D images. Theorem 2.2 was used to prove that this algorithm preserve connectivity.



(a) (b) (c)

Figure 2.3: Three deleting templates in [24].

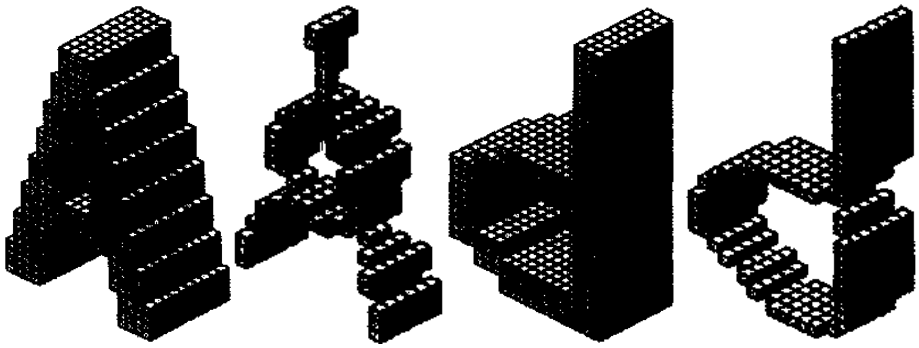


Figure 2.4: Some thinning results in [24].

Ma *et al*. [9] modified the previous algorithm [24] to generate medial lines. In [9], the algorithm is based on some pre-defined templates (Class A, B, C and D).

If the neighborhood of an object point matches one of the templates, it will be removed. Figure 2.5 shows the four basic template cores.

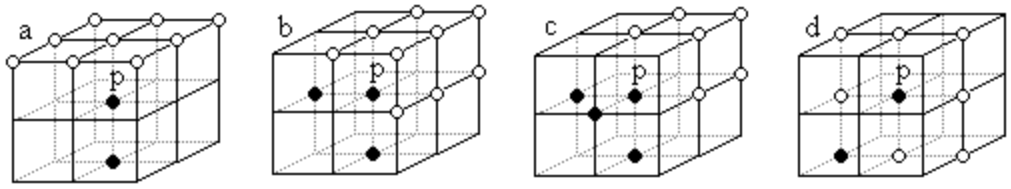


Figure 2.5: Four template cores (Class A, B, C and D) of the fully parallel thinning algorithm. For (d), there is an additional restriction that *p* must be a simple point.

The template cores themselves are not the deleting templates. Some translations [9] must be applied to the template cores to obtain the deleting templates. There are 6 templates in Class A, 12 templates in Class B and 8 templates in Class C and 12 templates in Class D. Templates in Class A-D are shown in Figures 2.6-2.9, In Figure 2.9, at least one point marked □ is an object point.

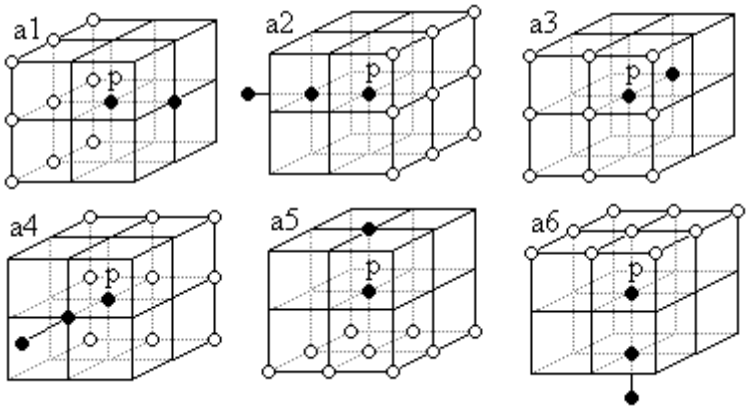


Figure 2.6: 6 deleting templates in Class A.

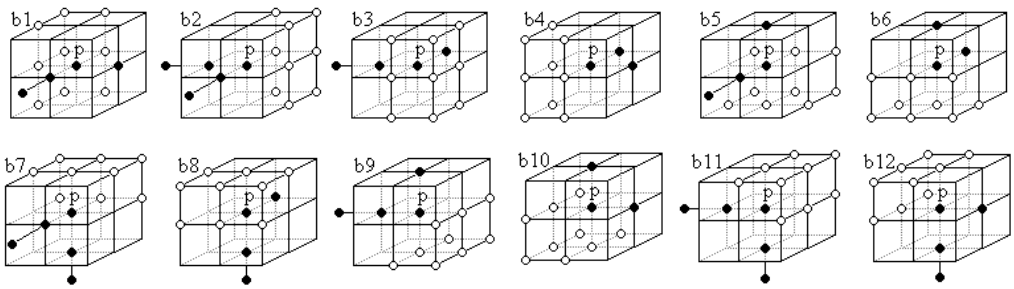


Figure 2.7: 12 deleting templates in Class B.

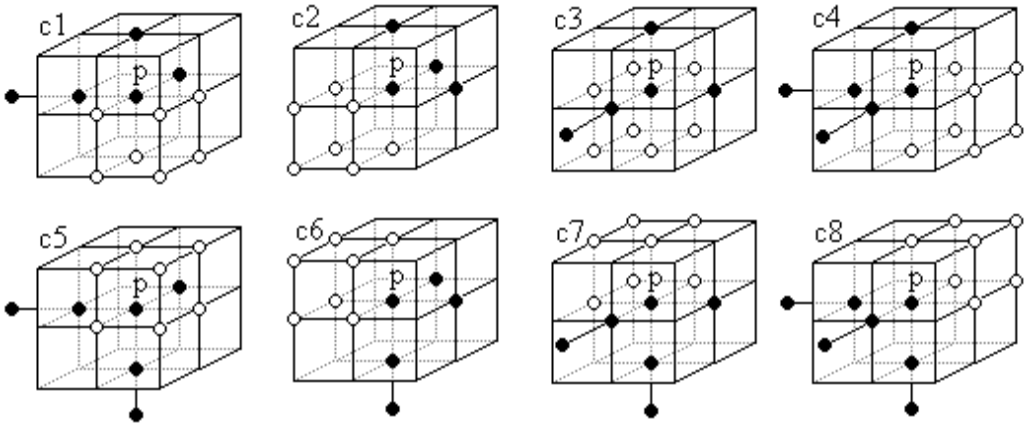


Figure 2.8: 8 deleting templates in Class C.

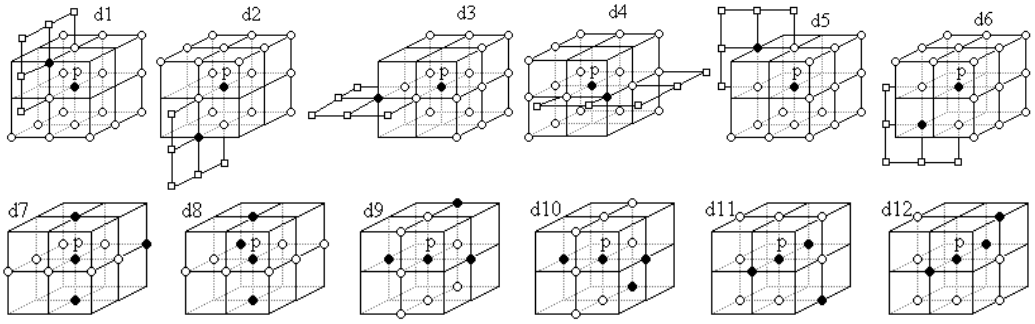


Figure 2.9: 12 deleting templates in Class D, where at least one point marked □ is an object point.

Ma and Sonka defined the preserving conditions as follows:

"Rule 2.2. Let *p* be an object point of a 3D image. Then:

1. *p* is called a line-end point if *p* is 26-adjacent to exactly one object point;

2. *p* is called a near-line-end point if p is 26-adjacent to exactly two object points which are:

(a) either *s(p)* and *e(p)*, or *s(p)* and *u(p)* but not both;

(b) either *n(p)* and *w(p)*, or *u(p)* and *w(p)* but not both; or

(c) either *n(p)* and *d(p)*, or *e(p)* and *d(p)* but not both;

3. p is called a *tail point* if it is either a line-end point near-line-end point; otherwise it is called a *nontail point*." Where *e(p), w(p), n(p), s(p), u(p)* and *d(p)* are the east, west, north, south, up, and down neighbors of *p*, respectively.

In each iteration, all non tail-points satisfying at least one of the deleting templates in Class A, B, C or D are deleted in the fully parallel thinning algorithm as follows:

Algorithm

Repeat

1) Mark every object point which is 26-adjacent to a background point;

2) **Repeat**

Simultaneously delete every non tail-point which satisfies at least one deleting template in Class A, B, C, or D;

Until no point can be deleted;

3) Release all marked but not deleted points;

Until no marked point can be deleted;

Figure 2.10 shows some results of this algorithm. Theorem 2.2 was used to prove this algorithm can preserve connectivity.

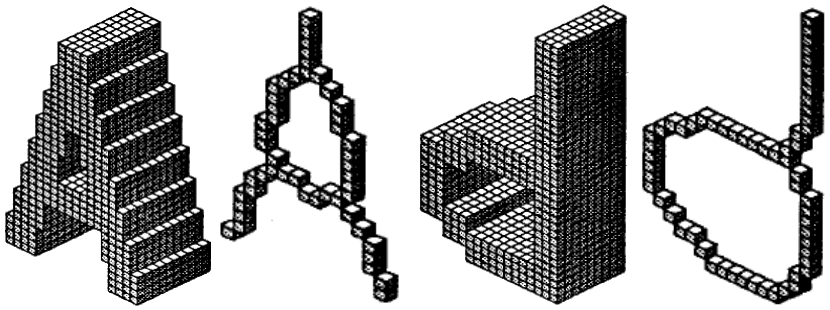


Figure 2.10: Two objects and their skeletons extracted by Ma and Sonka's algorithm [9].

Manzanera *et al*. [25] used morphological operators to extract medial surfaces in a fully parallel way. The algorithm is defined by five patterns (α_1, α_3, α_3, β_1, β_2). (α_1, α_3, α_3) are deleting patterns. (β_1, β_2) are non-deleting patterns which are used to preserve connectivity. Theorem 2.2 was used to prove this algorithm can preserve connectivity.

The advantage of a fully parallel thinning algorithm is that it requires the least number of iterations to extract a skeleton from a 3D image, as all the boundary points that satisfy the deleting conditions are deleted in a single iteration. In general, fewer iterations imply a faster thinning speed because a boundary point is tested only once. For the sub-iteration algorithms in the next section, a boundary point can be tested multiple times.

2.2.3 Sub-iteration parallel thinning algorithms

A sub-iteration parallel thinning algorithm examines a neighborhood (3×3×3, for most of the algorithms of this kind) for each border point. There are *n* sub-iterations in each iteration where only border points of a certain type can be deleted in each sub-iteration. Each sub-iteration uses a different deletion rule and is executed in a parallel fashion. All border points satisfying the deletion condition of the sub-iteration are deleted simultaneously.

There are four kinds of sub-iteration parallel thinning algorithms: 3-subiteration [26], 6-subiteration [4-6, 10-12], 8-subiteration [27], and 12-subiteration [29, 30], in the literature. The pseudo code for *n* sub-iteration parallel thinning algorithm can be sketched as follows:

Repeat
For i=1 to *n* do
Delete the border points that satisfy the condition assigned to the i-th direction
End for
Until no points can be deleted

As mentioned in Section 2.2.1, there are 6 major directions in 3D images; thus 6-subiteration parallel thinning algorithms were generally proposed.

Gong and Bertrand [6] developed a 6-subiteration thinning algorithm in 1990. This algorithm has two operations: a topological thinning operation T1 and a geometrical operation T2. The fist operation is designed to thin a 3D image while preserving topology and the second one is designed to thin a 3D image while preserving geometry. In each iteration, T1 and T2 are applied to one type of boundary points in a fixed sequence of directions: U, N, E, B, S and W. This

algorithm was published before [62]. They claimed that this algorithm preserves topology. However, they only proved that the algorithm does not disconnect or connect 3D object components, and it does not create holes or cavities. They did not prove that this algorithm does not remove holes or cavities.

Mukherjee *et al*. presented a 3D 6-subiteration thinning algorithm named ESPTA [10] (Extended Safe Point Thinning Algorithm). However, ESPTA disconnects a connected object. They improved it and proposed a new algorithm named MESPTA [11] (Modified Extended Safe Point Thinning Algorithm) in 1990. They used *safe* point instead of simple point to test if topology is preserved of not. The authors proved that the new algorithm preserves topology.

Palagyi *et al*. presented a 3D 6-subiteration thinning algorithm [4] in 1998. This algorithm is able to preserve geometry and topology of 3D models. However, it is sensitive to image rotation and noise. The deletion condition of this algorithm is described by some masks. An object point is to be deleted if – and only if – its neighborhood matches at least one of the given masks. The masks are constructed according to the six directions U, D, N, E, S, and W. Direction U masks consist of six base masks (M1-M6, shown in Figure 2.11) and rotations around the vertical axis (rotation angles are 90°, 180°, and 270°). Other masks for directions D, N, E, S, and W can be derived from the rotations and reflections of the masks for direction U.

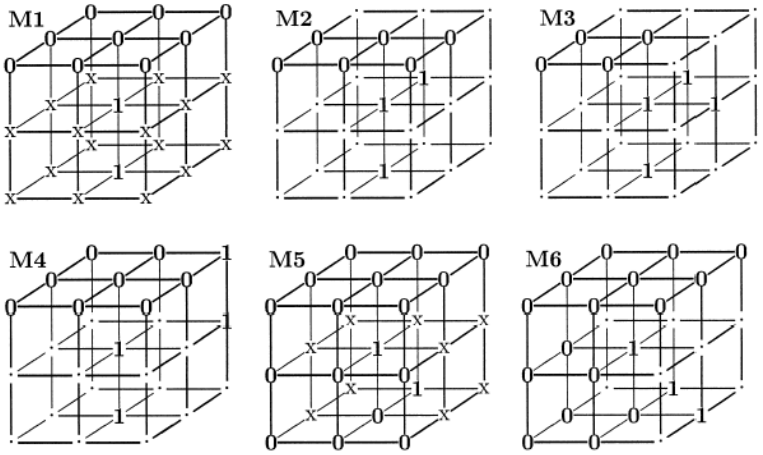


Figure 2.11: Base masks (M1-M6) in direction U [4]. 1: object point, 0: background point, •: object or background point, at least one point marked ‘‘x’’ is an object point.

In this algorithm, P = (26, 6, B) is a 3D image. P = (26, 6, B) is a notation which means that the background points are under 26-connectivity and the object points are under 6-connectivity. Point set B is stored in an array X. Thinning X in direction D results in an image T(X, D). The pseudo code of this algorithm is:

Input: binary array X representing the image P

Output: binary array Y representing the thinned image

```
Y=X;
Repeat
    Y=T(Y, U); Y=T(Y, D); Y=T(Y, N);
    Y=T(Y, S); Y=T(Y, E); Y=T(Y, W);
Until no points can be deleted
```

The deletion order is fixed in this algorithm. Another order of deletion leads to a different result. That explains why this algorithm is very sensitive to rotation and noise. Some thinning results of this algorithm are displayed in Figure 2.12.

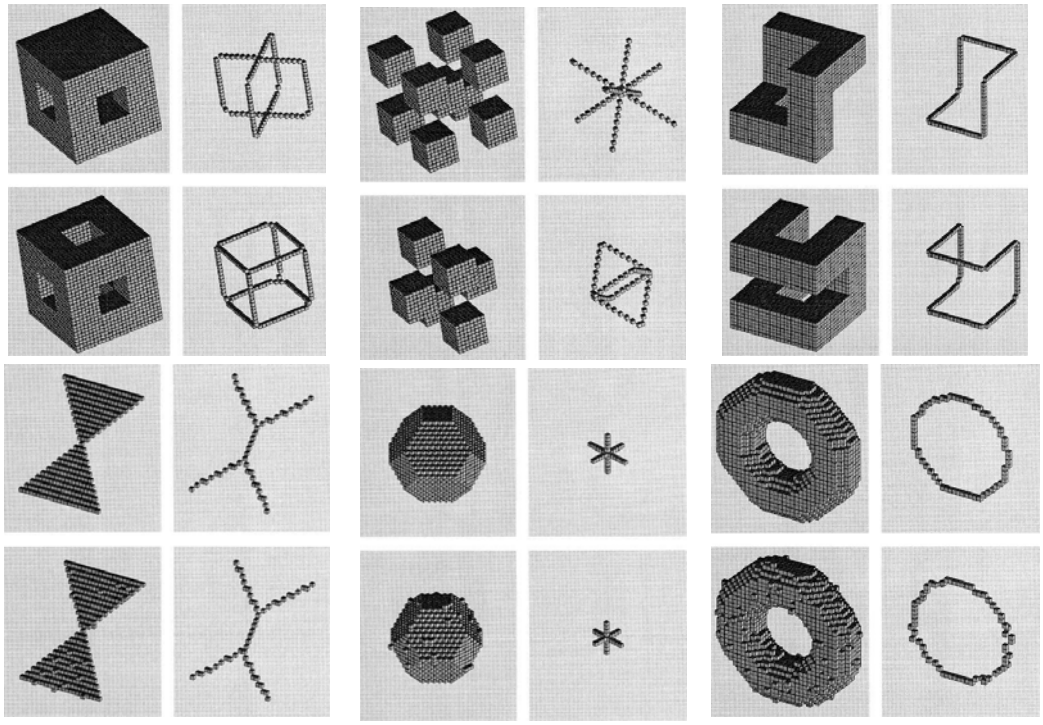


Figure 2.12: Some thinning results of Palagyi's 3D 6-subiteration thinning algorithm [4].

A recent 6-subiteration thinning algorithm [5] was proposed by Lohoua and Bertrand in 2005. They proposed a new methodology to build thinning algorithms based on the deletion of P-simple points instead of simple points. This methodology enables people to derive a thinning algorithm A from an existent thinning algorithm B, such that A deletes at least all the points removed by B while preserving the same end points. They applied this methodology and proposed a new 6-subiteration curve thinning algorithm which deletes at least all the points removed by two 6-subiteration curve thinning algorithms [4, 6]. They proved that any algorithm removing only subsets composed solely of P-simple points is guaranteed to preserve topology. Thereby, no proof is required in contrast to most of the existing thinning algorithms [24, 25, 9] that use simple points. Similar to the test of a simple point, a P-simple point can be tested by the examination of only its 26-neighborhood.

Ma and Wan [12] proposed a 6-subiteration thinning algorithm to extract medial surfaces and medial curves. While most other algorithms consider the object component as 26-connected, the object component is considered as 18-connected.

Palagyi [26] presented a 3-subiteration thinning algorithm for extracting medial surfaces. The three deletion directions are UD, NS and EW, corresponding to the three kinds of opposite pairs of points. The first sub-iteration deletes the U or D edge points that satisfy any deleting templates in direction UD. The second sub-iteration deletes the N or S edge points that satisfy any deleting templates in direction NS. The third sub-iteration deletes the E or W edge points that satisfy any deleting templates in direction EW. This work is non-trivial because it demonstrates a possible way for constructing non-conventional sub-iteration parallel thinning algorithms. Furthermore, less sub-iterations means less number of iteration are required to extract a skeleton from a 3D image.

An 8-subiteration thinning algorithm was proposed by Palagyi and Kuba [27] for extracting medial surfaces and medial curves. The eight deletion directions are USW, UWN, UNE, UES, DSW, DWN, DNE, and DES. For each deletion direction, some template cores are defined. Surface skeleton thinning algorithm

and curve skeleton thinning algorithm have different template cores. Template cores themselves are not the deleting templates. Deleting templates are defined as the reflections of the template cores. The three symmetry planes for reflecting templates are shown in Figure 2.13. For the surface skeleton thinning algorithm, surface-end points are kept. For curve skeleton thinning algorithms, curve-end points must be kept. Theorem 2.2 was used to prove this algorithm can preserve connectivity. Results show that the curve thinning algorithm is robust under noise. However, the surface thinning algorithm is sensitive to boundary noise because a noisy boundary may contain a number of surface-end points which must be kept.

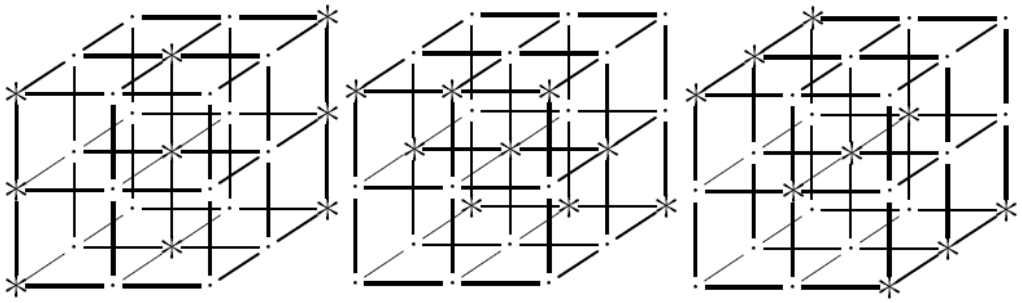


Figure 2.13: The three symmetry planes for reflecting templates in Palagyi and Kuba [27]. Points belonging to the reflecting planes are marked *.

A 12-subiteration thinning algorithm was proposed by Palagyi and Kuba [29] for extracting curve skeletons or surface skeletons. The twelve deletion directions are UN, UE, US,UW, NE,NW, ND, ES, ED, SW, SD, and WD. They are non-opposite and unordered. The order of the deletion directions is <US, NE, WD; ES, UW, ND; SW, UN, ED; NW, UE, SD>. Another order of the deletion directions leads to another algorithm. A number of deleting templates are defined for each direction. A drawback of this algorithm is that an object point may be checked for too many times before it is deleted.

Lohoua and Bertrand [30] presented a 12-subiteration thinning algorithm based on P-simple points. It takes advantage of the P-simple points and can delete at least all the points removed by [29].

2.2.4 Sub-field parallel thinning algorithms

Sub-field parallel thinning algorithms divide input points into different sub-fields and apply different deleting masks to points in different sub-field. For a parallel thinning algorithm, simultaneous deletion of two or more points (even if they are simple) may disconnect a 3D image. For instance, in Figure 2.14, object points *p* and *q* are simple points. Deletion of *p* or *q* will not disconnect the 3D image. However, simultaneous deletion of *p* and *q* breaks the 3D image into 2 pieces. The simple test is done in the 26-neighborhood of an object point. If the object points of a 3D image are partitioned into several subfields so that two object points are non 26-adjacent in the same subfield, deletion of one point will not bring the problem of connectivity violation. This is the motivation of sub-field parallel thinning algorithm.

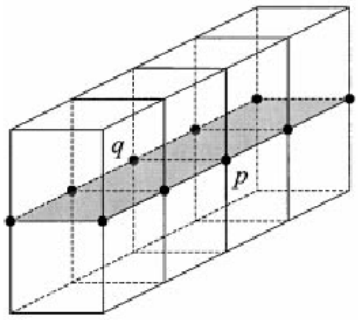


Figure 2.14: Simultaneous deletion of two simple points, *p* and *q*, disconnect the 3D image. Image courtesy of Ma [12].

A sub-field parallel thinning algorithm divides the 3D model into some disjointed subsets. At a given iteration, only the border points of the active subfield can be deleted. The pseudo code of subfield parallel thinning algorithm consisting of *n* subfields is as follows:

Repeat

For i=1 to *n* do

Delete the border points in the i-th subfield that satisfy the global condition (assigned to the i-th subfield)

End For

Until no points can be deleted

Saha *et al*. [7] proposed an 8 sub-fields parallel thinning algorithm extracting surface skeletons and curve skeletons. A 3D image is divided into eight disjointed sub-fields so that no two object points in the same sub-field are 26-adjacent. Hence, the members of each sub-field can be used for parallel thinning without the risk of connectivity violation described in Figure 2.14. They showed that the shape distortion increases linearly with the percentage of noise. The authors also found that the maximum shape distortion appeared when the rotation is 45°, which is expected due to the discrete nature of a 3D image.

Ma and his colleagues introduced three sub-field parallel thinning algorithms [8, 13, 15] during 2001 and 2002. In [8], a 2-subfield thinning algorithm was proposed to extract medial surfaces. The object points in a 3D image are divided into two subfields where two 18-adjacent object points are in the same subfield, and two 6-adjacent object points are in different subfields. The authors argued that the division of two subfields can prevent the risk of connectivity violation when deleting two 6-adjacent object points because 6-adjacent object points are in different subfields. However, as discussed in the beginning of Section 2.2.4, only when two object points are non 26-adjacent in the same subfield, deletion of one point will not bring the problem of connectivity violation. Theorem 2.2 was used to prove this algorithm can preserve connectivity. A 2-subfield thinning algorithm [13] was proposed to extract medial lines with similar technology. A 4-subfield thinning algorithm [15] was developed to extract medial lines and medial surfaces. The object points in a 3D image are divided into four subfields where two 26-adjacent object points are in the same subfield, and two 18-adjacent (and 6-adjacent) object points are in different subfields. A subfield based thinning algorithm starts thinning from the first subfield and then the other subfield(s) sequentially. Therefore, a 2-subfield thinning algorithm has only 2 sequential thinning steps in each iteration and thus a removable object point can only be tested up to 2 times before it is deleted. However, it has to avoid the problem of connectivity violation. An 8 subfield thinning algorithm has 8 sequential thinning steps in each iteration. Thereby, a removable object point may be tested up to 8 times before it is deleted. The advantage is that an 8 subfield thinning algorithm

has no risk of connectivity violation. A 4 subfield thinning algorithm is a tradeoff in-between.

2.3 General field based 3D skeletonization algorithms

General field based methods have two steps:

1. A certain field is created for the 3D object.
2. The local extrema are detected as skeleton points.

Different general field based algorithms take advantage of different fields; the skeleton depends on the chosen field. Over the past two decades, a number of different fields – such as distance field [36-44], potential field [45-47] and electrostatic field [31] – have been used by 3D skeletonization algorithms. Unfortunately, no matter what kind of field is used, an algorithm has to take some extra steps to connect skeletal points due to the threshold issue for extrema detection.

Among all kinds of fields, distance field based algorithms have been studied most extensively. First, a distance map is generated where each element gives the distance from an object point to the nearest boundary point. Then, the local maxima are detected as skeleton points. This is a direct implementation of "prairie fire" propagation. One of the advantages of distance transform is that it is very efficient. The computation complexity of distance transform can be linear ($O(n)$) time in arbitrary dimensions, where n is the number of the points in the image. The most serious disadvantage of distance transform is that it may not preserve topology; it needs some thresholds to detect the maxima and thereby the skeleton may be disconnected.

The skeleton depends on the chosen distance metric. City block distance metric was used in [37] and chessboard distance metric was used in [38]. Among all the different distance metrics, Euclidean distance is the "true" distance between points.

The Euclidean distance between two points a ($a_1, a_2, \ldots, a_n$) and b ($b_1, b_2, \ldots, b_n$) in n-space is defined as:

$$\text{Euclidean distance} = \sqrt{\sum_{i=1}^{n} | a_i - b_i |^2}$$

In 1D space, Euclidean distance = $| a_1 - b_1 | = | a_x - b_x |$

In 2D space, Euclidean distance = $\sqrt{| a_x - b_x |^2 + | a_y - b_y |^2}$

In 3D space, Euclidean distance = $\sqrt{| a_x - b_x |^2 + | a_y - b_y |^2 + | a_z - b_z |^2}$

The computation of the Euclidean distance is expensive because square root has to be calculated. Some approaches have been proposed to estimate the Euclidean distance. A fast estimation [65] of 2D Euclidean distance is:

$$\text{Euclidean distance} \approx \begin{cases} 0.41 | a_x - b_x | + 0.941246 | a_y - b_y |, \text{when } | a_y - b_y | < | a_x - b_x | \\ 0.41 | a_y - b_y | + 0.941246 | a_x - b_x |, \text{otherwise} \end{cases}$$

The difference between the estimated distance and the exact distance is between -6% and +3%. A distance transform skeletonization algorithm compares distances to detect the local maxima as part of the skeleton. We notice $|A| > |B| \Leftrightarrow |A|^2 > |B|^2$. Therefore, it is not necessary to calculate the square root at all. The square of Euclidean distance is used for comparison instead of the exact Euclidean distance. In 3D space, the chamfer distance [66] is widely used to as the distance metric instead of Euclidean distance. The chamfer distance [64] from an object point *c* to the nearest boundary point is defined as:

$D_{chamfer}(c) = \min \{D_{chamfer}(d) + D_{est}(c, d) \}$, where *d* is an object point in *c*'s 26-neighborhood and $D_{est}(c, d)$ is an integer estimation to the Euclidean distance between *c* and *d*. $D_{est}(c, d)$ is defined as:

$$D_{est}(c, d) = \begin{cases} 3, \text{if c and d are neighbors sharing an area} \\ 4, \text{if c and d are neighbors sharing an edge} \\ 5, \text{if c and d are neighbors sharing a point} \end{cases}$$

Borgefors [66] showed that this estimation minimized the upper bound on the difference between the chamfer and Euclidean distances. Pudney [36] presented a Distance Ordered Homotopic Thinning (DOHT) algorithm using the chamfer distance instead of Euclidean distance to generate skeletons.

Zhou [39-41] proposed a distance transform based 3D skeletonization algorithm and applied it to generate medial surfaces of cerebral sulci. Figure 2.15 displays the cerebral sulci and the medial surfaces [39]. The medical CT or MRI data in [39-41] are very large, so that precise calculation of Euclidean distances for skeletons is expensive. Therefore, a simpler distance transform based on F-neighbors [39] is used. The computation cost of this distance transform is low. However, the accuracy of this method is less than Borgefors's <3, 4, 5> chamfer distance transform [66]. Then, a global method is used to detect saddle points. The skeleton is extracted as a set of clusters with a set of local maximum paths (LMpaths), which associated with saddle points.

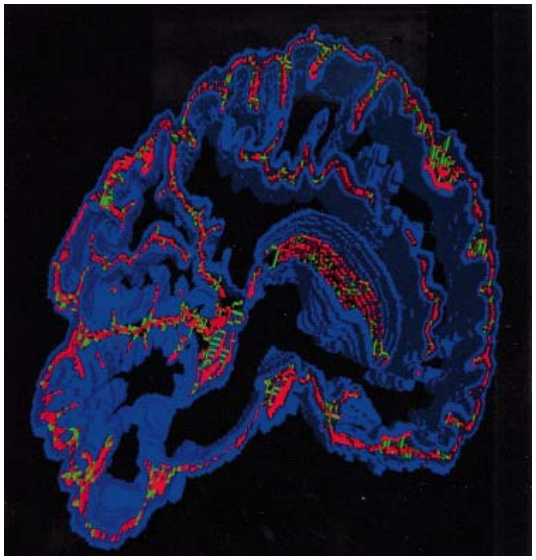


Figure 2.15: Cerebral sulci and the medial surfaces (turquoise) in [39].

Wan [42] presented a skeletonization algorithm for virtual navigation of flight path planning. The accurate Euclidean distance transform was used to generate a Distance From Boundary (DFB) field. The navigation path can be computed along the skeletons (central paths).

Sundar [43] used Euclidean distance transform to generate skeletons, which were utilized for 3D shape matching and retrieval. The approach has four steps. In the first step, Euclidean distances were calculated. A Thinness Parameter (TP) based algorithm was used to classify object points based on their importance for boundary coverage. If the Euclidean distance of an object point to the nearest boundary point is much greater than those of the object point's 26-neighbors, the

sphere centered at that object point is likely to include all of the spheres centered at its neighboring points. The TP determines how large such a sphere should be. In the second step, the skeletal points were connected by a clustering algorithm. In the third step, an undirected acyclic shape graph was generated with the Minimum Spanning Tree (MST) algorithm. A directed skeletal graph was then created by directing edges from points with higher Euclidean distance to the one with lower distance. In the last step, a shape matching approach based on directed acyclic graph was used for matching and retrieval.

A recent work [44] utilized the discrete bisector function to analyze and filter Euclidean distance based skeletons. The discrete bisector function was built with the Voronoi diagram.

Potential field is used in some algorithms [45-47]. A potential field based skeletonization approach [45] for extracting 2D and 3D medial lines was proposed by Chuang *et al*. The boundary of a 3D object is assumed to be charged. The magnitude of the potential field is infinity at the boundary points and decreases with increasing distance from the boundary points. This definition is very similar to distance transform. The points along potential valleys (potential minima) are closely related to the medial lines. The algorithm generates skeletons as potential valleys using a Newtonian potential model. The test results showed that the skeletons obtained with this method are closely related to the corresponding MAT skeletons.

Visible repulsive force field [46] was used to generate medial lines for 3D objects. The algorithm has two steps. In the first step, all the boundary faces are charged. Seed points were initiated with negative charges from the 3D model vertices. These seed points are pushed by static force to converge to local minima. The visible repulsive force is the sum of all repulsive forces derived from the visible charged planes. Faces that are invisible from the seed point do not apply force to the seed point. In the second step, the local minima were connected to generate the skeletons.

Electrostatic field [31] can also be used as the field function. A skeleton is defined as the union of electrostatic field lines passing through local extrema. This

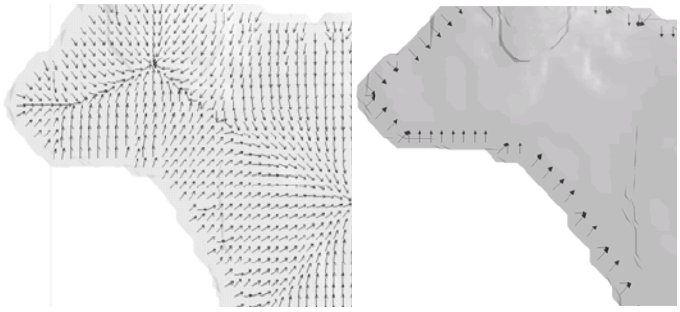
method has four steps. In this step, the space between the object and the non-object boundary area is modeled as a conductor of certain electrostatic potential and the interior region as a cavity in the conductor having a certain dielectric of permittivity. The potential distribution of the model is computed by the Jacob relaxation method. In the second step, the equipotential contour was calculated. The output contour was expressed with Freeman chain code. In the third step, the potential gradient along the contour was computed and then local extrema were detected. In the last step, the skeleton is generated by connecting the local extrema.

Cornea [47] proposed a hierarchical skeletonization algorithm to generate curve skeleton. The basic idea of this algorithm is to trace some critical points in a force field to generate skeleton.

This algorithm has three steps. First of all, it computes a vector field on a 3D model. This 3D model can be volumetric or polygonal. If the model is polygonal mesh, it must be converted into volumetric solid model by voxelization [75]. Secondly, some critical points are located in the vector field. And finally, the algorithm extracts the hierarchical curve-skeleton with a force following approach. In the last step, there are three options on seeds selection to initialize the force following approach. If the critical points are selected as seeds, the algorithm will extract the "core" skeleton (Level 0). If the lowest divergence voxels are chosen as seeds, the algorithm will generate a Level 1 skeleton with more branches and details. The algorithm can produce a Level 2 skeleton with even more branches and details if other user-selected seeds are included as seeds.

In the first step, the vector field is generated. It can be potential field, electrostatic field or repulsive force field. Two kinds of vector fields – Potential Field (PF) and Normal Diffusion Field (NDF) – are implemented. The basic idea of the PF based repulsive force approach is to generate a force field inside a 3D model by charging the model's contour. The repulsive force at a voxel by a nearby charge voxel pushes the voxel away from the charge. The strength (magnitude) of this force is inverse proportional to a power m of the distance between the point and the charge. The power m ($\in Z, \geq 2$) is called the order of

the force function. It plays a crucial role in shaping the vector field and determining the locations of critical points. The force is Newtonian force when it is 2. However, Newtonian force field does not generate stable results. Experiments showed that higher order power generated better and stable results. The potential field based repulsive force approach is not efficient. Therefore, NDF based approach was presented. In this approach, the repulsive force field is initialized to be normal to the surface of the 3D model. Then, the force is propagated into the model with an "onion pealing method". The NDF based approach is efficient. The computational complexity is *O(n)*, where *n* is the number of object voxels. However, NDF is more sensitive to noise than PF. Figure 2.16 shows the PF and NDF of a 3D cow model. It is obvious that the NDF is less dense than the PF. Therefore, the average effect of NDF is less that that of PF.



(a) (b)

Figure 2.16: Potential field (a) and normal diffusion field (b) of a 3D cow model, images courtesy of Cornea [47].

In the second step, critical points are detected. A critical point is defined as a point where the force vector vanishes. There are three kinds of critical points: *attracting (sink) points, repelling (source) points, and saddle points*. The force vectors flow toward an attracting point and away from a repelling point, from all directions. For a saddle point, some force vectors flow toward it and some flow away from it. Critical points are difficult to detect in a discrete space (integer voxel grid). The first reason is that they do not necessarily occur at any discrete location of the voxel grid, but often occur in-between some grid cells. Another reason is that fake critical point will be detected inside a grid cell when all

components of the vector field become zero, but at different locations, inside the grid cell. Newton's method was used to detect critical points based on the Jacobian matrix at a given position. It guarantees fast convergence when a critical point exists in a cell and quickly moves to other possible cells when a critical point does not exist in a cell. Critical points are classified according to the eigenvalues of the Jacobian matrix. An attracting point has negative real parts for all eigenvalues. A repelling point has positive real parts for all eigenvalues. A saddle point has both negative and positive real parts for the eigenvalues.

Three levels of skeletons are defined in [47]. Level 0 skeleton is defined as the skeleton generated by connecting critical points. Level 1 skeleton is defined as the skeleton generated by connecting lowest divergence seeds. Level 2 skeleton is defined as the skeleton generated by connecting user-selected seeds. However, as shown in Figure 2.17, only the Level 1 skeleton is useful.

Once the critical points are identified, a force-following algorithm is used to generate skeleton by connecting critical points (Level 0 skeleton), lowest divergence seeds (Level 1 skeleton), or user-selected seeds (Level 2 skeleton). Divergence is defined as a scalar value that measures the rate of the vector flow leaving a given point. To select seeds based on divergence, a local minimum criterion or a threshold can be used. A user-selected point can also be used as a seed. To maintain a strict hierarchical skeleton tree, a Level 2 skeleton must be a subset of a Level 1 skeleton and a Level 1 skeleton must be a subset of a Level 0 skeleton. Therefore, a Level 2 skeleton must be based on a Level 1 skeleton and a Level 1 skeleton must be based on of a Level 0 skeleton. The algorithm starts at a given seed. It computes the value of the vector field at the seed using tri-linear interpolation for x, y, and z directions. The algorithm stops when the value is zero (reached a critical point or moved outside of the object). Otherwise, it moves to the next position in the direction of the vector field. If the given seed is a critical point the force-following algorithm cannot move to new location because the value of the force vector is zero. In this special case, it moves out according to the eigen-direction at the critical point. If the next position is not a critical point, the force-following algorithm evaluate the value of the vector field to determine

where to move on; otherwise, moves out according to the eigen-direction. If the next position is a critical point and it has been visited, a backtracking method is used to move on to another branch of the skeleton.

Figure 2.17 shows a 3D cow model and its skeletons of different levels. This algorithm is efficient. The computational complexity is *O(n)* when normal diffusion field is used in the first step, where *n* is the number of object voxels. However, like other general field algorithms, the connectivity is not guaranteed because some critical points may not be connected to any of others. Figure 2.18 shows two models (ET and helicopter) with disconnected skeletons.

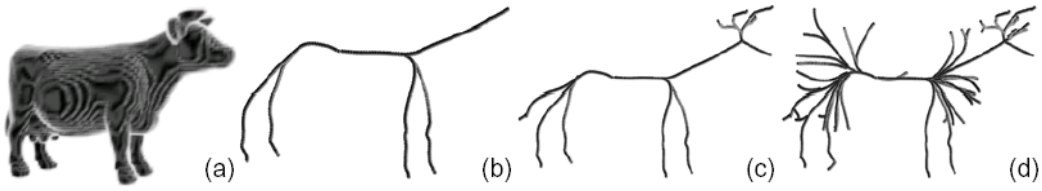


Figure 2.17: (a) A 3D cow model (b) Level 0 skeleton (c) Level 1 skeleton (d) Level 2 skeleton, images courtesy of Cornea [47].

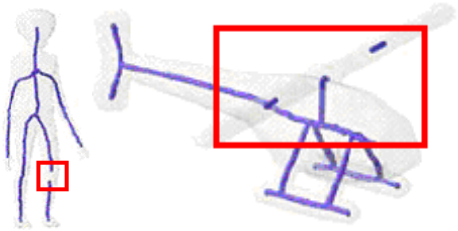


Figure 2.18: Models with disconnected skeletons, images adapted from [47].

2.4 Voronoi diagram based 3D skeletonization algorithms

The Voronoi diagram of a discrete point set (generating point set) is the partition of the given space into cells, where:

- Each cell contains exactly one generating point;
- All the points in a cell are closer to this generating point than to any other generating point.

Figure 2.19 (a) shows a point set *P* and Figure 2.19 (b) shows the Voronoi diagram of *P*. Points *a* and b are two generating points. Cell R_a contains point *a* only and cell R_b contains point b only.

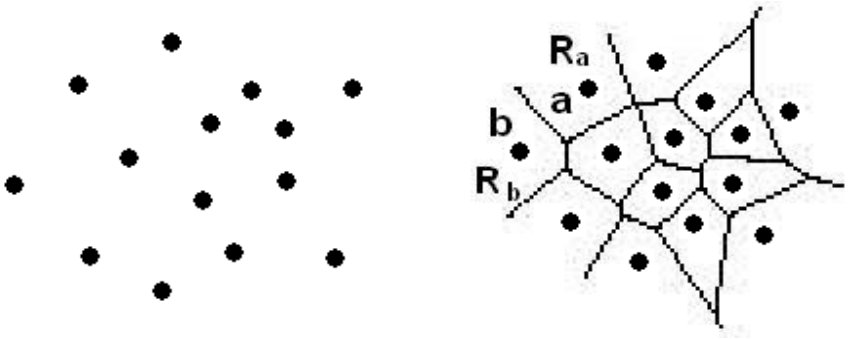


Figure 2.19: (a) Point set *P*. (b) Voronoi diagram.

The Voronoi diagrams can be computed by an incremental construction [67] of perpendicular bisectors between two generating points. Figure 2.20 shows this process. All the points in a cell are closer to this generating point than to any other generating point because the cell is enclosed by perpendicular bisectors.

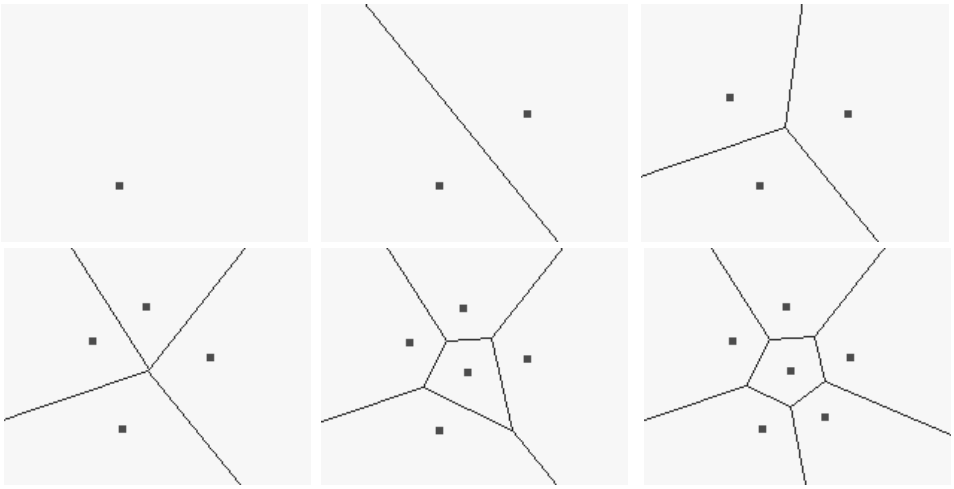


Figure 2.20: Computing Voronoi diagram by construction of perpendicular bisectors [67].

If we take the boundary points as generating points, then the Voronoi diagram converges to the skeleton, when the boundary points are dense enough. Figure 2.21 shows how to compute skeletons with Voronoi diagram. However, there are three drawbacks for Voronoi diagram based skeletonization algorithms:

- **Problem 2.4.1** When the boundary points are not dense enough the Voronoi diagram does not converge to the skeleton. Clustering methods must be used to connect skeletal points.
- **Problem 2.4.2** Computing the Voronoi diagram is computationally expensive.
- **Problem 2.4.3** Voronoi diagram based methods are very sensitive to noise; even the slightest disturbance of the boundary points can cause additional branches on skeletons.

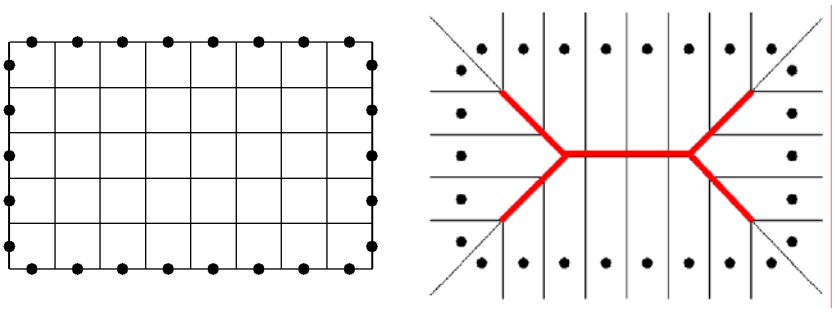


Figure 2.21: Voronoi diagram converges to the skeleton in [67].

Problem 2.4.1 and Problem 2.4.3 were tackled by a number of previous works [48-52]. However, Problem 2.4.2 is still an open problem.

Ogniewicz [48-49] tackled the Problem 2.4.3 by introducing a regularization method and the skeleton pyramid. The skeletons were represented by Voronoi Medial Axis (VMA). The Doubly Connected Edge List (DCEL) was used as the data structure of Voronoi Diagram (VD). The VMA consists of straight line segments and parabola. However, due to the Problem 2.4.3, the resulting VMA had many branches on skeletons. A regularization method was then used. It introduced four residual functions, potential residual, circularity residual, bi-circularity residual, and chord residual. The basic idea of the residual functions is to remove small branches less than some thresholds. The skeleton pyramid is a hierarchical tree of skeletons with different parameters. An adaptive skeleton traversal algorithm was used to build up a residual table. Two parameters were used to construct the skeletons with the residual table. The first parameter represents a trigger level of the hierarchy. The second parameter denotes the granularity of clustering of a residual table. With the regularization method and

the skeleton pyramid, the resulting Voronoi skeletons (VSK) were largely invariant to noise and geometric transformations. Some applications, such as object recognition and interpretation of road maps, were discussed in the paper.

Sheehy [51] proposed a Domain Delaunay Triangulation (DDT) based 3D skeletonization method for calculating medial surfaces. The Delaunay triangulation is the dual of the Voronoi diagram. The key idea is as follows. First of all, a description of the medial surface was associated with topological elements. As discussed in Section 2.1, a medial surface consists of the loci of the centers of all inscribed maximal spheres of a 3D model, where these spheres share at least two points with the boundary of the model. A sphere can be determined by four parameters: the three coordinates of its center and its radius. Thereby, four distinct non-coplanar points can form four equations to uniquely determine a sphere. Less distinct points can form a sphere with a higher degree of freedom. For instance, two distinct points can form a sphere with two degrees of freedom. A medial face, which was associated with E_TYPE Delaunay tetrahedron [51], was defined by the locus of the center of a sphere with two degrees of freedom. Medial edge, medial vertex and end point were defined in a similar way and were associated with different types of Delaunay tetrahedrons. In the second step, the Problem 2.4.1 was resolved by an adaptive refinement approach. This approach is guided by applying topological and geometric tests to the Delaunay simplexes. Once topological misrepresentations of the medial surface are identified, the invalid tetrahedrons are resolved by refining the boundary object point distribution using a tracing technique. In this step, rogue tetrahedrons are removed based on the application of the Newton and Singular Value Decomposition (SVD) methods. Then, the skeletal points were assembled to generate the resulting medial surfaces.

Turkiyyah [52] described an algorithm using accelerated Delaunay triangulation and local numerical optimization algorithms to generate 3D skeletons. The algorithm alleviated the point density requirement for Voronoi diagrams and accelerated the slow convergence of Voronoi diagrams to the skeleton. The skeleton generation procedure had three steps. First of all, an

approximation of the skeleton was generated by extracting a subset of the Voronoi diagram. In the second step, the approximation of the skeleton was adjusted by a local optimization procedure to generate the interior of the skeleton. In the last step, skeletal edge points– which were defined as the points of maximum principal surface curvature – were located with a specialized optimization method.

2.5 Shock graph based 3D skeletonization algorithms

Giblin and Kimia [69] classified 3D medial axis points into five types of points (shocks), which were then organized into sheets, curves, and vertices. Leymarie and Kimia [70-74] proposed the concept of shock scaffold and use it to approximate medial axis. The name "shock scaffold" comes from the scaffoldings used to erect buildings.

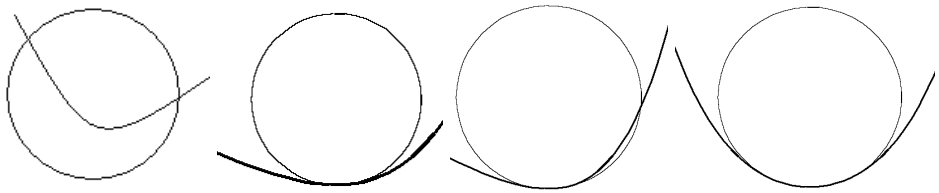
Definition 2.4: Two curves have a *1-point curve contact* if they intersect at a point but they are not tangent [68, 69]. See Figure 2.22 (a).

Definition 2.5: Two curves have a *2-point curve contact* if they intersect at a point and they are tangent [68, 69]. See Figure 2.22 (b).

Definition 2.6: Two curves have a 3-*point curve contact* if they intersect at a point and curvatures of the curves are equal [68, 69]. See Figure 2.22 (c).

Definition 2.7: Two curves have an *4-point curve contact* if they intersect at a point and the derivatives of the curvature are equal [68, 69]. See Figure 2.22 (d).

The types of contact [69] are classified into A_k series, where $k = m-1$ (A_0: 1-point contact, A_1: 2-point contact, A_2: 3-point contact, A_3: 4-point contact). The "A" means simple Lie groups [74] of type A. Figure 2.22 (a) shows two 1-point contacts, (b) shows an example of 2-point contact, (c) shows an example of 3-point contact, and (d) shows an example of 4-point contact.



(a) (b) (c) (d)

Figure 2.22: (a) two 1-point contacts (b) a 2-point contact (c) a 3-point contact (d) a 4-point contact. Images adapted from [68].

Definition 2.1 (see Section 2.1) requires the hyper-spheres are inscribed in a manifold, therefore, only 2-point contact (A_1) and 4-point (A_3) contact can contribute to a skeleton.

There are five categories of 3D medial axis points [69]: A_1^2, A_1^3, A_3, A_1^4, and A_1 A_3. A_k^n denotes n distinct A_k contact points. A medial axis is composed of sheets, curves, and vertices. A *sheet* (surface patch) consists of A_1^2 contact points. There are two kinds of curves: A_1^3 *curve* and A_3 *curve*. An A_1^3 curve is the intersection of three sheets. An A_3 curve is the boundary of a sheet. There are two types of vertices: A_1^4 *vertex* and A_1 A_3 *vertex*. An A_1^4 vertex is the intersection of four A_1^3 curves. An A_1 A_3 vertex is the intersection of an A_1^3 curve and an A_3 curve.

The shock scaffold [70] is a dynamic approximation of the medial axis. A shock propagates from boundaries with associated direction and speed of flow, as in Blum's grassfire [3].

Let A_k^n-m denotes m^{th} order shock. There are 3 kinds of 1^{st} *order shocks* (or regular shocks): A_1^2-1 (along a sheet) shock, A_1^3-1 (along an A_1^3 curve) shock, and A_3-1 (along an A_3 curve) shock. Flow goes smoothly through a 1^{st} *order shock* point.

There are 5 kinds of 2^{nd} *order shocks* (or shock source): A_1^2-2 (along a sheet) shock, A_1^3-2 (along an A_1^3 curve) shock, A_3-2 (along an A_3 curve) shock, A_1 A_3-2 (at an A_1 A_3 vertex) shock, and A_1^4-2 (at an A_1^4 vertex) shock. Flow begins at a 2^{nd} *order shock* point.

There are 5 kinds of 3^{rd} *order shocks* (or shock relay): A_1^2-3 (along a sheet) shock, A_1^3-3 (along an A_1^3 curve) shock, A_3-3 (along an A_3 curve) shock, A_1^4-3 (at an A_1^4 vertex) shock, and A_1 A_3-3 (at an A_1 A_3 vertex) shock. Flow begins and ends at a 3^{rd} *order shock* point.

There are 5 kinds of *4th order shocks* (or shock sink): A_1^2-4 (along a sheet) shock, A_1^3-4 (along an A_1^3 curve) shock, A_3-4 (along an A_3 curve) shock, A_1^4-4 (at an A_1^4 vertex) shock, and $A_1\ A_3$-4 (at an $A_1\ A_3$ vertex) shock. Flow ends at a *4th order shock* point.

Points on medial axis were therefore classified into 18 types of shock points. The 1st order shock points need not be computed explicitly. The basic idea of shock scaffold [70] based skeletonization algorithm is to trace the other 15 kinds of shock points, link them to generate shock scaffold and then use it to approximate the medial axis.

Definition 2.8 (Shock nodes, *P*) The set of *shock nodes*, denoted *P*, consists of shock sources, relays, sinks, and all remaining types of vertices [70].

Definition 2.9 (Shock links, *L*) The set of *shock links*, denoted *L*, is an ordered array (by the radius function) consisting of the curve segments (with geometry and direction) each between two shock nodes [70].

Definition 2.10 (Shock hyperlinks, *H*) The set of *shock hyperlinks*, denoted *H*, is an ordered, cyclic array of shock nodes of its boundary. A hyperlink is attributed with geometry and orientation of the sheet [70].

Definition 2.11 (Shock scaffold, *SC*) The shock scaffold, denoted *SC*, is a geometric directed graph with shock nodes *P* and shock links *L* [70].

The Voronoi graph was proved to be a sub-graph the shock scaffold in [72]. Therefore, shock scaffold can be used to approximate the medial axis. Shock scaffold can be augmented (*augmented shock scaffold*, SC^+) or reduced (*reduced shock scaffold*, SC^-), where $SC^+ \supset SC \supset SC^-$. They form a three-tier hierarchy representing the medial axis.

Brute-force computing shock scaffold is very expensive. A Lagrangian method of computing shock scaffold was proposed in [71]. The name Lagrangian comes from the dynamical system of shock flow. This method propagates along the scaffold from initial shock sources. Seven principles were presented to avoid the computation of those pairs of shock points that will not lead to a shock flow. The seven principals [71] are:

(1) One pair of shock points should be visible to each other,

(2) The group of shock points,

(3) The visibility of a group from another,

(4) The convex hull of a group,

(5) The vertices of such convex hulls as virtual points,

(6) A coarse-to-fine framework,

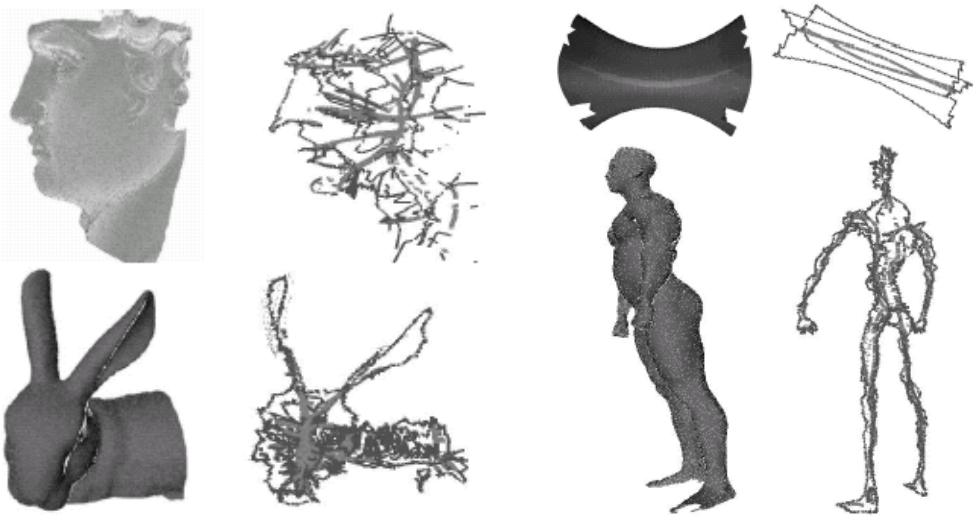
(7) A search method organized in layers.

The proposed Lagrangian method [71] does not require a fixed (Eulerian) grid. Instead, it depends on the shock scaffold itself as the fundamental grid. The basic idea of this approach is that the full bisectors do not have to be explicitly computed if a dynamical system of shock flow is adopted and shock sources, shock relays and shock sinks of flow are completely identified. Then, only the bisectors that initiate from valid shock sources need to be computed and all others can be ignored completely. Therefore, it features the exactness of bisector computations, but without the computational cost associated with them. This algorithm works for any initial shape geometry. Figure 2.23 shows some 3D shapes (unorganized point clouds) and their shock scaffolds. The applications, such as 3D shape reconstruction [73] and registration [74], of shock scaffold were investigated.

Siddiqi [76-78] proposed a different way to detect and track shock. The basic idea [78] is to detect locations (shocks) where a conservation of energy principle is violated based on measurement of the net outward flux of a vector field. This algorithm has two steps. In the first step, a Lagrangian method was introduced to simulate the eikonal equation for curve evolution. The wave front of the curve evolution was explicitly represented as a sequence of marker particles. The motion of these particles was controlled by a Hamiltonian system. This system was then interpreted by Hamiltonian and Lagrangian mechanics. In the second step, an efficient algorithm was proposed to detect shock based on the net outward flux per unit volume of the vector field in the Hamiltonian system. It is based on the fact that a Hamiltonian system is conservative [79]; however, energy will be absorbed on shock points so that the energy conservation will be violated. Therefore, a shock point is detected at the location the energy conservation is

violated. The authors reported that the efficiency of this algorithm is similar to the fast marching [80] algorithm. Some results are shown in Figure 2.24. A recent extension [81] of this work used a coarse-to-fine method to extract medial surfaces from polyhedral meshes.

It is difficult to apply the algorithm in continuous domain and two problems in discrete space. First of all, the algorithm needs a continuous representation of a model's boundary to place a dense sequence of marker particles. However, most 2D or 3D objects are represented by surface meshes, point clouds, or solid grids. The boundary of the input model must be detected and converted to a continuous form to initialize this algorithm. Secondly, since a shock point (singularity) is not differentiable, numerical computation will lead to errors near shocks. The authors suggested ENO (essentially non-oscillatory) [76] interpolations for estimating derivatives.



(a) (b) (a) (b)

Figure 2.23: (a) 3D shapes and their shock scaffolds (b). Images courtesy of Leymarie [71].

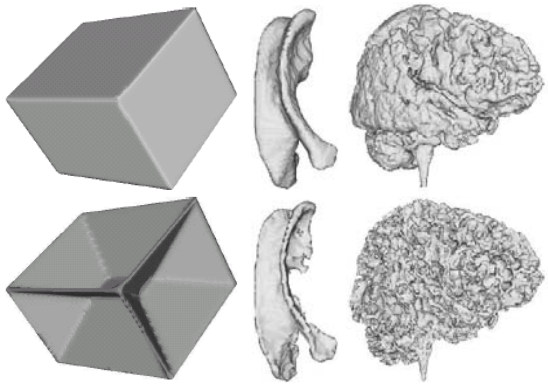


Figure 2.24: First row: some 3D objects (box, ventricles of brain, and the outer surface of a brain). Second row: the corresponding skeletons. Images courtesy of Siddiqi [78].

2.6 Important properties of skeleton or skeletonization

A good skeleton should be a useful, compact and faithful representation of a 3D object. In the literature [9, 47], a useful skeleton or skeletonization algorithm must have two important properties: **thin** and **connectivity preservation**. Thinness is important because the skeleton must be compact. Connectivity preservation is important because the skeleton should represent the 3D object faithfully and therefore should not break the skeleton of a connected 3D object into pieces. There are some other desirable properties for skeleton or skeletonization algorithm. They will be discussed in Section 6.1.

A skeleton should be thin because it is expected to be a compact representation of a 3D object. The extremely thin skeleton is called a ***unit-width curve skeleton***. A skeleton is unit-width if it is only one voxel thick. The formal definition of unit-width curve skeleton is given in Section 3.3.1. We will focus on unit-width curve skeleton in the next chapter.

A connectivity-preserving skeleton must satisfy the following condition:

Condition 2.6.1 there is a one-to-one mapping between the connected components of the object and the connected components of the skeleton, and vice-versa;

Connectivity preservation is closely related to topology (or geometry) preservation. A topology-preserving skeleton must satisfy condition (2.6.1) and the following two more conditions:

Condition 2.6.2 there is a one-to-one mapping between the cavities of the object and the cavities of the skeleton, and vice-versa;

Condition 2.6.3 there is a one-to-one mapping between the tunnels of the object to the tunnels of the skeleton, and vice-versa.

However, since a skeleton should be a compact representation of a 3D object, skeletonization algorithms often [9, 24-27, 47] remove cavities and tunnels to make the skeletons more compact. Therefore, conditions (2.6.2) and (2.6.3) have often, if not always, been ignored. Thus the remaining condition (2.6.1) becomes very crucial to ensure that a skeleton represents a 3D object as faithfully as possible. So, topology preservation is compromised and becomes to "connectivity preservation". If a 3D skeletonization algorithm preserves connectivity, it will never generate disconnected skeletons for a connected 3D object. Therefore, the skeleton it generated is considered [9, 47] to be a faithful representation of the 3D object. In the rest of this chapter, we will focus on connectivity preservation.

2.7 The problem of Ma and Sonka's algorithm and a solution

A 3D skeletonization algorithm should preserve connectivity. If a skeletonization algorithm fails to preserve connectivity, the skeletons extracted from the object will be disconnected, which is unacceptable in many applications. However, by studying the configuration in Figure 2.25, we find that Ma and Sonka's algorithm [9] fails to do so. Figure 2.25, wherein a "•" is an object point and all other points are background points, shows a 26-connected 3D object a-b-c-d-e-f-g. In Ma and Sonka's thinning algorithm, points *c*, *d* and *e* will be deleted because *c* satisfies template *a5* in Class A, *d* satisfies template *d7* in Class D and *e* satisfies template *a6* in Class A. However, the deletion of points *c, d* and *e* leads to disconnection of the object. See Section 2.2.2 for more details of the deleting templates.

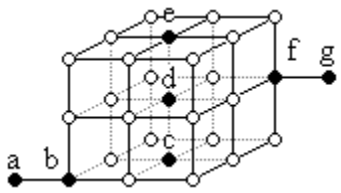


Figure 2.25: A connected object a-b-c-d-e-f-g in 3D space. A "•" is an object point. A "◦" is a background point. All other points in 3D space are background points. In Ma and Sonka's algorithm, point *c* will be deleted by template *a5* in Class A, point *d* will be deleted by template *d7* in Class D and point *e* will be deleted by template *a6* in Class A. Hence, the object will be disconnected.

Lohou discovered this problem and gave a counter example of Ma and Sonka's algorithm in [18]. Some other researchers, such as Chaturvedi [16], applied this algorithm and found it disconnected small segments but did not suggest how to fix this problem. In this section, we will explain the cause of this problem and how to modify the templates in Class D to solve this problem.

Ma and Sonka proposed a general theorem [9] and used it to prove that the 3D thinning algorithm preserves connectivity in the VERIFICATION section in that paper. According to our observation, we note that LEMMA 3.5 in the VERIFICATION section is problematic. "**LEMMA 3.5**: *Let p, q be two 6-adjacent object points in a 3D binary image where both p and q satisfy* Ω. *Then either* $q \notin \Omega(p)$ *or* $p \notin \Omega(q)$."

Ω is used to denote the set of deleting templates in Class A, B, C or D. An object point satisfies Ω if it satisfies any one of the deleting templates in Ω. *"*$q \in \Omega(p)$" means that *q* must be an object point for *p* to satisfy Ω. ''$q \notin \Omega(p)$" means *p* still satisfies Ω after *q* is deleted.

For the 3D object in Figure 2.25, *c* and *d* are two 6-adjacent points. According to LEMMA 3.5, either $c \notin \Omega(d)$ *or* $d \notin \Omega(c)$. However, if *c* is deleted, *d* will not satisfy any of the deleting templates. And, if *d* is deleted, *c* will not satisfy any of the deleting templates. Therefore, although *c* and *d* are 6-adjacent, $c \in \Omega(d)$ and $d \in \Omega(c)$. We can prove that for points *d* and *e*, $d \in \Omega(e)$ and $e \in \Omega(d)$, although *d* and *e* are 6-adjacent, in the same way.

LEMMA 3.5 requires that for two 6-adjacent points *p* and *q,* if both *p* and *q* satisfy Ω, then either $q \notin \Omega(p)$ or $p \notin \Omega(q)$. Let *p1* and *p2* be the two "don't care" points in *p*'s 6-neighborhood, as showed in Figure 2.26. According to LEMMA 3.5, if *p1* is 1, then *p2* must be 0; if *p2* is 1 then *p1* must be 0. So (*p1, p2*) can be (0, 0), (0, 1) or (1, 0), but not (1, 1). There is no template in Class A-C that has a value of (1, 1) for (*p1, p2*), however, the deleting templates in Class D violate this rule. For instance, in template *d7*, (*p1, p2*) is (1, 1), which fails LEMMA 3.5.

Based on this observation, we can change the deleting template *d1-d12* to satisfy LEMMA 3.5. For instance, we change template *d7* to three new templates as shown in Figure 2.27, according to different values of (*p1, p2*). In this way, we change the 12 deleting templates of Class D to 36 deleting templates according to different values of (*p1, p2)*. So there are 36 templates in Class D, and 6 + 12 + 8 + 36 = 62 templates in total. Figure 2.28 shows the modified templates in Class D. Each template in Class D is changed to three templates, in which (*p1, p2*) are (0, 0), (0, 1) or (1, 0) respectively. Since (*p1, p2*) is not (1, 1) in any template, LEMMA 3.5 is satisfied for the new set of templates.

Since LEMMA 3.5 is corrected, the reasoning process in the original paper becomes sound and complete. Therefore, a formal mathematical proof can be presented to prove that the modified algorithm preserves connectivity by replacing the old templates in LEMMA 3.5 with the new ones. And the other parts of the proof are exactly the same with the reasoning process in the original paper.

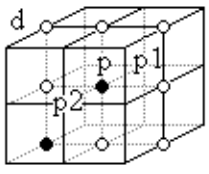


Figure 2.26: Template core of Class D.

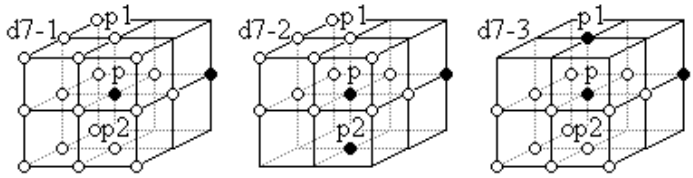


Figure 2.27: Template d7-1 to d7-3.

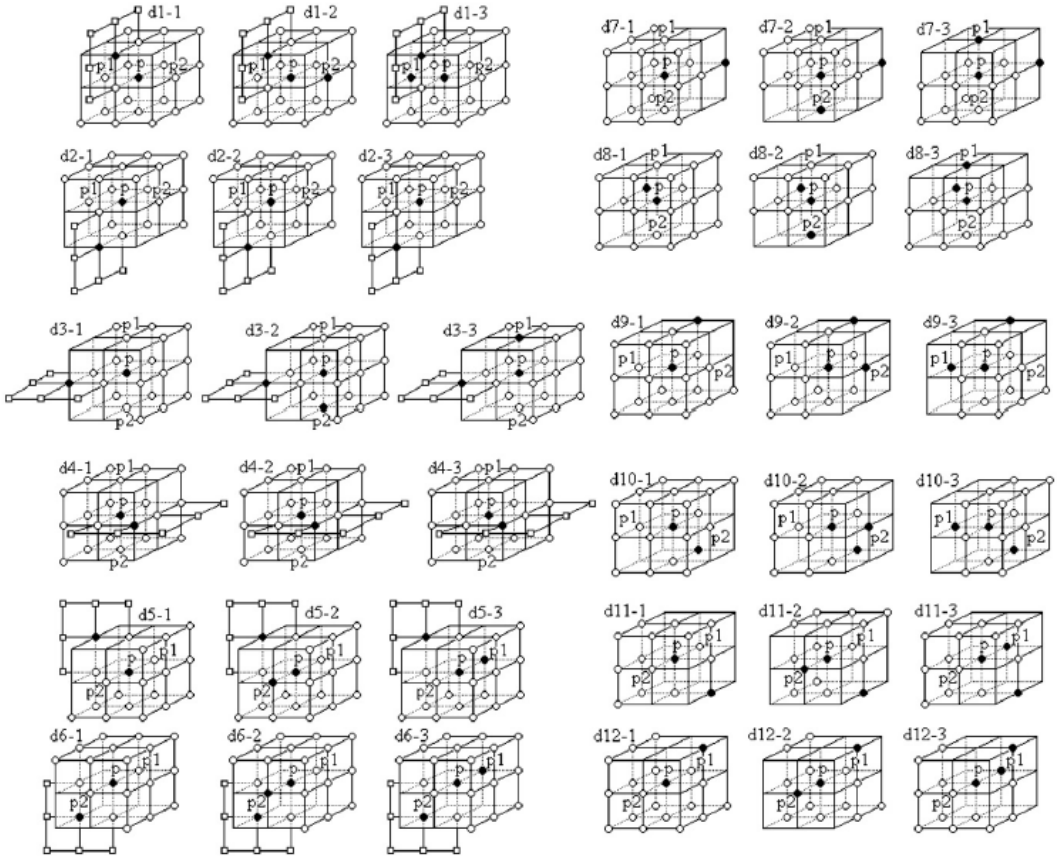
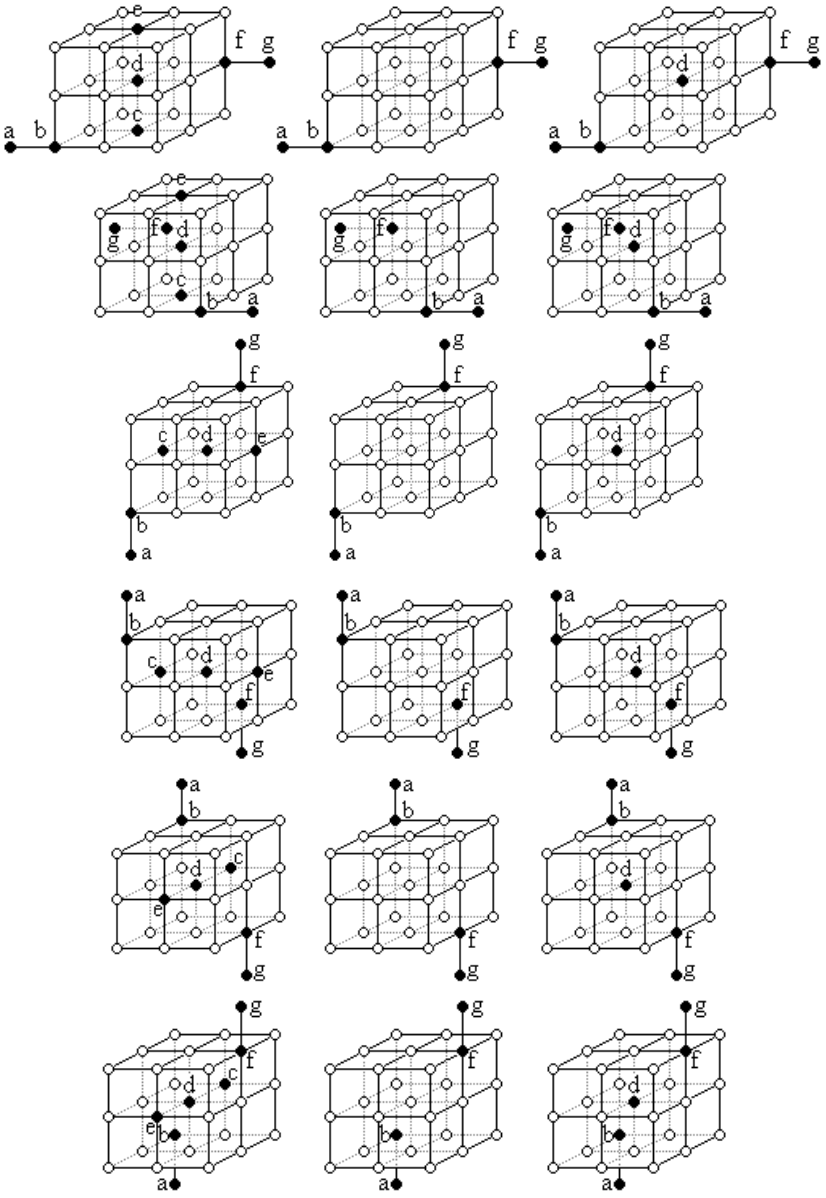


Figure 2.28: The modified deleting templates in Class D. Each template in Class D is changed to three templates, in which (*p1, p2*) are (0, 0), (0, 1) or (1, 0) respectively. At least one point marked □ is an object point.



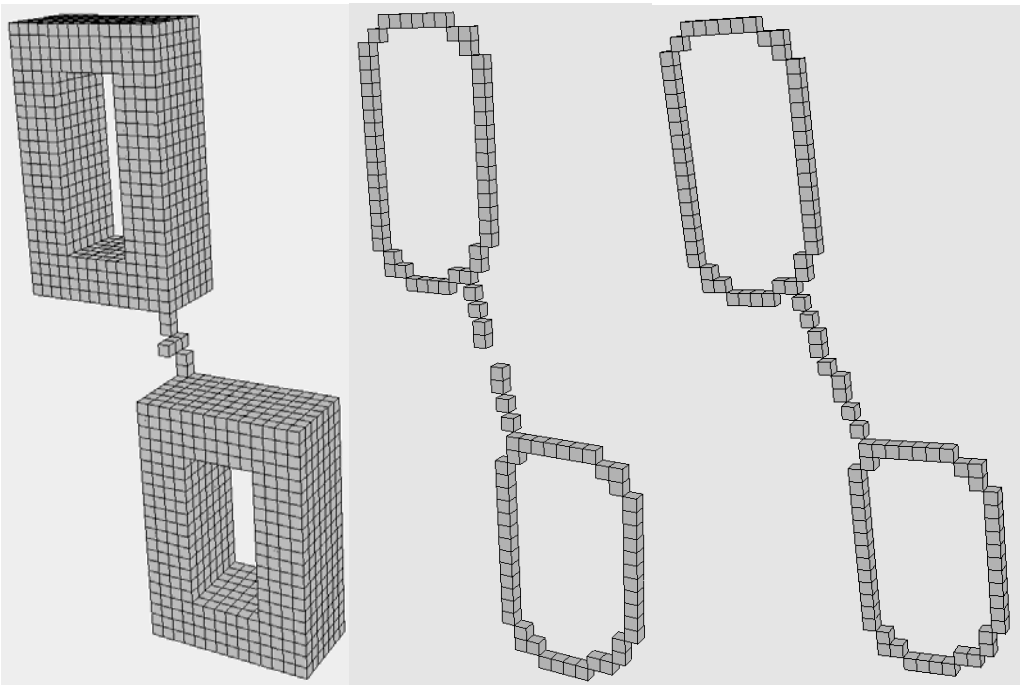
(a) (b) (c)

Figure 2.29: A "●" is an object point. A "◦" is a background point. All other points in 3D space are background points. (a) The original 3D object a-b-c-d-e-f-g. (b) The thinning result of Ma and Sonka's algorithm. Point *c*, *d* and *e* are deleted by some templates in Class A and Class D. Thus, the object gets disconnected. (c) The thinning result of the modified algorithm. Points *c* and *e* are deleted by some templates in Class A, but point *d* is not deleted, thus the object is still connected.

Figure 2.29 shows some different results of Ma and Sonka's algorithm and the modified one. Six 26-connected 3D objects are shown in (a). For Ma and Sonka's algorithm, point *c, d* and *e* are deleted, thus the connected object is disconnected after thinning, as shown in (b). For the modified algorithm, point *c* and *e* are deleted, but point *d* will not be deleted, thus connectivity is preserved after the thinning operations, as shown in (c).

2.8 Experimental results

Figure 2.30 shows a different result between Ma and Sonka's algorithm and the modified algorithm [109]. Two connected "0"s are shown in Figure 2.30(a). For Ma and Sonka's algorithm, the connected object is disconnected after thinning, as shown in Figure 2.30 (b). For the modified algorithm, the connected object is still connected after the thinning operations, as shown in Figure 2.30 (c). Figure 2.31 shows some real models and their skeletons generated by the modified algorithm.



(a) (b) (c)

Figure 2.30: (a) Original 3D object; (b) Result of Ma and Sonka's algorithm; (c) Result of modified algorithm.

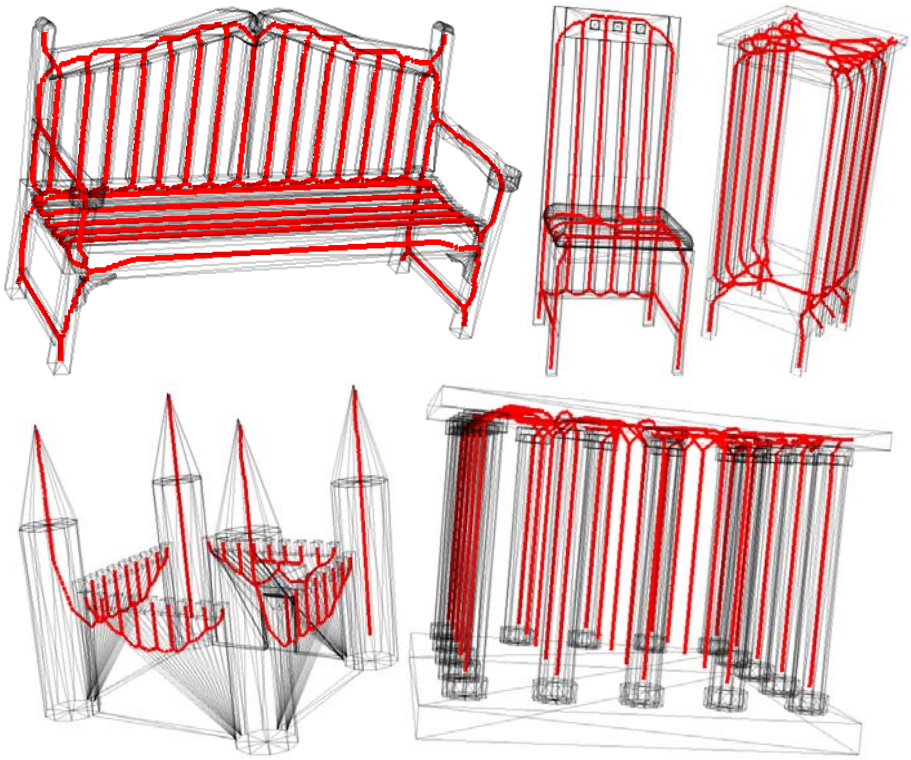


Figure 2.31: Some real models and their skeletons.

2.9 Discussions and Conclusions

This chapter presents a 3D skeletonization study on connectivity preservation. The motivation behind this work was that we found Ma and Sonka's algorithm [9] failed to preserve connectivity for some configurations, which is very important for 3D skeletonization algorithms. We then studied why Ma and Sonka's algorithm failed and proposed a solution to the problem. Experimental results demonstrate the validity of our solution. However, we observe that the modified algorithm cannot guarantee to extract unit-width skeleton, which is required by many applications such as retrieval and matching. Some examples of non-unit width skeletons are shown in Figure 2.32.

Note that the non-unit width skeletons are still useful in some applications [16]. We had used the non-unit width skeletons for automatic estimation of 3D transformation for object alignment [110-111]. Skeletons of the 3D objects are created using the modified algorithm [109] proposed in this chapter, feature point pairs (land markers) are extracted from skeletons automatically, and a least

squares method is applied to solve an over determined linear system to estimate the 3D transformation matrix. Experiments show that this non-unit width skeleton based method works quite well with high accuracy when the translations and rotation angles are small. However, many other applications [47, 61] require unit-width curve skeletons. In the next chapter, we will introduce a method to generate unit-width skeletons.

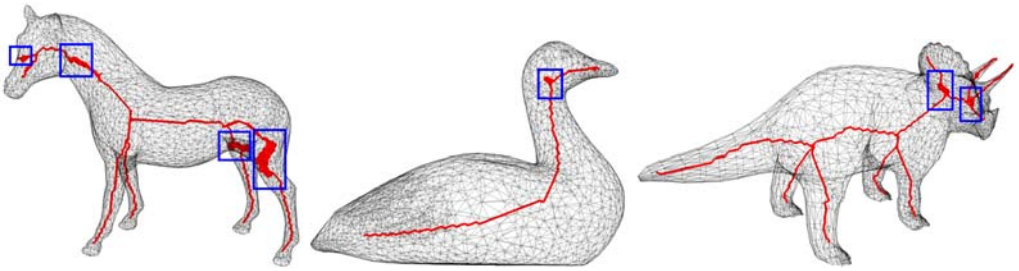


Figure 2.32: Examples of non-unit width skeletons.

Bibliography

[1] P. J. Besl and R. C. Jain. "Three-dimensional object recognition". *ACM Computing Surveys*, 17 (1): pp 75-145, 1985.

[2] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin and D. Jacobs. "A Search Engine for 3D Models", *ACM Trans. on Graphics*, 22(1): pp 83-105, 2003.

[3] H. Blum. "A transformation for extracting new descriptors of shape", *Models for the Perception of Speech and Visual Form*, pp. 362–380, MIT Press, Cambridge, MA, USA, 1967.

[4] K. Palagyi and A. Kuba. "A 3D 6-subiteration thinning algorithm for extracting medial lines", *Pattern Recognition Letters*, 19 (7): pp 613-627, 1998.

[5] C. Lohoua and G. Bertrand. "A 3D 6-subiteration curve thinning algorithm based on P-simple points", *Discrete Applied Mathematics*, Vol. 151, pp 198–228, 2005.

[6] W. X. Gong and G. Bertrand. "A simple parallel 3D thinning algorithm", In *Proceedings of* ICPR 1990, pp 188-190.

[7] P. K. Saha, B. B. Chaudhury and D. D. Majumder. "A new shape-preserving parallel thinning algorithm for 3D digital images", *Pattern Recognition*, 30 (12): pp 1939–1955, 1997.

[8] C. M. Ma and S. Y. Wan. "A medial-surface oriented 3-d two-subfield thinning algorithm", *Pattern Recognition Letters*, Vol. 22, pp 1439-1446, 2001.

[9] C. M. Ma, M. Sonka. "A fully parallel 3D thinning algorithm and its applications", *Computer Vision and Image Understanding*, 64 (3): pp 420-433, 1996.

[10] J. Mukherjee, P. P. Das and B.N. Chatterji. "Thinning of 3-D images using the Safe Point Thinning Algorithm", *Pattern Recognition Letters* Vol. 10, pp 167-173, 1989.

[11] J. Mukherjee, P. P. Das and B. N. Chatterji. "On connectivity issues of ESPTA", *Pattern Recognition Letter*, Vol. 11, pp 643-648, 1990.

[12] C. M. Ma and S. Y. Wan. "Parallel Thinning Algorithms on 3D (18, 6) Binary Images", *Computer Vision and Image Understanding*, Vol. 80, pp 364–378, 2000.

[13] C. M. Ma, S. Y. Wan, H. K. Chang. "Extracting medial curves on 3D images", *Pattern Recognition Letters* 23, 895–904, 2002.

[14] C. M. Ma. "On topology preservation in 3D thinning", *CVGIP: Image Understanding*, 59(3), pp. 328–339, 1994.

[15] C. M. Ma, S. Y. Wan, J. D. Lee. "Three-Dimensional Topology Preserving Reduction on the 4-Subfields", *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 12, 2002.

[16] A. Chaturvedi, Z. Lee. "Three-dimensional segmentation and skeletonization to build an airway tree data structure for small animals", *Physics in Medicine and Biology*, vol. 50 (7), pp. 1405-1419, 2005.

[17] M. S. Talukdar, O. Torsaeter, M. A. Ioannidis, J. J. Howard. "Stochastic reconstruction, 3D characterization and network modeling of chalk", *Journal of Petroleum Science and Engineering*, vol. 35 (1-2), pp 1-21, 2002.

[18] C. Lohou. "Contribution à l'analyse topologique des images", *Ph.D. thesis, UNIVERSITÉ DE MARNE-LA-VALLÉE*, 2001.

[19] J. B. A. Maintz and M. A. Viergever. "A Survey of Medical Image Registration", *Medical Image Analysis*, vol.2, p.1-36, 1998.

[20] L. G. Brown. "A survey of image registration techniques", *ACM Computing Surveys (CSUR)*, 24(4), p.325-376, 1992.

[21] P. J. Besl and N. D. Mckay. "A method for registration of 3D shapes", *IEEE Trans. Pattern Anal. Machine Intell*. 14(2), p. 239-256, 1992.

[22] http://www.sph.sc.edu/comd/rorden/dicom.html

[23] http://www.amiravis.com

[24] C. M. Ma, "A 3D fully parallel thinning algorithm for generating medial faces". *Pattern Recognition Letter*, 16, pp83–87, 1995.

[25] A. Manzanera, T. M. Bernard, F. Preteux and B. Longuet. "Medial faces from a concise 3D thinning algorithm". *ICCV*, pp 337–343, 1999.

[26] K. Palagyi, "A 3-subiteration 3D thinning algorithm for extracting medial surfaces", *Pattern Recognition Letters*, 23, pp663–675, 2002.

[27] K. Palagyi, A. Kuba. "Directional 3D Thinning Using 8 Subiterations", *Lecture notes in computer science* 1568: 325-336 1999.

[28] D. Hearn and M. P. Barker. *Computer Graphics (second edition)*, Pearson Education.

[29] K. Palagyi and A. Kuba, "A Parallel 3D 12-Subiteration Thinning Algorithm", *Graphical Models and Image Processing*, 61, 199–221, 1999.

[30] C. Lohou and G. Bertrand. "A 3D 12-subiteration thinning algorithm based on P-simple points", *Discrete Applied Mathematics*, 139, 171 – 195, 2004.

[31] G. H. Abdel-Hamid, Y.-H. Yang. "Multiresolution skeletonization: an electrostatic field based approach", *ICIP*, pp 949-953, 1994.

[32] J. J. Leader. *Numerical analysis and scientific computation*, Pearson Addison Wesley, Boston, 2004.

[33] J. Maciel. "Global matching: optimal solution to correspondence problems", *PhD Thesis, Universidade Técnica de Lisboa*, 2002

[34] P. Torr. "Motion Segmentation and Outlier Detection". *PhD thesis, U. Oxford*, 1995.

[35] J.Domke and Y.Aloimonos, "A Probabilistic Notion of Correspondence and the Epipolar Constraint", *Third International Symposium on 3D Data Processing, Visualization and Transmission*, June 2006.

[36] C. Pudney. “Distance-Ordered Homotopic Thinning: A Skeletonization Algorithm for 3D Digital Images”, *Computer Vision and Image Understanding*, 72(3):404-413, 1998.

[37] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, 2nd ed., Vol. 2, Academic Press, New York, 1982.

[38] C. Arcelli and G. S. di Baja, “A width independent fast thinning algorithm”, *IEEE Trans. Pattern Anal. Mach. Intell.* 7, 1985, 463–474.

[39] Y. Zhou, A. Kaufman and A. W. Toga. “Three-dimensional Skeleton and Centerline Generation Based on an Approximate Minimum Distance Field”, *The Visual Computer*, 14, pp 303-314, 1998.

[40] Y. Zhou, P.M. Thompson, A.W.Toga. “Extracting and Representing the Cortical Sulci”, *IEEE computer graphics and applications*, 19 (3): 49-55 may-jun 1999

[41] Y. Zhou and A.W.Toga. “Efficient skeletonization of volumetric objects”, *IEEE transactions on visualization and computer graphics,* 5 (3): 196-209, 1999.

[42] M. Wan, F. Dachille and A. Kaufman. “Distance-Field Based Skeletons for Virtual Navigation”, *IEEE Visualization*, 2001.

[43] H. Sundar, D. Silver, N. Gagvani and S. Dickinson. “Skeleton Based Shape Matching and Retrieval”, *Shape Modeling International*, 2003.

[44] M. Couprie, D. Coeurjolly and R. Zrour. “Discrete bisector function and Euclidean skeleton in 2D and 3D”, *Image and Vision Computing*, 25, 1543–1556, 2007.

[45] J. Chuang, C. Tsai and M.-C. Ko. “Skeletonization of three-dimensional object using generalized potential field”, *IEEE PAMI,* 22(11):1241-1251, 2000.

[46] F. Wu, W.-C. Ma, P. Liou, R. Liang and M. Ouhyoung. “Skeleton extraction of 3d objects with visible repulsive force”, *Eurographics Symposium on Geometry Processing*, 2003.

[47] N. D. Cornea. “Curve-Skeletons: Properties, Computation And Applications”, *Ph.D. Thesis, The State University of New Jersey*, May 2007

[48] R. L. Ogniewicz and M. Ilg. “Voronoi Skeletons Theory and Applications”, *CVPR,* 1992.

[49] R. L. Ogniewicz and O. Kubler. “Hierarchic Voronoi Skeletons », *Pattern Recognition*, 28 (3): 343-359 Mar 1995.

[50] E. C. Sherbrooke, N. M. Patrikalakis and E. Brisson. “An algorithm for the medial axis transform of 3d polyhedral solids”, *IEEE Transactions on Visualization and Computer Graphics*, 2 (1): 44-61 Mar 1996.

[51] D. J. Sheehy, C. G. Armstrong and D. J. Robinson. “Shape description by medial surface construction”, *IEEE Transactions on Visualization and Computer Graphics*, 2 (1): 62-72 Mar 1996.

[52] G. M. Turkiyyah, D. W. Storti and M.Ganter. “An accelerated triangulation method for computing the skeletons of free form solid models”, *Computer-Aided Design,* 29 (1): 5-19 JAN 1997.

[53] R. C. Veltkamp and M. Hagedoorn. "State-of-the-art in shape matching". *Technical Report UU-CS-1999-27*, Utrecht University, the Netherlands, 1999.

[54] T. B. Sebastian and B. B. Kimia. "Curves vs. skeletons in object recognition". In *Proceedings of IEEE International Conference of Image Processing*, 2001.

[55] P. E. Trahanias. "Binary Shape-Recognition Using The Morphological Skeleton Transform", *Pattern Recognition*, 25 (11): 1277-1288 Nov 1992.

[56] L. He, C. Y. Han, B. Everding, and W. G. Wee. "Graph matching for object recognition and recovery", *Pattern Recognition*, Vol. 37 Issue 7, 1557-1560, Jul 2004.

[57] M. Tanase and R. C. Veltkamp. "Polygon Decomposition based on the Straight Line Skeleton, Geometry, Morphology and Computational Imaging", *Lecture Notes in Computer Science,* 2616: 247-267 2003.

[58] C. D. Ruberto. "Recognition of shapes by attributed skeletal graphs", *Pattern Recognition,* 37 (1): 21-31 Jan 2004.

[59] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. "Skeleton based shape matching and retrieval". *SMI*, pp 130-139, 2003.

[60] D. P. Huttenlocher, G. A. Klanderman and W. J. Rucklidge. "Comparing Images Using The Hausdorff Distance", *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 15 (9): 850-863, Sep 1993.

[61] N. D. Cornea, D. Silver and P. Min. "Curve-Skeleton Applications", *IEEE Visualization*, pp 95-102, 2005.

[62] G. Malandain and G. Bertrand. "Fast characterization of 3D simple points", In *Proceedings of IEEE International Conference on Pattern Recognition*, 232–235, 1992.

[63] G. Bertrand and G. Malandain. "A new characterization of three-dimensional simple points", *Pattern Recognition Letters*, 15 (2): 169-175, Feb 1994.

[64] G. Borgefors. "On Digital Distance Transforms in Three Dimensions", *Computer Vision and Image Understanding*, 64(3):368-376, 1996.

[65] http://en.wikipedia.org/wiki/Euclidean_distance

[66] G. Borgefors. "Distance transformations in arbitrary dimensions", *Comput. Vision Graphics Image Process,* 27, 1984, 321–345.

[67] http://www.inf.u-szeged.hu/~palagyi/skel/skel.html

[68] http://en.wikipedia.org/wiki/Contact_%28mathematics%29

[69] P. Giblin and B. B. Kimia. "A formal classification of 3D medial axis points and their local geometry", *CVPR,* 2000.

[70] F. F. Leymarie and B. B. Kimia. "The Shock Scaffold for Representing 3D Shape", *LNCS 2059*, pp 216-229, 2001.

[71] F. F. Leymarie and B. B. Kimia. "Computation of the Shock Scaffold for Unorganized Point Clouds in 3D", *CVPR,* 2003.

[72] F. F. Leymarie. "Three-Dimensional Shape Representation via Shock Flows", *Ph.D. Thesis, Brown University, Division of Engineering*, May 2003.

[73] F. F. Leymarie, B. B. Kimia and P. J. Giblin. "Towards Surface Regularization via Medial Axis Transitions", *ICPR,* 2004.

[74] M.-C. Chang, F. F. Leymarie and B.B. Kimia. "3D Shape Registration using Regularized Medial Scaffolds", *3DPVT*, 2004.

[74] http://en.wikipedia.org/wiki/Lie_group

[75] M. Sramek and A.E. Kaufman. "Alias-Free Voxelization of Geometric Objects", *IEEE Transactions on Visualization and Computer Graphics*, 5(3): 251 – 267, 1999.

[76] K. Siddiqi, B. B. Kimia and C. W. Shu. "Geometric Shock-Capturing ENO Schemes for Subpixel Interpolation, Computation, and Curve Evolution", In *Proceedings of International Symposium on Computer Vision,* Coral Gables, USA, 437-442, 1995.

[77] K. Siddiqi and B. B. Kimia. "A Shock Grammar for Recognition", In *Proceedings of* Conference on Computer Vision and Pattern Recognition, San Francisco, USA, 507-513, 1996.

[78] K. Siddiqi, S. Bouix, A. Tannenbaum and S. W. Zucker. "The Hamilton-Jacobi Skeleton", In *Proceedings of International Conference on Computer Vision,* Corfu, Greece, September, 1999.

[79] L. Perko. *Differential Equations and Dynamical Systems*. Springer-Verlag, 1986.

[80] J. A. Sethian. "A fast marching level set method for monotonically advancing fronts". *Proc. Natl. Acad. Sci*. USA, Vol. 93, pp 1591-1595, February 1996.

[81] S. Stolpner and K. Siddiqi. "Revealing Significant Medial Structure in Polyhedral Meshes", In *Proceedings of 3rd International Symposium on 3D Data Processing*, Visualization and Transmission, 2006.

[82] H. Lester and S. R. Arridge. "A survey of hierarchical non-linear medical image registration". *Pattern Recognition*, 32:129–49, 1999.

[83] D. L. G. Hill, P. G. Batchelor, M. Holden and D. J. Hawkes. "Medical image registration". *Phys Med Biol,* 46: pp1–45, 2001.

[84] B. Zitova and J. Flusser. "Image registration methods: a survey". *Image Vision Comput*., 21: pp977–1000, 2003.

[85] W. R. Crum, T. Hartkens, and D. L. G. Hill. "Non-Rigid Image Registration: theory and practice", *The British Journal of Radiology*, 77, pp. 140–153, 2004.

[86] T. Makela, P. Clarysse, O. Sipila, N. Pauna, Q. C. Pham and T. Katila. "A review of cardiac image registration methods", *IEEE Trans Med Imaging,* 21:1011–21, 2002.

[87] B. F. Hutton, M. Braun, L. Thurfjell and D. Y. H. Lau. "Image registration: an essential tool for nuclear medicine". *Eur. J. Nucl. Med. Mol. Imaging*, 29: pp59–77, 2002.

[88] J. G. Rosenman, E. P. Miller, G. Tracton and T. J. Cullip. “Image registration: an essential part of radiation therapy treatment planning”. *Int J Radiat Oncol Biol Phys,* 40:197–205, 1998.

[89] E. H. W. Meijering, W. J. Niessen and M. A. Viergever. “Retrospective motion correction in digital subtraction angiography: a review”. *IEEE Trans Med Imaging,* vol. 18, pp. 2–21, 1999.

[90] A. W. Toga and P. M. Thompson. “The role of image registration in brain mapping”, *Image Vision Comput*., vol. 19, pp. 3–24 Special Issue, 2001.

[91] P. M. Thompson, R. P. Woods, M. S. Mega and A. W. Toga. “Mathematical/computational challenges in creating deformable and probabilistic atlases of the human brain”, *Human Brain Mapping*, 9: 81–92, 2000.

[92] P. A. van den Elsen, E. J. D. Pol, and M. A. Viergever. “Medical image matching – a review with classification”, *IEEE Engineering in medicine and biology*, 12(1), pp 26–39, 1993.

[93] C. R. Maurer and J. M. Fitzpatrick. “A review of medical image registration”. *In Maciunas, R. J. (ed.), Interactive image guided neurosurgery*, pp 17–44. American Association of neurological surgeons, Park ridge, IL, 1993.

[94] **T. Wang** and A. Basu. “Automatic Estimation of 3D Transformations using Skeletons for Object Alignment”, In *Proceedings of IEEE International Conference on Pattern Recognition*, Hong Kong, pp 51-54, 2006.

[95] J. Thirion. “Non-rigid matching using demons”, *Computer vision and pattern recognition*, pp 245–251, 1996.

[96] C. Davatzikos. “Nonlinear registration of brain images using deformable models”. *Mathematical methods in biomedical image analysis*, pp 94–103, 1996.

[97] J. Ashburner and K. J. Friston. “Nonlinear spatial normalization using basis functions”. *Human Brain Mapping*, vol. 7, pp 254–66, 1999.

[98] T. Peters, B. Davey, P. Munger, R. Comeau, A. Evans and A. Olivier. “Three-dimensional multimodal image-guidance for neurosurgery”, *IEEE Transactions on medical imaging*, 15(2), pp. 121–128.

[99] D. A. Simon, R. V. O’Toole, M. Blackwell, F. Morgan, A. M. DiGioia and T. Kanade. “Accuracy validation in image-guided orthopaedic surgery”, *Medical robotics and computer assisted surgery*, pp 185–192, 1995.

[100] M. Y. Wang, J. M. Fitzpatrick and C. R. Maurer, “Design of fiducials for accurate registration of CT and MR volume images”, *Medical imaging*, Vol. 2434, pp 96–108, 1995.

[101] M. Soltys, D. V. Beard, V. Carrasco, S. Mukherji and J. Rosenman, “FUSION: a tool for registration and visualization of multiple modality 3D medical data”, *Medical imaging: image processing*, Vol. 2434, pp 74–80, 1995.

[102] O. Peria, L. Chevalier, A. Francois-Joubert, J. Caravel, S. Dalsoglio, S. Lavallee and P. Cinquin. "Using a 3D position sensor for registration of SPECT and US images of the kidney", *Lecture notes in computer science*, Vol. 905, pp23–29, 1995.

[103] T. Lehmann, C. Goerke, W. Schmitt, A. Kaupp and R. Repges. "A rotation-extended cepstrum technique optimized by systematic analysis of various sets of X-ray images", *Medical Imaging: Image processing*, Vol. 2710, pp 390–401, 1996.

[104] L. Dong and A. L. Boyer. "A portal image alignment and patient setup verification procedure using moments and correlation techniques", *Physics in medicine and biology*, Vol 41, 697–723, 1996

[105] M. Singh, A. Basu and M. Mandal. "Event Dynamics based Temporal Registration", *IEEE Trans. on Multimedia*, vol.9(5), pp1004-1015, Aug. 2007.

[106] I. Cheng, S. Nilufar, A. Basu and R. Goebel, "Shape Tracking and Registration for 4D Visualization of MRI", *LNCS 4291*, pp 253-262 Springer-Verlag Berlin Heidelberg.

[107] M. Ferrant, A. Nabavi, B. Macq, P. M. Black, F. A. Jolesz and R. Kikinis R. "Serial registration of intraoperative MR images of the brain". *Med Image Anal,* 2002;6:337–359.

[108] J. P. W. Pluim, J. B. A. Maintz, and M. A. Viergever, "Mutual-Information-Based Registration of Medical Images: A Survey", IEEE transactions on medical imaging, VOL. 22, NO. 8, AUGUST 2003.

[109] **T. Wang** and A. Basu, "A note on 'A fully parallel 3D thinning algorithm and its applications' ", Vol. 28(4), pages 501-506, *Pattern Recognition Letters*, 2007.

[110] **T. Wang** and A. Basu, "Iterative Estimation of 3D Transformations for Object Alignment", *International Symposium on Visual Computing, LNCS 4291*, pages 212-221, 2006.

[111] **T. Wang** and A. Basu, "Automatic Estimation of 3D Transformations using Skeletons for Object Alignment", *IAPR/IEEE International Conference on Pattern Recognition*, pages 51-54, 2006.

Chapter 3 Unit-width Curve Skeletons

3.1 Introduction

Unit-width curve skeletons (*i.e.*, the skeletons are only one-voxel thick) are highly desirable in many applications such as mesh segmentation [24] and 3D object matching and retrieval [2].

Li *et al* [24] proposed a mesh segmentation method based on edge contraction and space sweeping. The sweeping path is a unit-width curve-skeleton. A component of a 3D mesh is defined to be the result of the sweeping of a *cross-section* along the sweeping path. Figure 3.1 shows the sweeping process.

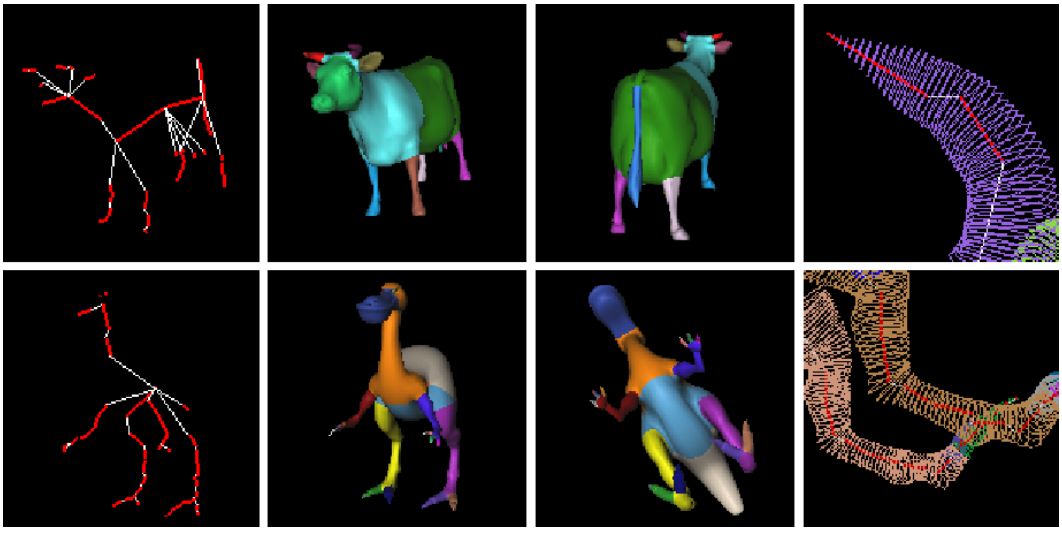


Figure 3.1: Mesh segmentation using unit-width curve skeletons [24].

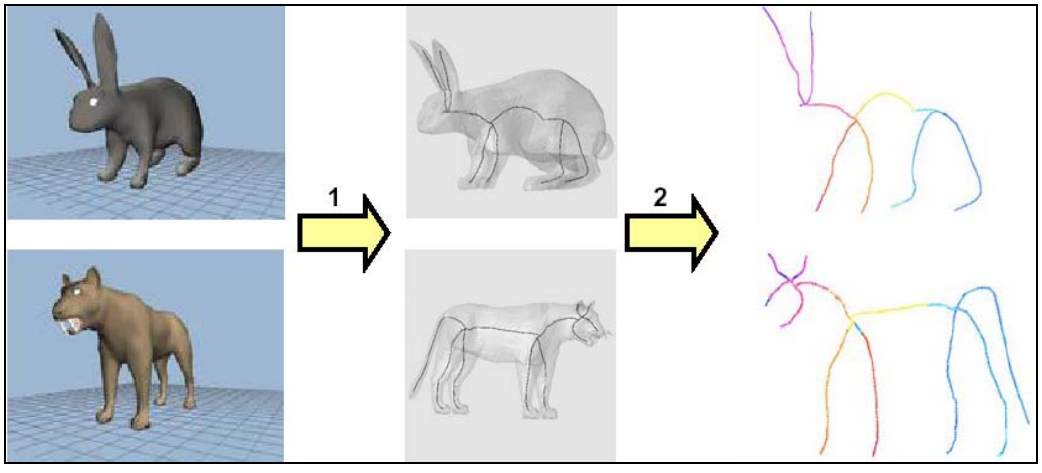


Figure 3.2: Matching and retrieval using unit-width curve skeletons [2].

Cornea presented a 3D object retrieval and matching system [2] using unit-width curve skeletons. Figure 3.2 shows an example of matching between two objects. In step 1, the unit-width curve skeleton for each object is computed while in step 2, the many-to-many matching establishes the distance and the correspondence between the two skeletal representations. The skeleton regions that were matched to each other are shown in the same color in Figure 3.2.

In addition to those applications mentioned above, there are many other applications of unit-width curve skeleton including animation, virtual reality, etc.

However, as we have seen in Chapter 2, some skeletons generated by skeletonization algorithms are not unit-width. This chapter presents a valence normalized spatial median (VNSM) algorithm, which eliminates crowded regions and generates unit-width curve skeletons. The proposed technique can also be used to refine skeletons generated from other 3D skeletonization algorithms to achieve unit-width. This work was published in [29].

The remainder of this chapter is organized as follows. In Section 3.2, some related algorithms are discussed. Then the proposed algorithm [29] is introduced in Section 3.3. Experimental results are presented in Section 3.4, before the work is concluded in Section 3.5.

3.2 Related works

Cornea proposed a potential field based algorithm to generate curve skeletons [2]. The idea is to extract some critical points in a force field to generate the skeleton. This algorithm has three steps. First is to compute the vector field on a 3D model. Second is to locate the critical points in the vector field, and finally the algorithm extracts the curve skeleton following a force directed approach. However, connectivity of the critical points is not guaranteed (see Figure 3.3).

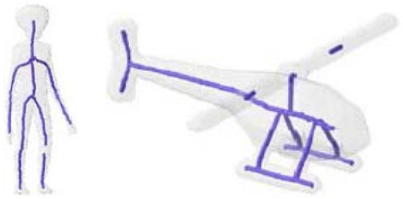


Figure 3.3: models with disconnected skeletons (images courtesy of Cornea [2]).

Voronoi diagram based algorithms assign the surface points on a 3D object as generating points and apply an incremental construction method to approximate the skeleton using Voronoi diagram computation [9-10, 25]. When the surface points are sufficiently dense, this technique can generate a satisfactory skeleton. However, it may fail when the surface points are sparse. Moreover, computing the Voronoi diagram is time-consuming.

Brunner *et al.* [16] use an iterative algorithm to merge junction knots, which generate the minimal cost, to create unit-width curve skeletons (Figure 3.4). However, this algorithm only works in non-equilateral 3D grid but not in 3D binary images (equilateral 3D grid) or 3D mesh (non-grid). Also, this algorithm is expensive because it needs to compute the different merging options. For example, if there are E junction knots and V edges between them, the complexity of this algorithm is $O(E \times V^3)$.

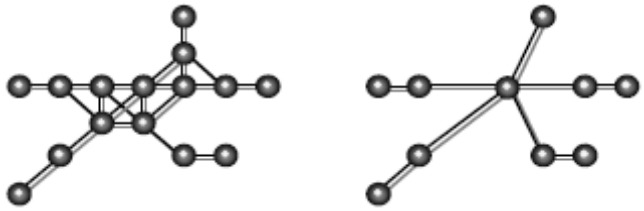


Figure 3.4: The left graph shows the junction knots in the curve skeleton. In the right graph, these junction knots are merged to a single junction knot to create a unit-width curve skeleton [16].

Sundar *et al*. [19] use a clustering method to reduce the number of points on a curve skeleton. A representative point is selected to replace a cluster of points that are within a distance of $D_{threshold}$. This algorithm has two drawbacks. First, different clusters need different $D_{threshold}$ values. How to determine different $D_{threshold}$ values for different clusters is non-trivial. Second, a cluster is not fully connected in most cases. Therefore, this algorithm may disconnect the skeleton by choosing a point that is not connected with other points.

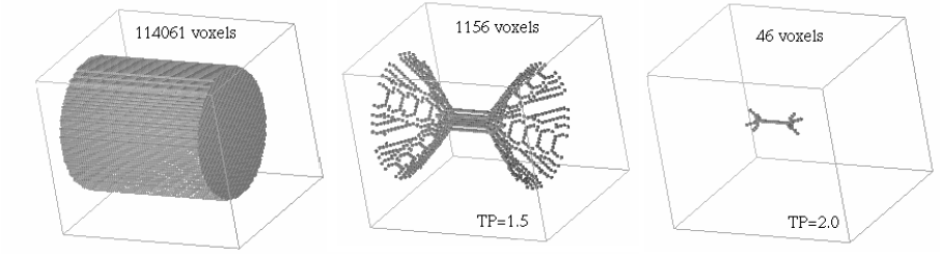


Figure 3.5: Sundar *et al*. [19], threshold and clustering.

Wang and Lee [28] presented a curve skeleton extraction algorithm. Their technique consists of three steps. First, it uses iterative least square optimization to shrink and simplify a 3D model. Then, it extracts the curve skeleton through a thinning algorithm. In the last step, a pruning approach is used to remove unnecessary branches based on shrinking ratios. However, this method requires too many free parameters. In addition, it often generates skeletons which deviate (shown in Figure 3.7) from the center of the model.

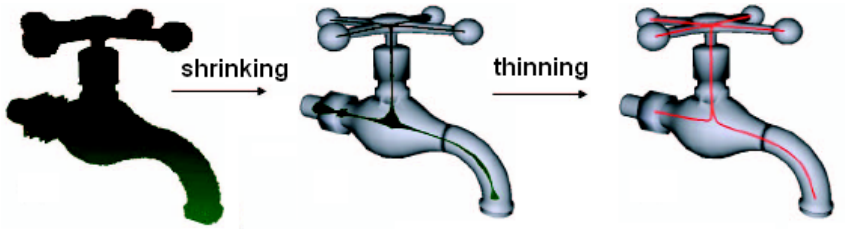


Figure 3.6: Wang and Lee [28], shrinking and thinning.

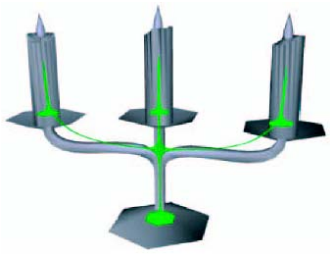


Figure 3.7: Skeleton deviates from the center of the model [28].

Svenssona *et al*. [26] propose an algorithm to extract unit-width curve skeletons from skeletons generated with 3D thinning. However, their method requires a really thin skeleton (*i.e.*, at most two-voxel thick) as input.

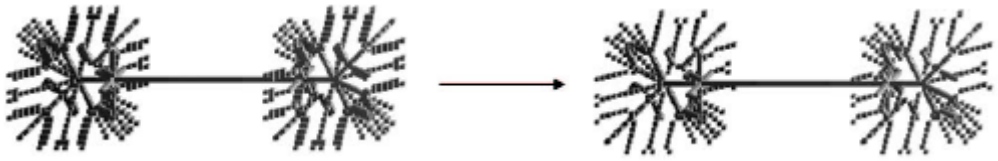


Figure 3.8: Svenssona *et al*. [26], simplifying.

Many 3D thinning techniques [3-5] can be used to generate curve skeletons. However, the skeletons generated are not guaranteed to be unit-width. In the next section, we will introduce our VNSM algorithm, which can generate a unit-width curve skeleton without requiring nearly thin skeleton and any threshold.

3.3 The proposed algorithm

In this section, we describe our algorithm for extracting unit-width curve skeletons. By comparison to other methods, our algorithm has five main characteristics. 1) It works on 3D binary images, and 3D meshes by performing voxelization as a pre-processing step. 2) It preserves connectivity. 3) The curve skeletons are unit-width. 4) It does not need control parameters, *e.g*. thresholds. 5) It is time efficient.

3.3.1 Definitions

Given an input 3D binary image (equilateral 3D grid)

$$B(x, y, z) = \{\, \delta(x, y, z) \,\}, \text{ where } \delta(x, y, z) = \begin{cases} 1 \ \textit{if } (x, y, z) \text{ is an object point} \\ 0 \ \text{otherwise} \end{cases},$$

we define a unit-width curve skeleton using the following definitions.

Definition 1: The ***26-neighborhood*** of an object point p is the 3x3x3 cube centered at p.

Definition 2: Let q be another object point. The Euclidean distance between p and q is defined as $dis = | p - q |$. Consider each edge in the cube be 1 unit. p and q are ***26-connected*** if $dis \leq \sqrt{3}$.

Definition 3: The ***valence*** of an object point p is denoted by $V(p)$, and is defined as the number of object points in p's 26-neighborhood, excluding p itself.

$$V(p) = (\sum_{i=-1}^{1}\sum_{j=-1}^{1}\sum_{k=-1}^{1}\delta(x+i, y+j, z+k)) - 1 \quad when\ B(p) = B(x, y, z) = 1 \qquad (3.1)$$

Figure 2.5 in Chapter 2 is used to demonstrate the geometric meaning of valence. The valence of point *p* in cube *a* is 1 because there is only one object (black) point in *p*'s 26-neighborhood excluding *p* itself. The valence of *p* in cube *b* is 2 because there are two object points in *p*'s 26-neighborhood excluding *p* itself. The valence of *p* in cube *c* is 3, and that of *p* in cube *d* is 1.

Definition 4: An object point *p* is called an ***end point*** if $V(p) = 1$ or $V(p) = 0$.

Definition 5: An object point *p* is called a ***middle point*** of p_1 and p_2 if

1) $V(p) = 2$ (p_1 and p_2 are the two neighbors of *p*);

2) p_1 and p_2 are not 26-connected in *p*'s 26-neighborhood.

Definition 6: An object point *p* is called a ***joint point*** of $p_1, p_2, \ldots, p_n$ if

1) $V(p) = n$ ($n > 2$, and $p_1, p_2, ..., p_n$ *are the n neighors of* p);

2) $p_1, p_2, \ldots, p_n$ are not 26-connected in *p*'s 26-neighborhood;

3) $p_1, p_2, \ldots, p_n$ are either end points or middle points.

Definition 7: An object point *p* is called a ***crowded point*** if it is neither an end point, a middle point, nor a joint point.

Definition 8: A ***crowded region*** is a set of 26-connected crowded points.

Definition 9: An object point is called an ***exit*** if it is

1) an end point or middle point;

2) 26-connected to one object point of a crowded region.

Definition 10: A skeleton is called a ***unit-width curve skeleton*** if it has no crowded regions.

3.3.2 Valence computation

The VNSM algorithm first computes the valence of each object point in the curve skeleton by counting the object points in the 26-connected neighborhood.

3.3.3 Crowded regions and exits

The next step is to mark the end points, middle points, joint points, and crowded points. Adjacent crowded points are consolidated into crowded regions. In each crowded region, exits are located and marked.

3.3.4 Valence Normalized Spatial Median (VNSM) algorithm

In this step, the center of each crowded region is computed. Once the center is obtained, we can connect it with all the exits in that region, and remove the object points that are not on the path between an exit and the center. The mostly often used definition of center is arithmetic mean. For a given crowded region R with n object points $\{P_1, P_2, \ldots, P_n\}$, the arithmetic mean is defined as:

$$\sum_{i=1}^{n} P_i / n \tag{3.2}$$

However, the center defined by arithmetic mean may be outside a given region. Figure 3.8 (Left) shows the center is out of the given crowded region. In this figure, white points are object points and black points form background. The red points are the centers in different definitions.

In [27], some other frequently used definitions of center, such as Tukey median, Liu median, Oja median, depth-based trimmed mean, coordinate median, and spatial median, are discussed and compared. According to [27], spatial median stands as the best overall. The spatial median is defined as:

$$\arg\min_{p}\left(\sum_{i=1}^{n} | p - P_i | / n\right) \tag{3.3}$$

where $|\cdot|$ is the Euclidean distance. Spatial median works very well for convex regions. However, in the context of this chapter, crowded regions may not be convex therefore spatial median may fail. Figure 3.8 (Middle) demonstrates that the center defined by spatial median is on the boundary of the crowded region.

We propose a Valence Normalized Spatial Median (VNSM) algorithm to compute the center of a crowded region. The center is given by:

$$\arg\min_{p}\left(\sum_{i=1}^{n}\frac{1}{V(p)}\mid p-P_i\mid/n\right) \tag{3.4}$$

where *V(p)* is the valence of the point *p*. $\frac{1}{V(p)}$ assigns penalties to boundary points which have smaller valences than inside points. As a result, the derived center is attracted towards the middle of the region.

In our implementation, we modify Equation (3.4) to Equation (3.5) to eliminate the computational cost in performing the square root and division operations.

$$\arg\min_{p}\left(\sum_{i=1}^{n}\frac{1}{V(p)}\mid p-P_i\mid^2\right) \tag{3.5}$$

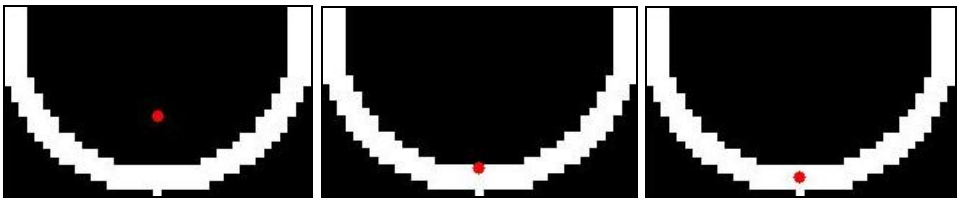


Figure 3.9: The red (gray in B&W) point denotes the "center" of a crowded region. From left to right, the locations of center defined by arithmetic mean, spatial median and VNSM are shown respectively.

Since *n* is a constant, removing *n* does not affect the value for which the expression attains its minimum. Figure 3.9 (Right) shows the center obtained by using VNSM.

3.3.5 Unit-width curve skeleton

In the last step, we apply the Dijkstra shortest path algorithm [17-18] to connect the exits with the center computed in the previous step. We remove other object points in the crowded region that are not in the paths connecting the exits and the center. The outcome is a unit-width curve skeleton.

The pseudo code of our algorithm is as follows:

Input: non-unit-width skeleton *I*

Output: unit-width curve skeleton *O*

Algorithm Generating_Unit_Width_Curve_Skeleton (*I*)

Initialization: Initialize output *O* and copy *I* to *O*.

Valence compution: Calculate the valence of each object point on the skeleton *O*.

Points classification: Mark end points, middle points, joint points, and crowded points. If there is no crowded point, output *O* and finish.

Crowded region location: Organize crowded points into crowded regions.

In each crowded region

Begin

Exit location: Find all exits.

Center determine: Determine the center point using the VNSM algorithm.

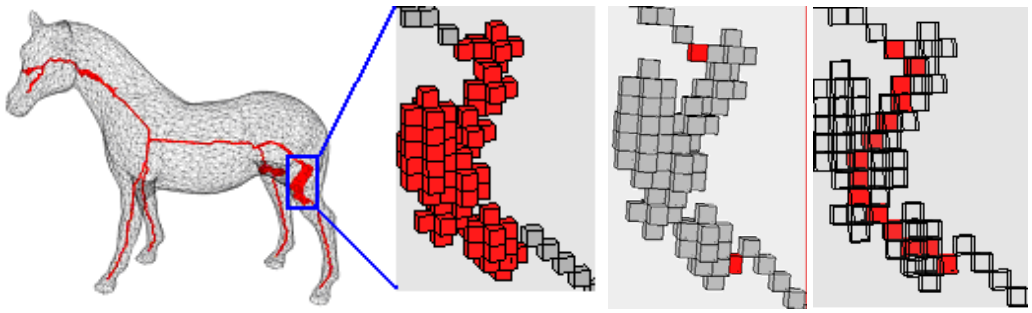
Shortest path computation: Apply the Dijkstra shortest path algorithm to find the shortest path between the center and each exit. Remove the object points that are not on the shortest paths.

End

Output: Output the unit-width curve skeleton *O*.

End algorithm

If there are *E* crowded points and *V* edges between them, the complexity of our method is $O(E+V^2)$ [17-18] based on the Dijkstra shortest path algorithm. Our VNSM algorithm inherits the characteristics of the Dijkstra algorithm: unique, connected and has no circle. Our technique also guarantees the generated skeleton to be unit-width. Figure 3.10 shows an example of unit-width curve skeleton generation.



(a) (b) (c) (d)

Figure 3.10: (a) Non-unit-width curve skeleton (b) a crowded region (c) two exits of the crowded region and (d) the constructed shortest path.

3.4 Experimental results

The 3D models used in this work were downloaded from the Mesh Compendium [20] and the Princeton Shape Benchmark [21], to validate the effectiveness of our algorithm. Surface meshes were converted to 3D binary images with binvox [22-23]. The input skeletons (non-unit-width) were extracted by a 3D thinning algorithm [3, 15]. Some examples of non-unit width skeletons are shown in Figure 3.11 and some examples of unit-width curve skeletons generated with our algorithm are shown in Figure 3.12.

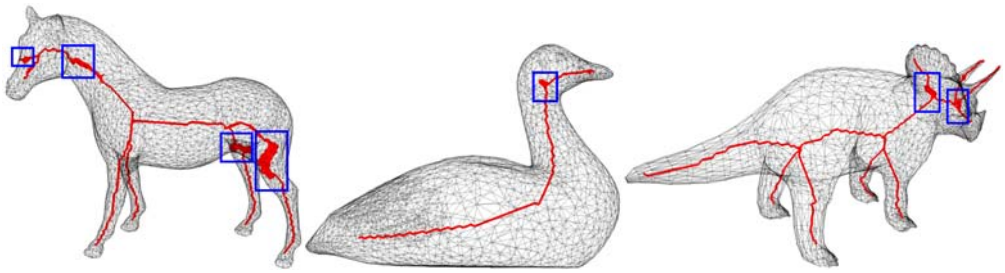


Figure 3.11: Examples of crowded regions

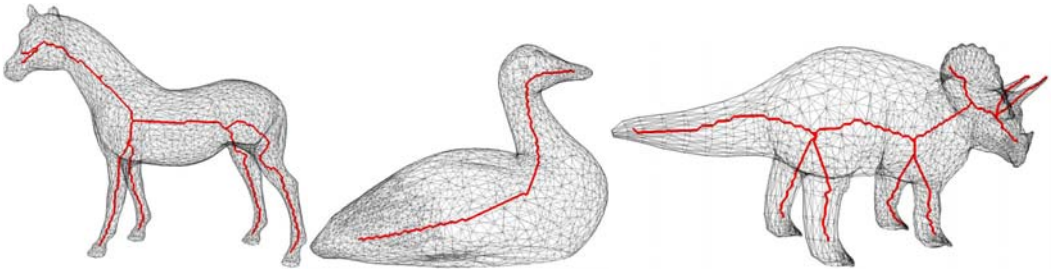
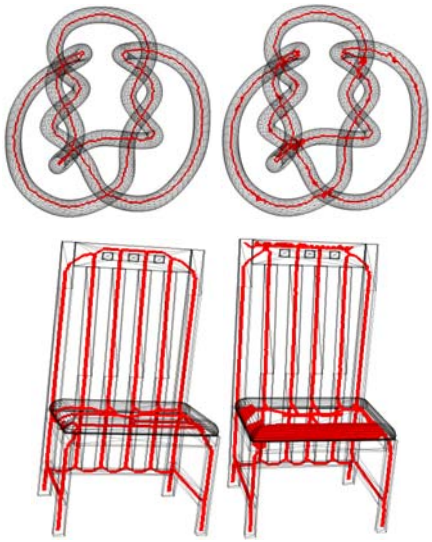


Figure 3.12: Examples of unit-width curve skeletons generated with our VNSM algorithm.

A thinness comparison of our approach for computing skeletons relative to another well known approach - Ma and Sonka's approach [3, 15] that creates connected skeletons is shown in Figure 3.13.

Until recently there have been few attempts to develop a metric to give a measure of the thinness of curve-skeleton. Cornea [2] defined the *thinness index*

of a skeleton as $TH = \frac{N}{NS}$, where N is the number of curve segments with no regular voxels [2] and have at least one junction voxel as one of their end-points, and NS is the total number of segments of the curve-skeleton. This thinness index can indicate the thinness of skeleton. However, the complexity of calculation of N and NS is $O(V_e^2)$, where V_e is the number of end points. In this section, we present a new *Thinness Metric* $TM = \frac{NC}{SUM} \in [0,1]$, where NC is the number of crowded points and SUM is the number of all points on the skeleton. Since our approach classified all points on the skeleton into end points, middle point, joint points and crowded points, the *Thinness Metric* can be obtained by only one division operation. The *Thinness Metric* is a normalized metric with values between zero and one. If TM is zero, the skeleton is unit-width thin; otherwise, it is not thin. If TM is one, all the points on the skeleton are crowded points. Table 3.1 shows the comparison between our method and Ma and Sonka's approach [19, 30]. Since the crowded points are removed from the skeletons, the score of our method is always zero, which indicates the curve skeletons generated by our method are always thin.



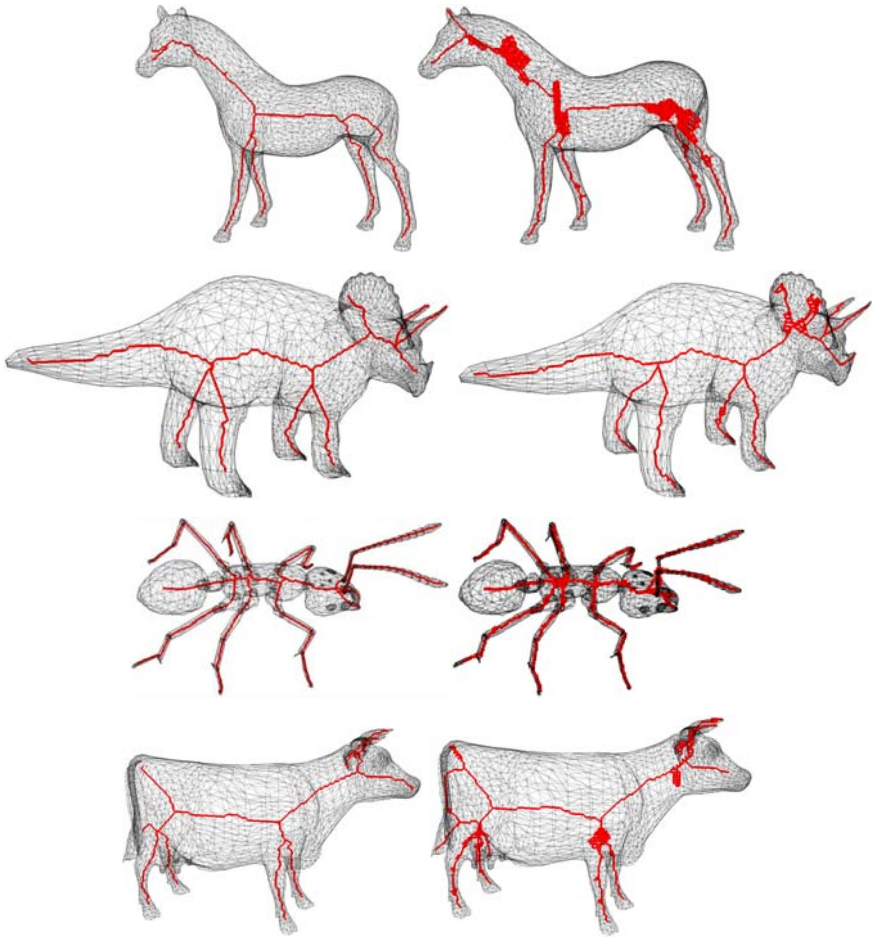


Figure 3.13: Comparing results of our algorithm (left column) with skeletons generated by Ma and Sonka [3, 15] (right column). Note that the skeletons generated by our algorithm are unit-width, while the skeletons on the right contain crowded regions.

Model	Proposed method	Ma and Sonka
Knot	0.00	0.54
Chair	0.00	0.53
Horse	0.00	0.73
Triceratops	0.00	0.43
Ant	0.00	0.42
Cow	0.00	0.64

Table 3.1: Thinness comparison of the models in Figure 3.16. Smaller values indicate better thinness.

3.5 Conclusions and Discussions

In this chapter, we present a Valence Normalized Spatial Median (VNSM) algorithm to generate unit-width curve skeletons from non-unit-width skeletons. Locating the center of a crowded region accurately is essential for generating a compact skeleton representation of a 3D model. Although the spatial median location estimator out-performs other frequently used locators, it does not work well on non-convex regions. We propose the Valence Normalized Spatial Median (VNSM) algorithm to compute the center of a crowded region, and apply the Dijkstra shortest path algorithm to generate a unit-width curve to replace the crowded region. This algorithm can be used in conjunction with another skeletonization algorithm, to ensure that the output skeleton is unit-width. We have already used this algorithm to generate unit-width curve skeletons, for constructing topology graphs and chain codes [30], and for matching [31] of 3D objects.

Bibliography

[1] H. Blum. “A transformation for extracting new descriptors of shape”, *Models for the Perception of Speech and Visual Form*, pp. 362–380, MIT Press, Cambridge, MA, USA, 1967.

[2] N. D. Cornea. “Curve-Skeletons: Properties, Computation And Applications”, *Ph.D. Thesis, The State University of New Jersey*, May 2007.

[3] C. M. Ma, M. Sonka. “A fully parallel 3D thinning algorithm and its applications”, *Computer Vision and Image Understanding*, 64 (3): pp 420-433, 1996.

[4] K. Palagyi and A. Kuba. “A 3D 6-subiteration thinning algorithm for extracting medial lines”, *Pattern Recognition Letters*, vol. 19 (7): pp 613-627, 1998.

[5] C. Lohoua and G. Bertrand. “A 3D 6-subiteration curve thinning algorithm based on P-simple points”, *Discrete Applied Mathematics*, Vol. 151, pp 198–228, 2005.

[6] C. Pudney. “Distance-Ordered Homotopic Thinning: A Skeletonization Algorithm for 3D Digital Images”, *Computer Vision and Image Understanding*, vol. 72(3):404-413, 1998.

[7] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, Academic Press, New York, 1982.

[8] C. Arcelli and G. S. di Baja, “A width independent fast thinning algorithm”, *IEEE Trans. Pattern Anal. Mach. Intell.* Vol. 7, pp. 463–474, 1985.

[9] R. L. Ogniewicz and M. Ilg. “Voronoi Skeletons Theory and Applications”, *CVPR*, 1992.

[10] R. L. Ogniewicz and O. Kubler. “Hierarchic Voronoi Skeletons”, *Pat. Rec.*, p. 343-359, 1995.

[11] E. C. Sherbrooke, N. M. Patrikalakis and E. Brisson. “An algorithm for the medial axis transform of 3d polyhedral solids”, *IEEE T. VCG*, vol. 2 (1), pp. 44-61, 1996.

[12] P. Giblin and B. B. Kimia. “A formal classification of 3D medial axis points and their local geometry”, *CVPR*, 2000.

[13] F. F. Leymarie and B. B. Kimia. “The Shock Scaffold for Representing 3D Shape”, *LNCS 2059*, pp 216-229, 2001.

[14] F. F. Leymarie and B. B. Kimia. “Computation of the Shock Scaffold for Unorganized Point Clouds in 3D”, *CVPR*, 2003.

[15] **T. Wang** and A. Basu. “A note on ‘A fully parallel 3D thinning algorithm and its applications’ ”, *Pattern Recognition Letters*, vol. 28(4): 501-506, 2007.

[16] D. Brunner, G. Brunnett. “An extended concept of voxel neighborhoods for correct thinning in mesh segmentation”, *Spring Conference on Computer Graphics*, pp.119-125, 2005.

[17] E. W. Dijkstra, “A note on two problems in connexion with graphs”. *In Numerische Mathematik*, pp. 269–271, 1959.

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 24.3: Dijkstra's algorithm, pp.595–601.

[19] H. Sundar, D. Silver, N. Gagvani, S. Dickinson, Skeleton "Based Shape Matching and Retrieval", *Shape Modeling International*, pp. 130-142, 2003.

[20] http://www.cs.caltech.edu/~njlitke/meshes/toc.html (retrieved in April 2008)

[21] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, *The Princeton Shape Benchmark, Shape Modeling International*, Genova, Italy, June 2004.

[22] F. Nooruddin and G. Turk, "Simplification and Repair of Polygonal Models Using Volumetric Techniques", *IEEE Trans. on VCG*, vol. 9(2), pp. 191-205, 2003.

[23] http://www.cs.princeton.edu/~min/binvox/ (retrieved in April 2008)

[24] X. Li, T.W. Woon, T.S. Tan, Z. Huang. "Decomposing polygon meshes for interactive applications", *ACM Symposium on Interactive 3D Graphics*, 35-42, 2001.

[25] D. Attali and A. Montanvert, "Computing and Simplifying 2D and 3D Continuous Skeletons", *Computer Vision And Image Understanding*, Vol. 67, No. 3, pp. 261–273, 1997.

[26] S. Svenssona and G. S. di Bajab, "Simplifying curve skeletons in volume images", *Computer Vision and Image Understanding*, vol. 90, pp. 242–257, 2003.

[27] J. C. Masse and J. F Plante, "A Monte Carlo study of the accuracy and robustness of ten bivariate location estimators", *Comput. Statistics & Data Analysis*, vol. 42, pp. 1-26, 2003.

[28] Y. S. Wang and T. Y. Lee, "Curve-Skeleton Extraction Using Iterative Least Squares Optimization", *IEEE T. VCG*, VOL. 14, NO. 4, pp. 926-936, 2008.

[29] **T. Wang** and I. Cheng, "Generation of Unit-width curve skeletons based on Valence Driven Spatial Median (VDSM)", *International Symposium on Visual Computing, LNCS 5358*, pages 1061-1070, 2008.

[30] V. Lopez, I. Cheng, E. Bribiesca, **T. Wang** and A. Basu, "Twist-and-Stretch: A Shape Dissimilarity Measure based on 3D Chain Codes", *ACM SIGGRAPH Asia Research Poster*, 2008.

[31] **T. Wang**, I. Cheng, V. Lopez, E. Bribiesca and A. Basu, "Valence Normalized Spatial Median for Skeletonization and Matching", *Search in 3D and Video workshop (S3DV), in conjunction with IEEE International Conference on Computer Vision (ICCV)*, 2009.

[32] E. Bribiesca, "A method for representing 3D objects using chain coding", *J. Vis. Commun.*, 2008.

Chapter 4 Flexible Vector Flow and Applications in 2D Brain Tumor Segmentation

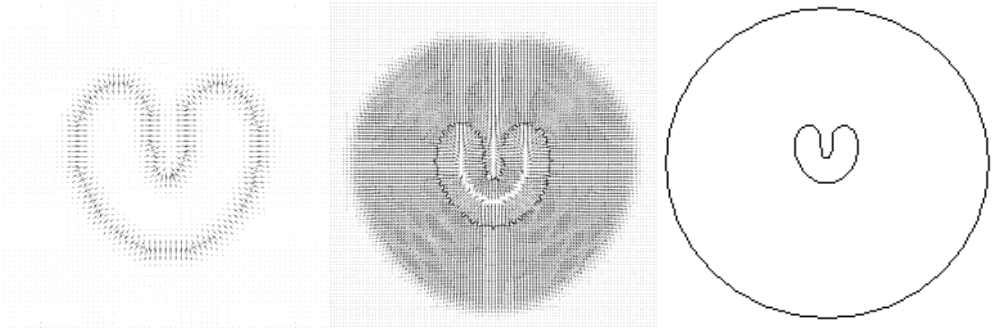
4.1 Introduction

The segmentation study presented in this chapter is a semi-automatic approach called Flexible Vector Flow (FVF) active contour model. This study was published in [15].

The goal of segmentation is to locate the target object, *i.e.*, the Region Of Interest (ROI) in 2D or the Volume Of Interest (VOI) in 3D. Active contour models or snakes [1-5] have been adopted as effective tools for segmentation [6-7]. Active contour based segmentation algorithms have many applications such as medical image processing and analysis. Our application is brain tumor segmentation in Magnetic Resonance (MR) images. Active contour models discussed in the literature can be classified into two categories: parametric [1-3] and level set [4-5].

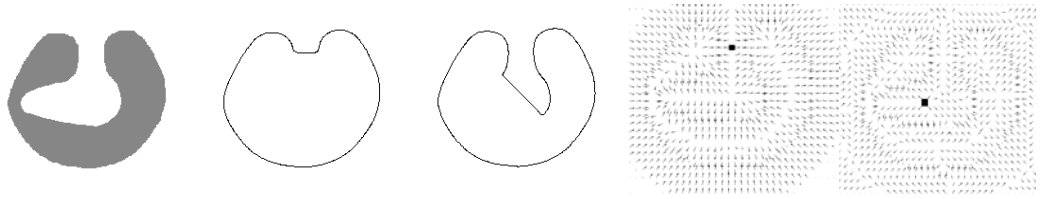
Parametric active contour models are represented explicitly as polynomials or splines. Given an initial contour, the evolution of a parametric active contour model is driven by external forces while the shape of the contour is maintained by the internal forces [1]. Due to the availability of efficient numeric methods [1-2], parametric active contour models are often faster than level set ones [4-5]. Given a single initial contour as the input, parametric active contour models are able to extract a single object. Despite the above strength, parametric snakes have two weaknesses. First, the capture range is limited. Capture range is the region that the external forces are strong enough to drive contour evolution. The external forces of the traditional [1] and Gradient Vector Flow (GVF) [2] parametric snakes are represented as small arrows in Figure 4.1 (a) and (b). The length of an arrow represents the magnitude of an external force at that location. In Figure 4.1, the capture range is the region with dense arrows (external forces) that are strong enough to drive the contour evolution. We can see that the capture range of the traditional snake is a very limited region around the object boundary. GVF

diffuses the external forces from the object boundary to its surroundings to obtain a larger capture range. However, the capture range of GVF is still not the entire image. If the initialization is out of the capture range, the active contour will not evolve (Figure 4.1 (c)). Second, some parametric snakes, *e.g.*, traditional, GVF and Boundary Vector Flow (BVF) [3], are unable to extract acute concave shapes (Figure 4.2 (b) and (c)) because their external force fields are static. There could be some saddle points or stationary points [4] where the composition of external forces is zero (Figure 4.2 (d) and (e)) in static force fields. Therefore, the contours will get stuck at those locations and equilibrium will be achieved too early [4].



(a) (b) (c)

Figure 4.1: The limited capture range of (a) a traditional parametric snake and (b) a GVF parametric snake. If the initialization (outer circle in (c)) is outside the capture range, convergence does not occur.

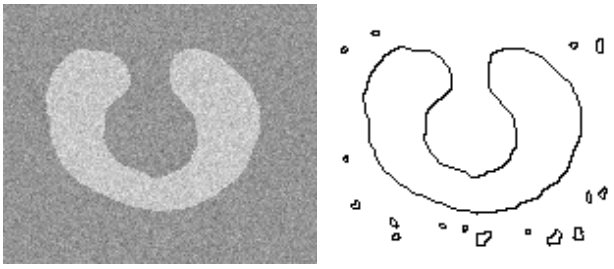


(a) (b) (c) (d) (e)

Figure 4.2: (a) An acute concave shape. (b) GVF and (c) BVF are not able to capture the acute concave shape. A saddle point in GVF is shown in (d) and a stationary point in BVF is shown in (e).

Level set active contour models [4-5] are implicitly represented in the zero level set. The evolution of a level set active contour model is achieved by deforming the level set function. The advantages of level set include the abilities

to capture multiple objects and complex geometries [4]. However, they are usually slower than parametric methods because the deformation of a higher dimensional function is required [4]. Morse *et al.* [12] proposed to implicitly represent snakes using radial basis functions by placing them at some landmarks [12]. This can avoid manipulating a higher dimensional function but it requires insertion and deletion of landmarks dynamically. Moreover, "false" objects can be extracted in the presence of noise which may cause multiple zero level sets (Figure 4.3).



(a) (b)

Figure 4.3: (a) A "U-shape" object in noisy environment (b) false objects (*i.e.*, small enclosed contours) can be extracted by a level set snake.

In this study, we focus on parametric snakes taking advantage of time efficiency. To address the issues of capture range and acute concave shape, we propose a parametric active contour model using a new concept, that we call "Flexible Vector Flow" (FVF), which emphasizes that the external force fields are flexible and dynamically updated to drive the contour evolution. FVF has the largest capture range, *i.e.*, the entire image. FVF is also able to extract acute concave shapes due to its non-static external force fields. In this model, the external force field changes dynamically with the contour evolution. Thus, the FVF contour does not get stuck and acute concavities can be extracted.

The limitations with previous models that FVF will overcome are as follows:

- Limited capture range
- Inability to handle acute concave shapes

4.2 Background

4.2.1 Traditional snake

A traditional snake [1] is a parametric active contour:

$$c(s) = (x(s), y(s)), s \in [0,1] \tag{4.1}$$

Given an initial contour, it evolves within an image $I(x, y)$ to minimize the energy function:

$$E_{snake} = \int_0^1 [E_i(c(s)) + E_e(c(s))]ds \tag{4.2}$$

where E_i is the internal (spline) energy and E_e is the external energy.

The internal energy is given by:

$$E_i = \frac{\alpha(s)|c'(s)|^2 + \beta(s)|c''(s)|^2}{2} \tag{4.3}$$

In many implementations, the coefficient of the first-order term in (4.3) is a constant, $\alpha(s) = \alpha$; and $\beta(s)$ is set to zero to allow the snake to be second-order discontinuous and contain corners. Many parametric snakes share the same internal energy. They differ mostly in the external energy. A snake should evolve to minimize the energy functional E_{snake}. This problem can be formulated with the Euler-Lagrange equation. In calculus of variations, the Euler-Lagrange equation of:

$$J[c(s)] = \int_{s_0}^{s_1} F(s, c(s), c'(s), c''(s))ds \tag{4.4}$$

is represented by:

$$F_c - \frac{d}{ds}F_{c'} + \frac{d^2}{ds^2}F_{c''} = 0 \tag{4.5}$$

Therefore, the Euler-Lagrange equation of (4.2) is represented by:

$$\alpha c''(s) - \beta c''''(s) + \nabla E_e = \alpha\frac{d^2c}{ds^2} - \beta\frac{d^4c}{ds^4} + \nabla E_e = 0 \tag{4.6}$$

To find a numeric solution of (4.6), the snake is treated as a function of time t as well as s:

$$\alpha \frac{\partial^2 c}{\partial s^2}\frac{\partial c}{\partial t} - \beta \frac{\partial^4 c}{\partial s^4}\frac{\partial c}{\partial t} + \nabla E_e = 0 \tag{4.7}$$

When the contour stabilizes, the time term vanishes and a solution is obtained.

4.2.2 GVF snake

GVF snake [2] has a larger capture range than the traditional snake. It diffuses the edge information from the object contour to its neighborhood. The external force of GVF snake differs from the traditional snake in that it cannot be written as the negative gradient of a potential function. In addition to this, the GVF snake is formulated directly from a force balance condition rather than a variational formulation. The gradient vector flow is defined as the vector field:

$$G_{gvf}(x,y) = (u(x,y), v(x,y)), x \in [x_0, x_1] \; and \; y \in [y_0, y_1] \tag{4.8}$$

that minimizes the energy function:

$$E_{gvf} = \int_{y_0}^{y_1}\int_{x_0}^{x_1} (k(u_x^2 + u_y^2 + v_x^2 + v_y^2) + \left|\nabla f\right|^2 \left|G_{gvf} - \nabla f\right|^2) dxdy \tag{4.9}$$

where k is a blending parameter, u_x, u_y, v_x, *and* v_y are the derivatives of the vector field, and ∇f is the gradient of the edge map. The GVF snake is computed by solving the following Euler-Lagrange equations:

$$k\nabla^2 u - (u - f_x)(f_x^2 + f_y^2) = 0 \tag{4.10}$$

$$k\nabla^2 v - (v - f_y)(f_x^2 + f_y^2) = 0 \tag{4.11}$$

4.2.3 BVF snake

BVF [3] extends the capture range further to the entire image based on interpolation. It applies a threshold to generate a binary boundary map of the input image. Then, four potential functions Ψ_x, Ψ_y, Ψ_{xy} and Ψ_{yx} are computed using line-by-line interpolations in the horizontal, vertical and two diagonal directions. The boundary vector flows are defined based on the gradients of the following potential functions: $\Phi_1 = (\nabla\Psi_x, \nabla\Psi_y)$ (4.12)

$$\Phi_2 = (\frac{\sqrt{2}}{2}(\nabla\Psi_{xy} + \nabla\Psi_{yx}), \frac{\sqrt{2}}{2}(\nabla\Psi_{xy} - \nabla\Psi_{yx})) \tag{4.13}$$

Equation (4.12) represents the horizontal and vertical interpolations of the gradient forces. Equation (4.13) represents the interpolations of the gradient forces in two diagonal directions.

The external force is defined as:

$$E_e(x, y) = \Phi(x, y) \tag{4.14}$$

Similar to GVF, BVF is unable to extract acute concavities.

4.2.4 Magnetostatic Active Contour (MAC) Model

The MAC snake [4] is a level set active contour model. The external force of the MAC is based on magnetostatics and hypothesized magnetic interactions between the active contours and object boundaries. It is able to capture complex geometries and multiple objects with a single initial contour. However, as stated in the introduction, it is slower than parametric methods and may detect multiple false objects in the presence of noise, which may cause multiple zero level sets to arise. MAC snake represents the active contour with an implicit model in which the contour consists of all points in:

$$c = \{x \mid \phi(x) = 0\}, \text{ where } \phi : R^2 \to R \tag{4.15}$$

MAC relates the motion of that contour to a PDE (Partial Differential Equation) on the contour:

$$\frac{\partial \phi}{\partial t} = -\nabla \phi \cdot v(t) \tag{4.16}$$

where $v(t)$ describes the velocity of the contour movement. For image segmentation:

$$\frac{\partial \phi}{\partial t} = \alpha s(x) \nabla \cdot \left(\frac{\nabla \phi}{|\nabla \phi|} \right) |\nabla \phi| - (1 - \alpha) F(x) \cdot \nabla \phi \tag{4.17}$$

where α is a real constant, $s(x)$ is the stopping function, and $F(x)$ is the magnetostatic force.

4.3 Proposed Flexible Vector Flow (FVF) Method

Given an input image *I(x, y)* and a closed parametric contour *c(s)* given in (4.1),

the objective is to evolve the contour to extract a target object *O(x, y)*, *i.e.*, the brain tumor. This method is executed in three stages: binary boundary map generation, vector flow initialization and flexible vector flow computation. Figure 4.4 shows the flowchart of this method, which starts with initializing the contour, then the binary boundary map is generated and vector flow is initiated, and flexible vector flow is computed and dynamically updated until the object contour is extracted.

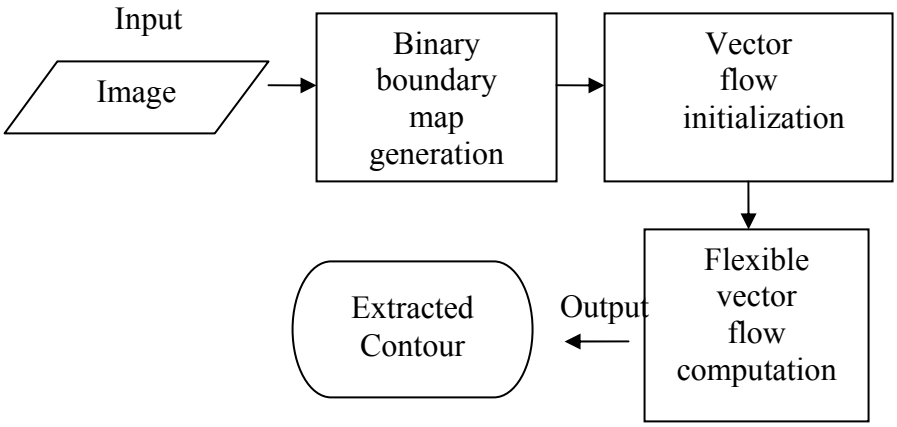


Figure 4.4: The process of FVF.

In the first stage, we apply a smoothing filter to the input image and apply a gradient operator to find edges in the image. A threshold (free parameter) $T \in [0, 1]$ is then used to generate the binary boundary map. At the second stage, the contour can be generated to initialize the external force field. The initial contour can be inside, outside, or overlapping the target. Our algorithm automatically detects the initialization and generates the external force field accordingly. The computation of the internal energy follows (4.3). The initial forces will push the active contour to the neighborhood of the target object. At the last stage, a control point is automatically selected from the object boundary and it generates new external force field to evolve the active contour. This point can flow flexibly along the object boundary, dynamically updating the external force field to avoid the problem of saddle points and stationary points [4], and therefore further evolve the active contour until convergence is achieved. The details follow.

4.3.1 Binary Boundary Map Generation

The boundary map is defined as:

$$M_B(x, y) = \left|\nabla(G_\sigma(x, y) * I(x, y))\right| \tag{4.18}$$

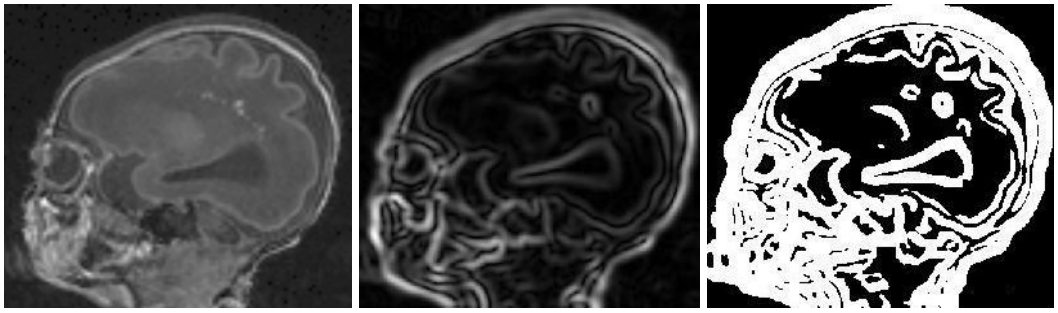
where $G_\sigma(x, y)$ is a Gaussian smoothing filter with standard deviation σ, $*$ is the convolution operator and ∇ is the gradient operator. We compute the normalized boundary map:

$$M_{NB}(x, y) = \frac{M_B(x, y) - \min(M_B(x, y))}{\max(M_B(x, y)) - \min(M_B(x, y))} \tag{4.19}$$

Similar to BVF [3], we apply a threshold $T \in [0, 1]$ to generate the binary boundary map:

$$M_{BB}(x, y) = \begin{cases} 1, & \text{if } M_{NB}(x, y) > T \\ 0, & \text{otherwise} \end{cases} \tag{4.20}$$

The choice of a suitable threshold value varies depending on the intensity distribution and contrast associated with the set of images being analyzed. For the brain MR images tested in our implementation, a default value of 0.1 works well. Observe that the blurred contour of the brain ventricle (low intensity region) is extracted successfully in the boundary map using $T = 0.1$ (Figure 4.5). We tested with a higher T value and then decreased T progressively but object boundary continuity was not obtained until 0.1 was reached. The extracted boundary provides an envelope to ensure that the final convergence is not out of bound. The threshold of 0.1 is consistent with the threshold of 0.13 commonly used by other snake models as suggested by Sum and Cheung in [3].



(a) (b) (c)

Figure 4.5: (a) A head MRI image, (b) its gradient map and (c) its extracted boundary map using a default threshold of 0.1.

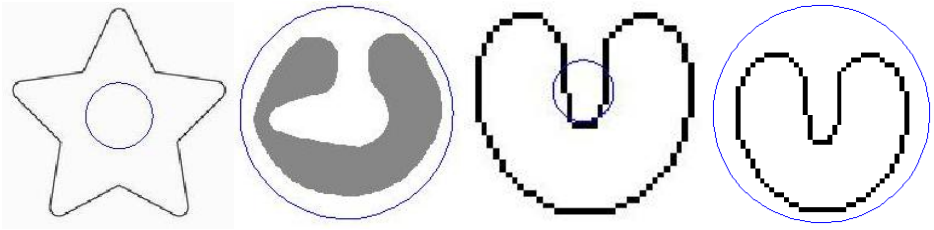
4.3.2 Vector Flow Initialization

At this stage, the contour should be generated to initialize the external force field. The initial parametric contour *c(s)* can be initialized either inside, outside, or overlapping (Figure 4.6 (a)-(c)) the target object *O(x, y)*. The FVF method is insensitive to the initialization by taking advantage of the binary boundary map generated at the previous stage. Suppose C is the initial contour, R_c is the region enclosed by contour C, and R_b is the region enclosed by the binary boundary map (Figure 4.7), we define $R_{bc} := R_b \cap R_c$. The following criteria determine the spatial relationships between the initial contour and the binary boundary map:

(a) C is inside the binary boundary map when $R_{bc} = R_c$;

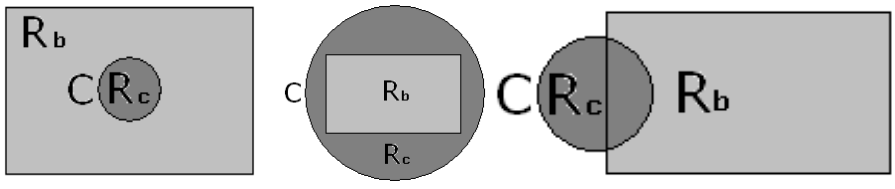
(b) C is outside the binary boundary map when $C \cap R_{bc} = \phi$;

(c) Otherwise, C is overlapping the binary boundary map.



(a) (b) (c) (d)

Figure 4.6: The initial contour (circle) is (a) inside (b) outside and (c) overlapping the target object.(d) the initial contour is automatically enlarged to enclose the object so that "overlapping" can be handled as "outside."



(a) (b) (c)

Figure 4.7: (a) Initial contour C is inside R_b, (b) contour C is outside R_b, and (c) contour C overlaps R_b. FVF is able to evolve in each of these initialization cases.

When overlap is detected, contour C will be automatically enlarged guided by the boundary map to enclose R_b. Therefore, "overlapping" is eventually handled as "outside" after enlargement (Figure 4.6 (d)). During enlargement, neighboring objects in the binary boundary map that do not contain R_{bc} remain outside the contour and only the one connected component (the target) in the binary boundary map that contains R_{bc} is included in R_c. A connected component is a region of 8-connected object pixels (ones) in the binary boundary map.

We use the discrete form of (4.1) to represent the contour C:

$$c(i) = \{(x_i, y_i)\}, i \in [0,1,..., P-1] \tag{4.21}$$

where P is the number of points on the contour.

The center point of the bounded region is located at:

$$(x_c, y_c) = (\sum_{i=0}^{P-1} x_i / P, \sum_{i=0}^{P-1} y_i / P) \tag{4.22}$$

An external energy function is defined as:

$$E_e(x,y) = \begin{cases} \chi(f_x + \delta\cos\phi, f_y + \delta\sin\phi) & when\ M_{BB}(x,y) = 0 \\ 0 & otherwise \end{cases} \tag{4.23}$$

where χ is a normalization operator, $\delta = \pm 1$ (controls the inward or outward direction, when the contour is "outside" or "inside"), $(f_x, f_y) = \chi(\nabla I(x,y))$, and:

$$\phi(x,y) = \begin{cases} \arctan(\dfrac{y-y_c}{x-x_c}) & when \quad x \neq x_c \\ \dfrac{\pi}{2} & when \quad x = x_c \ and \ y > y_c \\ \dfrac{3\pi}{2} & when \quad x = x_c \ and \ y < y_c \end{cases} \tag{4.24}$$

where $\phi(x,y) \in [0,2\pi]$.

The external energy E_e has a gradient component and a directional component. The gradient force is computed in a manner similar to the traditional snake [1] and GVF snake [2]. The characteristic of FVF lies in the computation of the

directional force, which is based on a polar transformation. When the contour is far away from the object, the directional force dominates and attracts the contour towards the object. When the contour is close to the object, the gradient force fits the contour to the object.

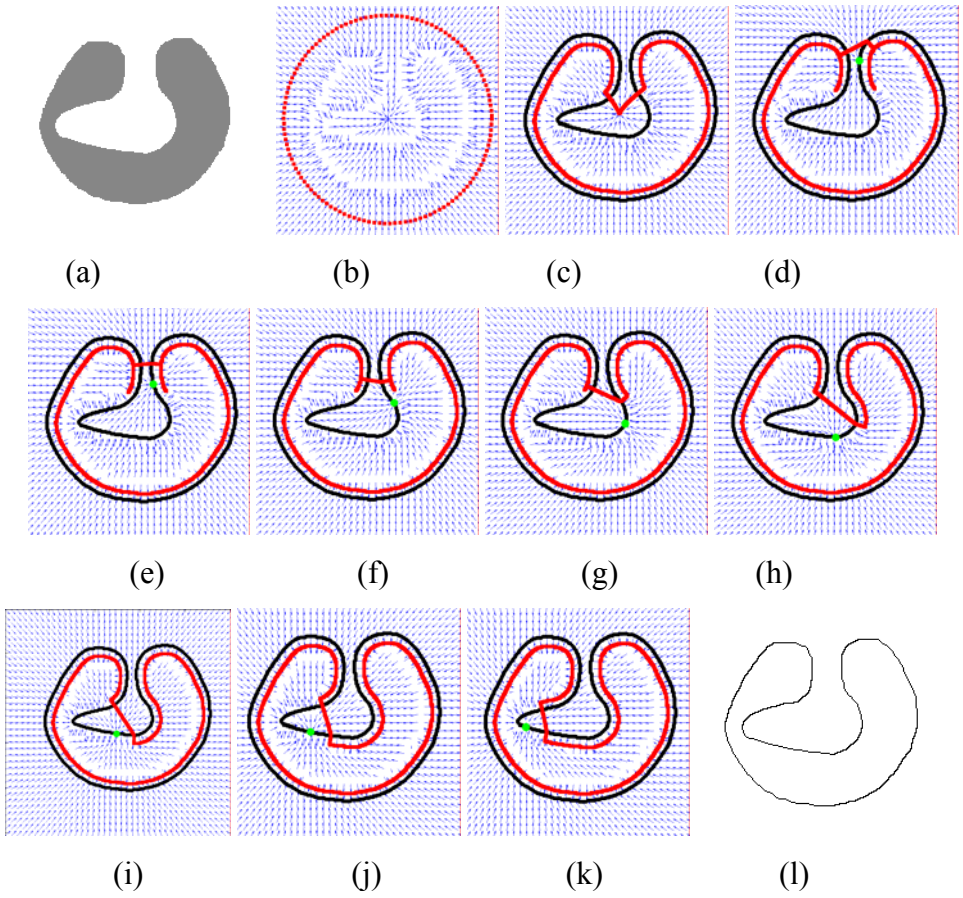


Figure 4.8: An example of FVF contour evolution: (a) The target object and (b) the initial contour and vector flow initialization, (c)-(k) a sequence of flexible vector flow processes and (l) the convergence result.

The capture range of FVF extends to the whole image because the vector flow energy defined in (4.23) spreads around the entire image *I(x, y)*. Even if the initial contour is far from the object, the snake can still evolve towards the object. In other words, the border of an image can be used as the initial contour when the initial contour is not given. This feature makes FVF more effective than either the traditional snake [1] or the GVF snake [2]. Although the capture range of BVF [3] can also extend to the entire image, the performance of FVF is more efficient

because interpolation is avoided. Furthermore, the BVF interpolation is executed in only four directions, whereas FVF is direction invariant ($\phi \in [0,2\pi]$). Note that the capture range of MAC is also the entire image. Figure 4.10 (p) – (t) illustrates the capture ranges of GVF, BVF, MAC and FVF. Figure 4.10 (p) shows an object without a given initial contour. The image border is then used as the initial contour. GVF failed to extract the object since the initial contour (image border) is out of its capture range. BVF, MAC and FVF can extract the object since their capture ranges cover the entire image domain.

Note that if concavities exist, convergence will not be achieved at the vector flow initialization step. Figure 4.8 (b) shows the vector flow initialization. The circle is the initial contour and the blue (dark gray in B&W print) small arrows represent vector flows. Figure 4.8 (c) shows that the evolution stops at the center of the image where the composition of external forces is zero. The contour will evolve to the red (middle grey in B&W print) line and is not able to extract the concave region. To extract the complete contour, the flexible vector flow computation step is performed.

4.3.3 Flexible Vector Flow Computation

In this step, a trace method is applied to the binary boundary to get a list of control points:

$$B(x_q, y_q) = \xi(M_{BB}(x, y)), q \in [0,1,...,Q-1] \tag{4.25}$$

where ξ is a boundary trace operator and Q is the number of the control points. In our implementation, we use the trace function in MATLAB.

The Flexible Vector Flow energy function is defined by:

$$E_{fvf}(x, y) = \begin{cases} \chi(\chi(\nabla \mathrm{I}(\mathrm{x}, \mathrm{y})) + \delta\chi(x - x_q, y - y_q)), if\ M_{BB}(x, y) = 0 \ \ \& \ \ (x_q, y_q) \in B \\ 0 \quad otherwise \end{cases} \tag{4.26}$$

The pseudo code of Flexible Vector Flow computation and active contour evolution is as follows:

0 Get a list of control points *B* with (4.25). *B* contains *Q* points.
1 *ind*=0;
2 While convergence is not achieved

```
3     q=ind+δ;
4     if q>=Q
5         break;
6     x=B[q].x;  y=B[q].y;
7     Generate new force field at (x, y) with (4.26);
8     Evolve the active contour in the new force field;
9 End while
10    Output the result of contour evolution.
```

In the pseudo code, *ind* (line 1) is the index of control point, q (line 3) is the new index of the control point, and (x, y) (line 6) is the new location of the control point.

Our intention is to use the control points to generate the external force fields. First, a list of control points B is computed with (4.25). Then, in each iteration of the above while loop, one control point is sequentially selected in the list B. Imagine that the control point is a moving point, this process looks like the point moving flexibly along the object boundary and generating vector flow (external force field) dynamically. This is the reason why we name the method Flexible Vector Flow.

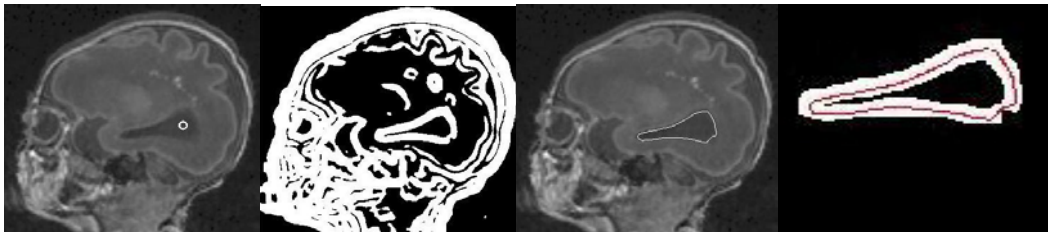
Using all the control points to generate force fields can be time-consuming. In addition to that, adjacent control points may generate external force fields with little differences. Therefore, a parameter δ is used to manage the selection of the control point. δ can be imagined as the velocity of the movement of the point. The method selects 1 out of δ control points to achieve better time efficiency. For instance, in Figure 4.8, we assign δ =20 to select one out of 20 control points. When δ=1, all points extracted by the trace operator are used one by one.

Once the control point moves to its new location (i.e., a new control point is selected), it generates new external force field to avoid the problem incurred by saddle points and stationary points, and therefore is able to further evolve the contour until convergence is achieved.

The energy E_{fvf} has a gradient term and a directional term. The directional force attracts the evolving contour towards the control points even for control points in a concave region. When the contour is close to the object, the gradient force fits the contour onto the object. Convergence is achieved when the contour stops evolving.

In Figure 4.8, the target object is given in (a), and (b) shows the initial contour and vector flow initialization. From (c) to (k), the flexible vector flows are shown as blue (dark grey in B&W print) arrows, and the evolving contour is illustrated in red (middle gray in B&W print). The control point involved in each convergence step is marked as a green (light gray in B&W print) dot. Note that the control points selected by the boundary trace operator are located along the binary boundary map.

In Figure 4.9, we show the results of applying our technique on a head MRI image to extract the low intensity brain ventricle region.



(a) (b) (c) (d)

Figure 4.9: Illustration of FVF process: (a) the target object (brain ventricle) with initial contour (small circle in the ventricle) added, (b) the binary boundary map, (c) the final contour of FVF in the image, and (d) a zoomed-in view of the binary boundary map which restricts the final contour inside an envelop.

As we can see from Figure 4.9, the control point can appear at any location on the boundary of the binary boundary map. It requires that initial contour must be inside the target object, or outside the target object, or overlapping the target object without overlapping other parts of the binary boundary map. Otherwise, the control point may appear at an undesirable location and drag the active contour to that location. In the next chapter, the initialization issue will be addressed.

4.4 Experimental results

We tested and compared FVF against GVF, BVF and MAC for three data sets: synthetic images, head MR images, and IBSR brain tumor MR images [13]. The head MR images were provided by the Department of Pediatrics at the University of British Columbia. The IBSR brain tumor MR images [13] were provided by the Center for Morphometric Analysis at Massachusetts General Hospital, and are

available at http://www.cma.mgh.harvard.edu/ibsr/. These T1-weighted images contain multiple scans of a patient with a tumor taken at roughly 6 month intervals over three and a half years. The images are in 256x256. The resolutions on these images are 0.9375x0.9375 mm in-plane by 3.1 mm slice thickness.

Segmentation ground-truth images come with the dataset. The ground-truth segmentations were obtained from the manual segmentations of human experts.

We also set up parameters for those snake models to compare them as fairly as possible. For GVF, we keep the default settings unchanged. For BVF, we test the input images with 9 different values (0.1, 0.2, …, 0.9) of the threshold T and report the best result. For MAC snakes, in addition to these 9 threshold values, we also tested it with 2 much smaller values (0.01 and 0.05) according to the suggestion of the authors [4]. Moreover, since dual snake contours (Contour 0 and Contour 1) are implemented in MAC, we report the contour that has better result. For FVF snakes, we use the threshold value determined by BVF.

For each test image, initial contour was placed inside, outside and overlapping the target to test the robustness and sensitivity of the methods to initializations.

4.4.1 Synthetic Images

We first tested and compared FVF with GVF, BVF and MAC snakes for a set of synthetic images. Some results are shown in Figure 4.10.

The 1st row (Figure 4.10 (a)-(e)) shows an acute concave object with an initial contour at the outside, and the result of GVF, BVF, MAC (Contour 1) and FVF. We can see that both MAC and FVF can extract the boundary of the object. However, GVF and BVF failed to do so. This is because both GVF and BVF are incapable of extracting acute concave shapes.

The 2nd row (Figure 4.10. (f)-(j)) shows an object with a small initial contour inside, and the results of GVF, BVF, MAC (Contour 0) and FVF. We can see that the GVF snake does not evolve at all because of the static equilibrium force field. BVF, MAC, and FVF can extract the boundary of this object.

The 3rd row (Figure 4.10. (k)-(o)) shows an object with an overlapping initial contour, and the result of GVF, BVF, MAC (Contour 0 and Contour 1) and FVF.

We can see that the GVF and BVF snakes did not extract the boundary of the object but instead evolved to a point. Each one of the dual contours of MAC snake superimposes on each other and outlines both the internal and external boundaries of the object. The FVF snake extracts the boundary of the object.

The 4th row (Figure 4.10. (p)-(t)) shows an object without a given initial contour (the image border is then used as the initial contour), and the results of GVF, BVF, MAC (Contour 1) and FVF. GVF failed to extract the object since the initial contour (image border) is out of its capture range. BVF, MAC and FVF can extract the object since their capture ranges cover the entire image domain.

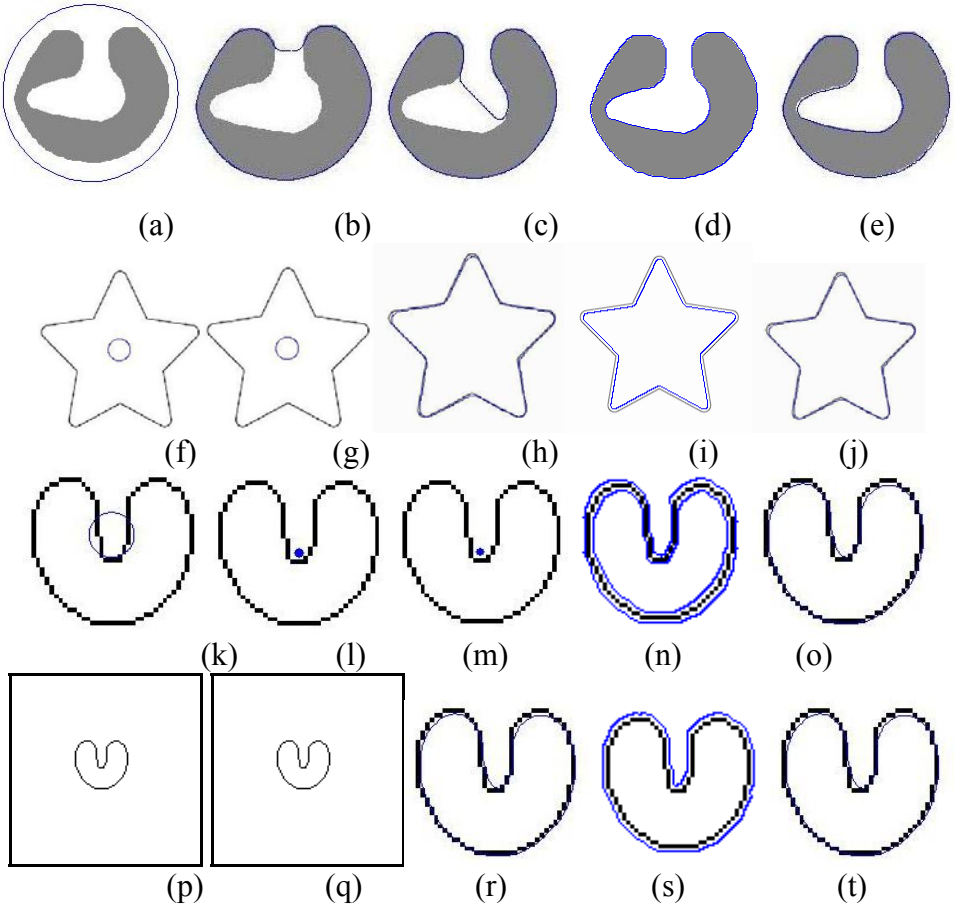


Figure 4.10: (a) An acute concave object with an initial contour at the outside, and the results of: (b) GVF, (c) BVF, (d) MAC (e) FVF; (f) an object with a small initial contour at the inside, and the results of: (g) GVF, (h) BVF, (i) MAC (j) FVF; (k) an object with an overlapping initial contour, and the results of (l) GVF, (m) BVF, (n) MAC (o) FVF; (p) an object with the image border as the initial contour, and the results of (q) GVF, (r) BVF, (s) MAC (t) FVF.

4.4.2 Head MR images

We then tested and compared FVF with GVF, BVF and MAC snakes for a set of head MR images. Some results are shown in Figure 4.11.

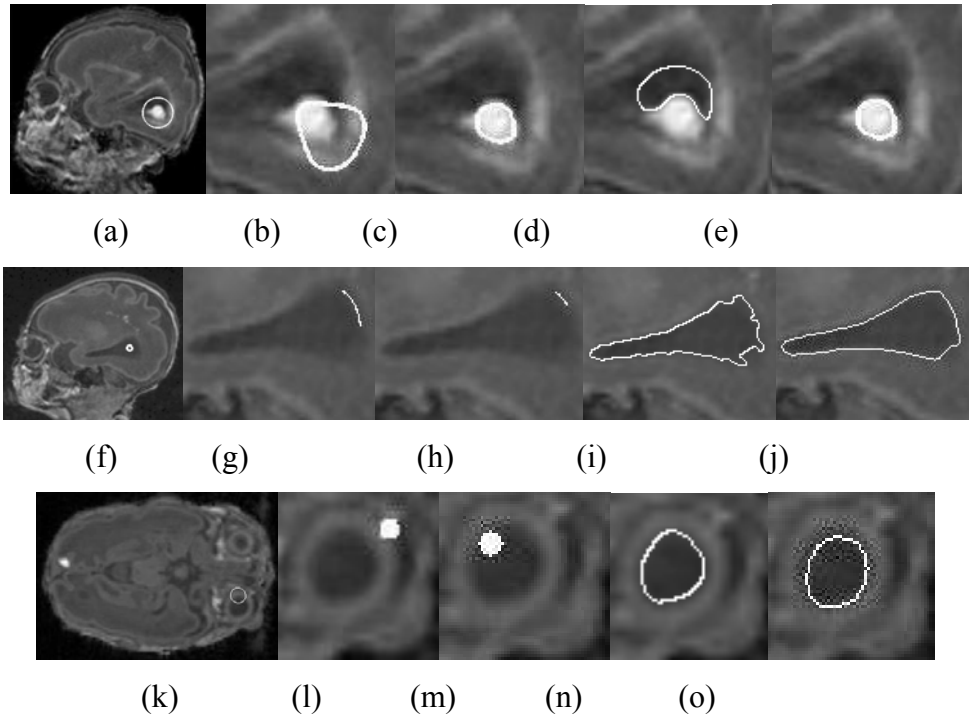


Figure 4.11: (a) An image with an initial contour on the outside of the high intensity region (intra-ventricular hemorrhage), and the results (zoomed-in) of: (b) GVF, (c) BVF, (d) MAC (e) FVF; (f) an image with a small initial contour at the inside of the brain ventricle, and the results (zoomed-in) of: (g) GVF, (h) BVF, (i) MAC (j) FVF; (k) an image with an initial contour overlapping the eye, and the results (zoomed-in) of (l) GVF, (m) BVF, (n) MAC (o) FVF.

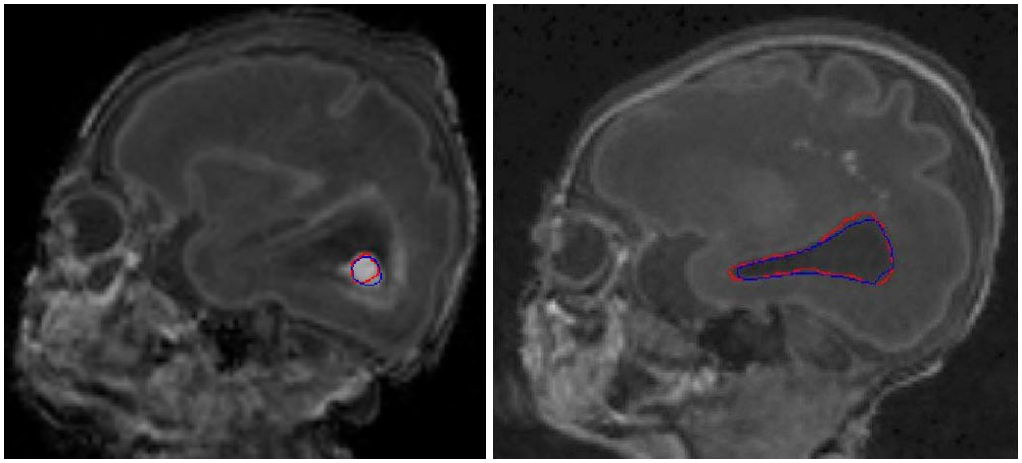
The first row (Figure 4.11 (a)-(e)) shows an image with an initial contour outside the high intensity region (intra-ventricular hemorrhage), and the result of GVF, BVF, MAC (Contour 1) and FVF. We can see that only BVF and FVF can extract the boundary of the object.

The second row (Figure 4.11. (f)-(j)) shows an image with a small initial contour inside the brain ventricle, and the result of GVF, BVF, MAC (Contour 1) and FVF. We can see that the GVF and BVF snakes evolved to lines on the right side of the brain ventricle but failed to extract the boundary of the ventricle. MAC

and FVF can both extract the boundary of the brain ventricle. However, the result of FVF is smoother.

The third row (Figure 4.11. (k)-(o)) shows an image with an initial contour overlapping the eye, and the result of GVF, BVF, MAC (Contour 1) and FVF. We can see that the GVF and BVF snakes degenerated to points on the top right and top left of the eye. MAC and FVF can both extract the eye and the results are similar.

Two examples of visual evaluation comparing the manually defined contour and the FVF generated contour are shown in Figure 4.12. Quantitative evaluations of GVF, BVF, MAC and FVF are reported in the next section.



(a) (b)

Figure 4.12: A visual inspection of the FVF generated contour: The images in (a) and (b) show the FVF detected contour (blue) overlaid with the ground truth (red).

4.4.3 IBSR Brain Tumor MR images and Quantitative Analysis

Brain tumor images (in which brain tumors are visible) from IBSR [13] are tested in our experiment. The Tanimoto Metric [11] is used for quantitative analysis.

Tanimoto Metric is defined as:

$TM = \frac{\|R_X \cap R_G\|}{\|R_X \cup R_G\|}, 0 \le TM \le 1$, where R_X is the region enclosed by the contour generated by the test method, R_G is the region enclosed by the ground-truth

contour, and $\|\cdot\|$ is set cardinality (number of elements). $TM = 0$ would indicate two completely distinct contours; while $TM = 1$ would indicate completely identical contours. Table 4.1 shows the test results.

Table 4.1 has 5 columns. The 1st column is the image ID (1 to 10). The 2nd column lists the name of the 4 methods. The 3rd to 5th columns are the Tanimoto Metric when the initial contour is inside, outside and overlapping the target object. The best method for each image is bold. FVF outperforms GVF, BVF and MAC in general.

It is important to note that the four methods apply different computational models to generate the force fields which are governed by the underlying image properties. Force fields are unevenly distributed in an image. In other words, a method may perform well in one region but may not do well in another region of the same image. An example is illustrated in Figure 4.13, which shows the results of GVF, BVF, MAC and FVF on test image #4 when the initial contour (not shown) is inside (2nd and 3rd rows) and outside (4th and 5th rows) the brain tumor respectively. Observe that when the initial contour (not shown) is inside the tumor, the Tanimoto Metric (TM) value of MAC is the best among the four methods (0.876, see Table 4.1). In addition to that, MAC is 6 times out of 10 better than FVF (only 4 out of 10). However, when the initial contour (not shown) is outside the tumor, the TM value of MAC is the worst (0.080, see Table 4.1). Also note that a perfect TM value of 1.0 is difficult to achieve especially when the target object is small. For example, the TM value is only 0.876 even though the method generated contour and the ground-truth are similar (see (b) and (i) in Figure 4.13). This is because the brain tumors are small, composed of only 50 to 200 pixels in the MR images; a few pixels of deviation from the ground-truth can result in a less than perfect TM value.

Table 4.1: Quantitative analysis of GVF, BVF, MAC and FVF based on IBSR brain tumor MR images.

Image	Method	TM (inside)	TM (outside)	TM (overlap)
#1	GVF	0.173	0.340	0.201
	BVF	0.004	0.350	0.430
	MAC	0.013	0.061	0.013
	FVF	**0.515**	**0.519**	**0.521**
#2	GVF	0.005	0.217	0.005
	BVF	0.014	0.468	**0.664**
	MAC	0.015	0.048	0.022
	FVF	**0.582**	**0.589**	0.598
#3	GVF	0.004	0.117	0.004
	BVF	0.004	0.572	0.007
	MAC	**0.722**	0.074	**0.719**
	FVF	0.709	**0.716**	0.710
#4	GVF	0.410	0.418	0.414
	BVF	0.640	0.574	0.649
	MAC	**0.876**	0.080	**0.842**
	FVF	0.667	**0.656**	0.661
#5	GVF	0.005	0.092	0.009
	BVF	0.585	0.478	0.576
	MAC	0.005	0.075	0.041
	FVF	**0.653**	**0.653**	**0.658**
#6	GVF	0.005	0.189	0.010
	BVF	0.250	0.372	0.275
	MAC	**0.791**	0.056	0.060
	FVF	0.561	**0.563**	**0.571**
#7	GVF	0.294	0.375	0.317
	BVF	0.540	0.520	0.531
	MAC	**0.764**	0.066	0.036
	FVF	0.565	**0.571**	**0.588**
#8	GVF	0.415	0.370	0.398
	BVF	0.591	0.526	0.572
	MAC	0.012	0.072	0.056
	FVF	**0.608**	**0.607**	**0.599**
#9	GVF	0.000	0.300	0.000
	BVF	0.361	0.000	0.631
	MAC	**0.787**	0.079	**0.781**
	FVF	0.599	**0.597**	0.598
#10	GVF	0.004	0.106	0.011
	BVF	0.000	**0.626**	0.386
	MAC	**0.813**	0.162	0.041
	FVF	0.617	0.616	**0.589**

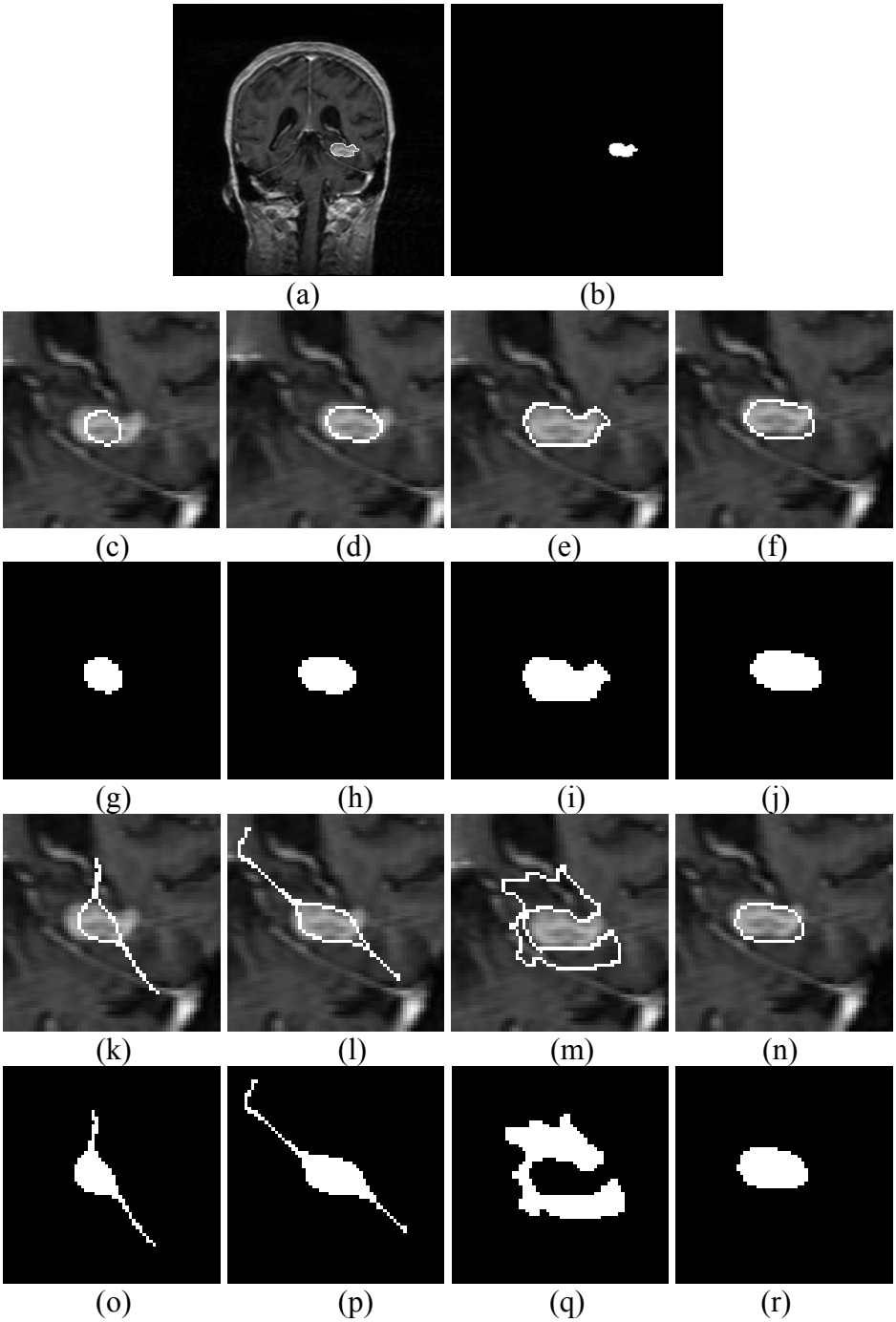


Figure 4.13: (a) Ground-truth and (b) the segmented region of ground-truth of Image #4 in Table 4.1, and the results of (c) GVF, (d) BVF, (e) MAC, (f) FVF, and the segmented regions of (g) GVF, (h) BVF, (i) MAC, (j) FVF, when the initial contour (not shown) is inside the brain tumor; and the results of (k) GVF, (l) BVF, (m) MAC, (n) FVF, and the segmented regions of (o) GVF, (p) BVF, (q) MAC, (r) FVF, when the initial contour (not shown) is inside the brain tumor.

Table 4.2 shows the mean, median and standard deviation of the TM of GVF, BVF, MAC, and FVF. FVF has the largest mean and median with smallest standard deviation. Randomized Block Factorial (RBF) model [14] is used to statistically analyze the experiment. Table 4.2 is the ANOVA table for RBF design. In Table 4.2, *b* represents the set of three test conditions (inside, outside and overlap), *a* represents the set of methods (GVF, BVF, MAC, and FVF), *s* represents the set of test images (#1-#10), *a* * *b* represents the combination of *a* and *b*. The *ab* interaction is significant according to this analysis. Table 4.3 shows the comparison between FVF and GVF, BVF, and MAC. Since all the P-values are smaller than 0.01, FVF is statistically better than the other models in this experiment.

Table 4.2: ANOVA Table for RBF Design. *b* = {inside, outside, overlap}, *a* = {GVF, BVF, MAC, and FVF}, *s* = {test image #1, …, test image #10}, *a* * *b* represents the combination of *a* and *b*.

Source	Partial ss	Df	MS	F	Prob > F
Model	5.18591618	20	0.259295809	6.60	0.0000
s	0.91861415	9	0.102068239	2.60	0.0098
a	3.18540322	3	1.06180107	27.05	0.0000
b	0.022600796	2	0.011300398	0.29	0.7505
a*b	1.05929801	6	0.176549668	4.50	0.0005
Residual	3.88652906	99	0.039257869		
Total	9.07244524	119	0.076239036		

Number of obs = 120, R-squared = 0.5716, Root MSE = 0.198136, Adj R-squared = 0.4851

Table 4.3: Comparison between FVF and GVF, BVF, and MAC.

Comparison	Estimate	P-value
FVF-GVF	0.4349	<0.0001
FVF-BVF	0.2020	0.0072
FVF-MAC	0.3358	<0.0001

4.4.5 Implementation

The MATLAB source codes of GVF and BVF are obtained from the authors of [2] and [3]. The executable java code of MAC is provided by the authors of [4]. FVF is implemented in MATLAB. In our implementation, the default value of T is 0.1. The program also provides the binary boundary maps generated by 8 other values (0.2, 0.3, …, 0.9) of T so that the software user can choose the best value to define the edges of the target object, if the default value does not work. FVF is implemented in MATLAB and not optimized for speed. In general, it takes FVF about 1 to 5 seconds to process a 256x256 image on a Pentium 4 (3GHz CPU, 2GB RAM) desktop computer.

4.4.6 Conclusions

We proposed a new parametric Flexible Vector Flow (FVF) active contour model to address the issues of limited capture range and the inability to extract complex contours with acute concavities. Experiments on synthetic images and head MR images show that FVF produces better results compared to GVF, BVF and MAC. Quantitative experiments on brain tumor images show that FVF has the largest mean (0.61) and median (0.60) with smallest standard deviation (0.05) using Tanimoto Metric. Mixed effects model with random data and test effects is used to statistically compare the differences between FVF and other three methods. Since all the P-values are smaller than 0.01, FVF is statistically better than the other models in this experiment. In the next chapter, FVF will be extended to 3D for brain tumor segmentation.

Bibliography

[1] M. Kass, A. Witkin, and D. Terzopoulus, "Snakes: Active contour model", *Intl. J. of Computer Vision*, vol. 1(4), pp. 321-331, 1988.

[2] C. Xu and J. L. Prince, "Snakes, shapes, and gradient vector flow", *IEEE Trans. on Image Processing*, pp. 359-369, 1998.

[3] K.W. Sum and P. Y. S. Cheung, "Boundary vector field for parametric active contours", *Pattern Recognition*, pp. 1635-1645, 2007.

[4] X. Xie and M. Mirmehdi, "MAC: Magnetostatic Active Contour Model", *IEEE Trans. PAMI*, vol. 30(4), pp. 632-645, 2008.

[5] O. Juan, R. Keriven, and G. Postelnicu, "Stochastic motion and the level set method in computer vision: Stochastic active contours", *Intl. J. of Computer Vision*, vol. 69(1), pp. 7–25, 2006.

[6] I. Dagher and K. E. Tom, Water, "Balloons: A hybrid watershed Balloon Snake segmentation", *Image and Vision Computing*, vol. 26 (7), pp. 905-912, 2008.

[7] S. W. Yoon, C. Lee, J. K. Kim and M. Lee, "Wavelet-based multi-resolution deformation for medical endoscopic image segmentation", *J. of Med. Systems*, vol. 32 (3), pp. 207-214, 2008.

[8] B. H. Lee, I. Choi and G. J. Jeon, "Motion-based boundary tracking of moving object using parametric active contour model", *IEICE Trans. on Info. and Sys.*, E90-D (1), pp. 355-363, 2007.

[9] M. Li and C. Kambhamettu, "Automatic Contour Tracking in Ultrasound Images", *Clinical Ling. and Phon.*, vol. 19 (6-7), pp. 545-554, 2005.

[10] L. He, Z. Peng, B. Everding, X. Wang, C. Y. Han, K. L. Weiss, and W. G. Wee, "A comparative study of deformable contour methods on medical image segmentation", *Image and Vision Computing*, vol. 26, pp.141-163, 2008.

[11] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, USA: Academic Press, 1999, p. 366.

[12] B. Morse, W. Liu, T. Yoo, and K. Subramanian, "Active contours using a constraint-based implicit representation", in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2005, pp. 285–292.

[13] http://www.cma.mgh.harvard.edu/ibsr/

[14] R. E. Kirk, *Experimental design: procedures for the behavioral sciences*, Pacific Grove, Calif.: Brooks/Cole, 1995.

[15] **T. Wang**, I. Cheng and A. Basu, "Fluid Vector Flow and Applications in Brain Tumor Segmentation", *IEEE Transactions on Biomedical Engineering*, Vol. 56(3), pages 781-789, 2009.

Chapter 5 Fully Automatic Brain Tumor Segmentation using a Normalized Gaussian Bayesian Classifier and 3D Flexible Vector Flow

5.1 Introduction

Brain tumor segmentation from Magnetic Resonance Imaging (MRI) is an important task for neurosurgeons, oncologists and radiologists to measure tumor responses to treatments [1-2]. It can indicate drug efficacy in clinical trials of new drugs, and also be used for planning of radiation therapy. Manual segmentation takes up much time and is prone to error. Therefore, automatic brain tumor segmentation methods have been highly desirable in recent decades. However, automatic brain tumor segmentation is a very challenging task due to many factors. First, different types of brain tumors have large variations in sizes, shapes, locations and intensities. Second, similarities between brain tumors and normal tissues are often observed. Last but not least, most brain tumor databases are not publicly available due to proprietary and privacy reasons. Thus, it is very difficult to compare and improve brain tumor segmentation techniques based on a benchmark database.

Many brain tumor segmentation methods have been proposed in the last decade. They can be classified into two major categories: training based methods and non-training based methods. Note that the distinctions between these two categories are sometimes blurry. In the literature, methods in both areas can also be classified using other criteria, such as region-based or voxel-based. Table 5.1 compares the related methods. In Table 5.1, NA means not available or not applicable. The 1st column shows the method names: Hopfield Neural Network (HNN), Knowledge Based (KB), K Nearest Neighbour (KNN), Adaptive, Template Moderated (ATM), Expectation Maximization (EM), Support Vector Machine (SVM), Fuzzy Connectness (FC), Spiral Scanning (SS), Segmentation by Weighted Aggregation (SWA), Graph Cuts (GC), Fuzzy C-means Segmentation (FCM), Fuzzy Algorithms for Learning Vector Quantization

(FALVQ), and Active Contour Model (ACM). The 2nd column lists the two categories (CA) of the methods: training based (TB) or non-training based (NB). The 3rd column reveals the availability of the database (DB): private (PR) or public (PU). The 4th column indicates automatic (AU) status of the method: fully (F) automatic, semi (S) automatic or unknown (NA). The 5th column displays the dimensionality (DI) of the method: 2D or 3D. The 6th column demonstrates the modality of the MRI: T1 weighted (T1), T2 weighted (T2), T1 with contrast enhancement (T1c), or Proton Density weighted (PD). The 7th column shows the tumor types: glioblastoma multiforme (GBM), astrocytoma (AA), low grade glioma (GA), meningioma (MA). The 8th column shows the number of training cases (TR #). The 9th column compares the number of test cases (TE #). The 10th column displays the executing time. Note that the methods were implemented with various programming languages on a variety of platforms. The executing times may not be compared directly. The 11th column lists the accuracy.

Table 5.1: Comparison of related methods.

Method	**CA**	**DB**	**AU**	**DI**	**Modality**	**Tumor**	**TR #**	**TE #**	**Time**	**Accuracy**
HNN[3]	TB	PR	F	2D	NA	NA	NA	2	NA	NA
KB[4]	TB	PR	F	3D	T1,T2,PD	GBM	3	7	NA	.70
KNN[7]	TB	PR	NA	2D	T1,T1c,T2,PD	GA	1	9	1-2m	NA
ATM[8,9]	TB	PR	F	3D	NA	MA,GA	NA	20	10m	.99
EM[10,11]	TB	PR	F	3D	T1,T1c,T2	GBM,MA	1	5	1h4m	.49-.94
SVM[12]	TB	PR	NA	2D	NA	NA	NA	11	6.85s	.86-.94
FC[13]	TB	PR	S	3D	T1,T1c,FLAIR	GBM	NA	10	16m	.99
SS [14]	TB	PR	F	2D	T1	NA	NA	16	21s	.66-.99
SWA[15,16]	TB	PR	F	3D	T1,T1c,T2,FLAIR	GBM	10	10	7m	.27-.88
GC[17]	TB	PR	F	3D	T1,T1c,T2	AA	NA	6	5m	.78±.17
FCM[18]	NB	PR	S	3D	T1,T2	GBM	0	1	NA	NA
FALVQ[19]	NB	PR	NA	2D	T1,T2	MA	0	1	NA	NA
ACM[20]	NB	PR	F	3D	T1,T1c	GBM	0	3	NA	.85-.93
Proposed	**TB**	**PU**	**F**	**3D**	**T1**	**MA,GA,AA**	**1**	**10**	**7m**	**.22-.88**

Training based methods often use some brain tumor images and/or healthy brain images to train a segmentation model and use other brain tumor images to

test the model. Zhu *et al.* [3] formulated brain tumor segmentation as an optimization process that seeks the boundary points to minimize an energy-functional based on snakes. A modified Hopfield Neural Network (HNN) was constructed and trained to solve this optimization problem. The neural network can ensure convergence of the energy minimization by strictly reducing the energy in each iteration. However, this method was applied in 2D and the image modality and tumor type are not given in the paper. A Knowledge Based (KB) clustering method [4] with multi-spectral analysis (T1, T2, and PD) was proposed and trained to segment Glioblastoma Multiforme (GBM) [5]. The KB system took advantage of its coarse-to-fine operation to apply incremental refinement with easily identifiable brain tissues that had already been located and labelled. GBM is an almost non-treatable brain tumor with nearly zero five-year recurrence-free survival rate [6]. It is not clear whether this method can segment other types of brain tumor or not. Vinitski *et al.* [7] proposed a brain tumor segmentation algorithm based on a 4D (T1, T1 with contrast enhancement, T2, and PD) feature map. The k-nearest neighbour (KNN) algorithm was then modified by discarding a few image points according to some heuristic rules to speed up segmentation. It also demonstrated that utilizing multiple MRI protocols often provide better segmentation. However, the accuracy of this algorithm was not given in the paper. Warfield *et al*. [8] presented an adaptive, template moderated (ATM), spatially varying statistical classification (SVC) method for brain tumor segmentation. Kaus *et al*. [9] extended this idea and proposed a classification algorithm to segment brain MRIs into five different tissue classes (background, skin, brain, ventricles, and tumor). The algorithm was validated in a dataset of 20 patients with low-grade gliomas and meningiomas. Prastawa *et al*. [10, 11] proposed a brain tumor segmentation framework based on an Expectation Maximization (EM) algorithm and outlier detection. The EM algorithm was used to estimate a Gaussian mixture model for the global intensity distribution, while tumors were considered as outliers of the Gaussian model. This framework was validated in a relatively small dataset containing only five patients and the algorithm took a very long time (1 hour) to segment the brain tumor of one patient. Zhang *et al.* [12]

proposed a brain tumor segmentation method based on an unsupervised one-class support vector machine (SVM) algorithm, which had the ability of learning the nonlinear brain tumor distribution without any prior knowledge. Morphological filters such as dilation and erosion operations were applied as the post-processing technique to merge tumor regions and remove isolated and small non-tumor parts. This method was applied in 2D and the image modality and tumor type are not given in the paper. Liu *et al.* [13] collected and learned information about different aspects of the tumor and its neighbourhood areas from multiple MRI protocols (FLAIR, T1, and T1 with contrast enhancement). A fuzzy logic framework was then used for tumor segmentation. This method is semi-automatic and operator interaction is required. Cobzas *et al* [56] proposed a variational brain tumor segmentation algorithm using a high dimensional feature set trained from MRI data and registered atlases. The paper focused on how to use a conditional model to discriminate between normal tissues and brain tumors. Wang *et al.* [14] transformed the 3D brain tumor MRIs into 2D with a "spiral scanning" technique. Dynamic programming was used to delineate an optimal outline of the brain tumor in the transformed 2D image. The optimal outline was transformed back into 3D space to determine the volume of the tumor. This method was applied in 2D and it is not clear which types of tumor it can segment. Dube *et al.* [15] integrated contextual filter responses into the multilevel segmentation by weighted aggregation (SWA) algorithm to segment GBM. The SWA algorithm used voxel intensities in a neighbourhood to compute an affinity between the respective voxels. The affinity was recursively calculated for every voxel pair in the brain MRIs to generate a series of "cuts" (segments) containing voxels with similar intensities. A contextual filter response that was computed by texture filter responses based on the gray level co-occurrence matrix (GLCM) method was then integrated to label the cuts as tumor or non-tumor. Corso *et al.* [16] extended [15] and integrated a Bayesian formulation into the SWA algorithm to segment GBM. However, it is not clear whether this method can segment other types of brain tumor or not. Wels *et al.* [17] presented a top-down segmentation algorithm based on a Markov random field (MRF) and graph cuts (GC). Probabilistic boosting

trees (PBT) were used for supervised learning of the model. However, the test dataset (6 patients) was relatively small and contains only one type of brain tumor.

Non-training based methods, on the other hand, do not have any training or learning process. Since the training or learning process can provide valuable information for brain tumor segmentation, only a handful of non-training based methods have been proposed in the literature. One of the first non-training based methods of 3D brain tumor segmentation was presented by Phillips *et al.* [18] in 1995. A fuzzy C-means (FCM) clustering algorithm was used to segment brain tumors from normal brain tissues. FCM is similar to the k-means algorithm for unsupervised clustering but allows labels to be “fuzzy", which means a pixel can be partly in one class and partly in others. Clustering is based on the concept of separated distributions. One of the drawbacks of this method is that human interaction is required to segment brain tumor. Karayiannis *et al.* [19] proposed a fuzzy algorithm for learning vector quantization (FALVQ) to segment the meningioma of an individual. Feature vectors were formed by the values of different relaxation parameters. Brain tumor segmentation was formulated as an unsupervised vector quantization process, which does not rely on *a priori* information. This algorithm was applied in 2D and was tested in a dataset that contains only one patient. Ho *et al.* [20] incorporated region competition into an active contour model for brain tumor segmentation. The algorithm started with an intensity-based fuzzy logic classification of voxels into tumor and background. The active contour model was then used to segment a brain tumor until convergence is achieved. The algorithm was validated in a relatively small dataset that contains three patients and the time efficiency was not given in the paper.

In clinical practice and research, good brain tumor segmentation methods are highly desired. A useful brain tumor segmentation method should: 1) be fully automatic 2) be able to perform in 3D 3) require only one MR modality because multiple modalities may not always be available 4) be able to segment multiple types of brain tumor 5) work in real-time (seconds) or near real-time (minutes). As we have discussed in the previous two paragraphs (also see Table 5.1), the

existing methods fail to meet one or more desired properties. Moreover, before clinical practice, the method should be validated in a publicly available dataset with reasonable size so that other methods could be meaningfully compared with it to advance the state-of-the-art research.

In this chapter, a brain tumor segmentation method is presented and validated on a publicly available brain tumor segmentation repository [8, 9, 21] with ten patients. This method: 1) is fully automatic 2) is able to perform in 3D 3) requires only T1 MRIs 4) is able to segment three types of brain tumor 5) works near real-time (about 7 minutes). In our method, brain MR images are pre-processed with the software MIPAV [22, 23]. After this pre-processing procedure, there are three stages. In the first stage, a "Normalized" Gaussian Mixture Model (NGMM) is proposed and estimated by Expectation-Maximization (EM) based on the ICBM452 brain atlas [24]. NGMM is then used to model healthy and normal brain tissues. In the second stage, the ICBM Tissue Probabilistic Atlases [25] are utilized to obtain the prior probabilities of different brain tissues. After that, a Gaussian Bayesian Classifier based on the NGMM and the prior probabilities of different brain tissues is exploited to acquire a Gaussian Bayesian Brain Map (GBBM) from the test 3D brain MR images. GBBM is further processed to highlight the brain tumor and initialize a Flexible Vector Flow (FVF) algorithm. In the last stage, FVF is used to segment the brain tumor.

There are two major contributions in this chapter. First, we introduce a new NGMM algorithm to model healthy and normal brain. This model can be easily modified for modeling other tasks in various application domains. Second, we extend our 2D FVF algorithm [26] to 3D space and use it for automatic brain tumor segmentation. One drawback of our previous 2D FVF algorithm was that an initial contour was needed to start the vector flow evolution. In this chapter, we take advantage of the GBBM to provide an initial position of a brain tumor to the 3D FVF algorithm to make this process fully automatic.

The rest of this chapter is organized as follows. Section 5.2 introduces the proposed brain tumor segmentation method. Experimental results are reported in Section 5.3, before the work is concluded in Section 5.4.

5.2 Proposed Method

5.2.1 Pre-processing

MR images must be pre-processed before further processing and analysis. The details of the pre-processing steps are described in Section 5.3 Experiments.

5.2.2 Normalized Gaussian Mixture Model and Gaussian Bayes Brain Map

In this section a new Normalized Gaussian Mixture Model (NGMM) is proposed. Gaussian Mixture Model (GMM) had been utilized to model the distribution of brain tissues in the literature [10, 11]. The basic idea of GMM is to use multiple Gaussian distributions to model multiple brain tissues such as gray matter (GM), white matter (WM), and cerebrospinal fluid (CSF). To utilize GMM, the Gaussian distributions of the brain atlas and the brain tumor dataset must be aligned correctly. This means that the Gaussian distribution of the GM in the brain atlas must be similar to the Gaussian distribution of the GM in the brain tumor dataset, and the distributions of WM and CSF must match their peers too. Unfortunately, this is not true in our dataset. For example, Fig. 5.1 shows the histogram of the ICBM452 atlas and the histogram of the MRIs of Patient #1. The intensity regions of CSF, GM, and WM are marked. The intensity range of the atlas is [0, 712] while the intensity range of the MRIs of Patient #1 is [0, 567]. Note that the Gaussian distribution of the GM in the brain atlas does not match the Gaussian distribution of the GM in the brain tumor dataset, and the distributions of WM and CSF do not match their peers. Contrast stretching technique is usually used to address the above situation. If the lower and the upper intensity limits are a and *b*, while the corresponding values present in the current image are *c* and *d* respectively, the intensity value of the current voxel P_{in} needs to be stretched to P_{out} which is defined as:

$$P_{out} = (P_{in} - c)(\frac{b-a}{d-c}) + a \tag{5.1}$$

However, this approach does not work in our test dataset because of the "long tail" problem that exists in all patients' data. A "long tail" in the histogram of MR images of Patient #1 can be clearly observed in Figure 5.1 (right). The "long tail"

region contains a small number of voxels with high intensity values. These voxels could represent image noise or brain abnormalities. Noise filters can reduce the noise at the risk of eliminating potential brain abnormalities with high intensities, and cannot remove the "long tail" entirely. Contrast stretching cannot align the Gaussian distributions in case a "long tail" exists.

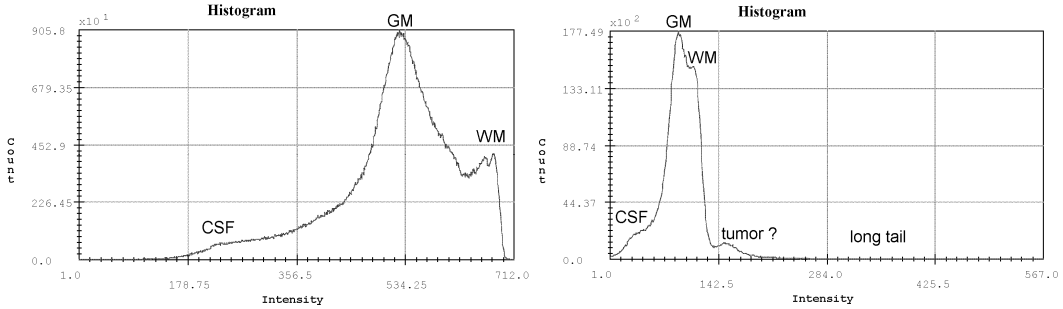


Figure 5.1: (Left) histogram of ICBM452 brain atlas. (Right) histogram of the MR images of patient #1.

To align the Gaussian distributions without eliminating potential brain abnormalities, we define the normalized intensity value P_{out} as:

$$P_{out} = \frac{P_{in}}{m} \tag{5.2}$$

where m is the mean image intensity.

With this definition, the mean image intensity is normalized to 1 and the majority of image intensity is stretched to a range around the mean value. The Gaussian distributions are also aligned so that GMM can be utilized.

We present a Normalized Gaussian Mixture Model (NGMM) to model the healthy or normal brain in the ICBM452 atlas. The basic idea of the NGMM is to estimate a Gaussian Mixture Model based on the normalized image intensities. The NGMM is defined as:

$$p(x_i) = \sum_{k=1}^{K} p(k)\, p(x_i \mid k) \tag{5.3}$$

where x_i is a voxel in the image, $p(k)$ is the prior probability, and $p(x_i \mid k)$ is the conditional probability density, which is define as:

$$p(x_i \mid k) = N(x_i; \mu_k, \sigma_k) = \frac{1}{\sigma_k \sqrt{2\pi}} \exp\left(-\frac{(x_i - \mu_k)^2}{2\sigma_k^2}\right) \tag{5.4}$$

where μ_k is the mean and σ_k is the standard deviation of the Gaussian $N(x_i; \mu_k, \sigma_k)$ for class *k*.

Expectation Maximization

The Expectation Maximization (EM) algorithm [39, 40, 41] is often used to estimate the parameters of a Gaussian Mixture Model. A Quasi-Newton Method [42] had been proposed to accelerate the EM algorithm. The EM algorithm has two stages: *expectation* and *maximization*.

Expectation

With an initial guess for the parameters of the GMM, partial membership of each image voxel in each distribution is estimated by computing expectation values for the membership variables of each data voxel. For each data voxel x_j and distribution Y_i, the membership value $y_{i,j}$ is:

$$y_{i,j} = \frac{a_i f_Y(x_j)}{f_X(x_j)} \tag{5.5}$$

Maximization

Once the expectation values are computed for group membership, estimates are re-calculated for the distribution parameters. The blending coefficients (a_i's)are the means of the membership values over the *N* data voxels.

$$a_i = \frac{1}{N}\sum_{j=1}^{N} y_{i,j} \tag{5.6}$$

The mean values (θ_i's) are also computed by expectation maximization using image voxels x_j that have been weighted using the membership values.

$$\theta_i = \frac{\sum_j y_{i,j} x_j}{\sum_j y_{i,j}} \tag{5.7}$$

With new estimates for the blending coefficients (a_i's) and mean values (θ_i's), the process will be repeated again until the parameters of GMM converge.

K-means

The initial guess for the parameters of the GMM is often given by K-means algorithm [43, 44, 45]. Given a set of N data points $(x_1, x_2, ..., x_N)$, the K-means algorithm divides the N observations into K sets $(K < N)$ $S = \{S_1, S_2, ..., S_K\}$ to minimize the within-set sum of squares.

$$\arg\min_S \sum_{i=1}^{K} \sum_{x_j \in S_i} (x_j - \theta_i)^2 \tag{5.8}$$

where θ_i is the mean value of set S_i. In this chapter, K=3 because we aim to model three types of brain tissues, *i.e.*, CSF, GM, and WM. Note that there is not a class for tumor because tumors have quite different intensities (some tumors are black and some tumors are white) in the image. Using a class to represent tumor will make this entire method become trivial and useless.

Prior probabilities

The ICBM Tissue Probabilistic Atlases are utilized to obtain the prior probabilities of different brain tissues. At a given voxel x_i, the prior probability $p(k, x_i)$ is defined as:

$$p(k, x_i) = \frac{\xi(k, x_i)}{\sum_{k=1}^{K} \xi(k, x_i)} \tag{5.9}$$

where $\xi(k, x_i)$ is image intensity of different brain tissues in the three ICBM Probabilistic Atlases (CSF, GM, or WM).

Gaussian Bayesian Brain Map

At a given 3D location (*u, v, w*) the conditional probability $p(k \mid \xi)$ can be calculated by Bayes' Theorem:

$$p(k \mid \xi) = \frac{p(k) \cdot p(\xi \mid k)}{\sum_{k=1}^{K} p(k) \cdot p(\xi \mid k)} \tag{5.10}$$

The correlation coefficient $CC \in [-1,1]$ at this voxel is then calculated:

$$CC = \frac{Cov(\alpha, \beta)}{\sqrt{Cov(\alpha, \alpha) \cdot Cov(\beta, \beta)}}$$

where $\alpha = (p(CSF \mid \xi), p(GM \mid \xi), p(WM \mid \xi))$, $\beta = (p(CSF), p(GM), p(WM))$, and *Cov* is covariance.

The correlation coefficient *CC* can reveal the likelihood of finding a candidate tumor voxel at a given 3D location. When *CC* is close to -1, it means the intensity of this voxel disagrees with the NGMM and this voxel is probably abnormal or likely to be a tumor voxel. When *CC* is close to 1, it means that the intensity of this voxel agrees with the NGMM and this voxel is likely to be normal. The correlation coefficient is then used to define the Gaussian Bayesian Brain Map (*GBBM*):

$$CM_{uvw} = \begin{cases} 1.0 - CC_{uvw} & when\ CC_{uvw} > 0.0 \\ 0.0 - CC_{uvw} & otherwise \end{cases} \tag{5.11}$$

$$GBBM = [a_{uvw}]_{m \times n \times o} \qquad \text{where } a_{uvw} = \Omega \cdot CM_{uvw} \tag{5.12}$$

CC is first mapped to *CM* in the region of [0, 1]. Then *GBMM* is calculated based on CM and Ω. Ω is a scaling parameter that represents the intensity range of *GBMM*. The resulting *GBBM* is a 3D matrix or image with dimension of $m \times n \times o,\ m, n\ and\ o \in N$. GBBM can reveal the likelihood of finding a candidate tumor in the entire image domain.

5.2.3 Candidate Tumor Region

GBMM is further processed to highlight a candidate tumor region. This region will be used to automatically initialize the 3D Flexible Vector Flow algorithm, which will finally segment the brain tumor. The post-processing stage has six steps: removing boundary voxels, thresholding, morphological erosion, locating the largest 3D region, morphological dilation, and reverse transformation.

Brain tumor MR images are registered to a brain atlas in the pre-processing stage using the non-rigid registration method in software MIPAV. The pre-processing stage will be described in Section 5.3.3. However, the registered image and the atlas are still different at a few voxels on the brain boundary. Therefore, the boundary voxels in *GBMM* often have high intensity values. However, the boundary voxels are not in the region of any brain tumor in our dataset. Therefore, boundary voxels must be removed from *GBMM*. A boundary voxel is defined as a voxel that has at least one neighbor with intensity 0 in a $3 \times 3 \times 3$ neighborhood.

A threshold Ψ is then applied to *GBMM* to create a Binary Gaussian Bayesian Brain Map (*BGBBM*):

$$b_{uvw} = \begin{cases} 1 & when\ a_{uvw} > \Psi \\ 0 & otherwise \end{cases} \tag{5.13}$$

$$BGBBM = [b_{uvw}]_{m \times n \times o} \tag{5.14}$$

Morphological filters, such as dilation and erosion, are often used as post-processing techniques [12]. Erosion is applied to merge relatively large separated regions. Then, the largest 3D region is automatically located. This region represents the candidate tumor. Dilation is then used to restore the region to approximately its original size and shape before the erosion. Figure 5.2 shows the GBMM and the candidate tumor region after the dilation step for Patient #1.

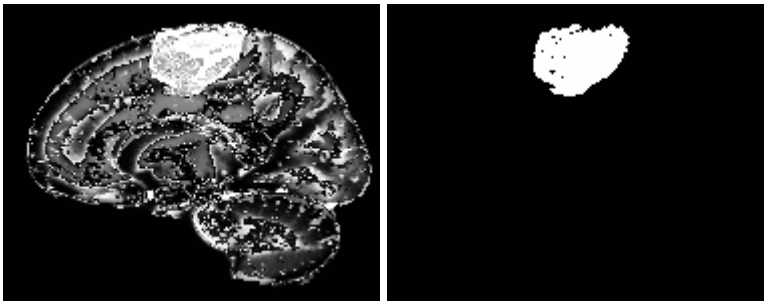


Figure 5.2: (Left) Gaussian Bayesian Brain Map of the brain. (Right) The candidate tumor region after dilation.

Finally, we apply T^{-1} to the dilated image to transform it back to the original image space. T is the non-rigid registration matrix calculated in the pre-processing stage. Figure 5.3 shows the original image, the segmentation ground-truth, and the candidate tumor region after the reverse transformation. The ground-truth was defined as the area of those brain tumor voxels in which at least three of four expert evaluators agreed regarding their identification [9]. Although the difference is still noticeable, the candidate tumor region is similar to the segmentation ground-truth. This candidate tumor region will be used to initialize the 3D Flexible Vector Flow algorithm, which will finally segment the brain tumor.

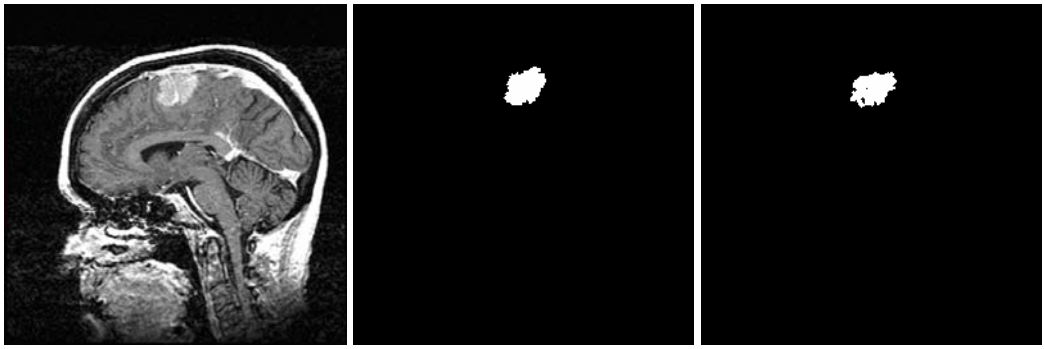


Figure 5.3: (Left) Original image (Middle) ground-truth (Right) candidate tumor region after the reverse transformation.

5.2.4 3D Flexible Vector Flow

Flexible Vector Flow (FVF) [26] is an active contour model that addresses the limitations of other active contour models, such as insufficient capture range and poor convergence for concavities. It had been applied to semi-automatic brain tumor segmentation in 2D space. One drawback of the previous 2D FVF algorithm was that an initial contour was needed to start the vector flow evolution. In this chapter, the FVF algorithm is extended to 3D space and the candidate tumor region is used to initialize the 3D FVF algorithm to make this process fully automatic.

In our work, segmentation is expressed as the mathematical process of finding the surface which can delineate the region of brain tumor given the candidate tumor region in the 3D space of the MRIs. This is a non-linear optimization problem, which is often addressed by using a variational approach. The basic idea is to start with an initial candidate tumor model and deform this model to better delineate the brain tumor until convergence is achieved. The deformable model is usually called active contours or snakes [26, 46, 47]. Active contour models or snakes have been adopted as effective tools for segmentation [20, 26] in the literature.

The traditional active contour model proposed by Kass *et al.* [46] is a 2D parametric active contour:

$$c(s) = (x(s), y(s)), s \in [0,1] \tag{5.15}$$

Given an initial contour, it evolves within an image $I(x, y)$ to minimize the energy function:

$$E_{snake} = \int_0^1 [E_i(c(s)) + E_e(c(s))]ds \tag{5.16}$$

where E_i is the internal (spline) energy and E_e is the external energy.

The internal energy is given by:

$$E_i = \frac{\alpha(s)\,|\,c'(s)\,|^2 + \beta(s)\,|\,c''(s)\,|^2}{2} \tag{5.17}$$

where α and β are first-order and second-order blending parameters.

Many snakes in the literature share the same internal energy and differ mostly in the external energy. A snake should evolve to minimize the energy functional E_{snake}. This problem can be formulated with the Euler-Lagrange equation.

$$\alpha c''(s) - \beta c''''(s) + \nabla E_e = 0 \tag{5.18}$$

To find a numeric solution of (16), the snake is treated as a function of time *t* as well as *s*:

$$\alpha c''(s,t) - \beta c''''(s,t) + \nabla E_e = 0 \tag{5.19}$$

A solution is obtained when the contour stabilizes and the time term vanishes [46]. There are two problems with the traditional active contour model. The first problem is its limited capture range, *i.e.*, it is not able to deform far from its initial contour. The second problem is its poor convergence for concavities. These two problems were addressed in the previous chapter.

While parametric active contour models mainly focus on the 2D domain, level set snakes had been used for 3D segmentation and reconstruction [47]. A level set [48, 49, 50] snake is an implicit model, which is not explicitly expressed as a parametric model but is implicitly specified as a level set of a scalar function ϕ. A 3D surface may be written as:

$$f(x,y,z) = (x(s_1,s_2), y(s_1,s_2), z(s_1,s_2)) \tag{5.20}$$

F = F(K) is the speed of the surface evolution

$$F^2(f_x^2 + f_y^2 + f_z^2) = 1 \tag{5.21}$$

where K is the mean curvature:

$$K = \frac{f_{xx}(f_y^2 + f_z^2) + f_{yy}(f_x^2 + f_z^2) + f_{zz}(f_y^2 + f_x^2) - 2f_{xy}f_xf_y - 2f_{xz}f_xf_z - 2f_{yz}f_zf_y}{(f_x^2 + f_x^2 + f_x^2)^{3/2}} \tag{5.22}$$

The motion of the surface is formulated as a Partial Differential Equation (PDE):

$$\frac{\partial \phi}{\partial t} - F(K)\,|\nabla \phi| = 0 \tag{5.23}$$

where $\phi(x,y,z,t)$ is a scalar function such that at time *t* the zero level set of ϕ is the surface.

Level set snakes have a few drawbacks. First, a significant amount of computations is required to solve these equations over the entire domain. Adalstein and Sethian [49] proposed a fast level set method for propagating interfaces based on a narrow-band method, which used a finite band of 6–12 grid voxels on either side of the level set to reduce the computations. Whitaker [47] presented a sparse-field method, which took the narrow-band method to its extreme by calculating updates on a band of grid voxels that is only one voxel wide. The second drawback of level set is that the resulting snakes may segment wrong objects when there are multiple similar objects near the expected object. The third drawback is the absence of explicit and direct representation of the surface during its evolution.

3D Flexible Vector Flow (FVF) is used to segment brain tumors in this chapter. It is an extension of our 2D FVF algorithm [26], which had been applied to semi-automatic brain tumor segmentation in 2D space. 3D FVF takes the candidate tumor region obtained in the previous section and automatically segments the brain tumor. It uses a parametric representation of the initial surface and also takes advantage of the level set method.

We first address the second drawback of level set snakes − segmenting wrong objects when there are multiple similar objects near the expected object. Figure 5.4 demonstrates this problem. There are four objects in the left figure. The upper left one is the expected object and the rectangle is the initial contour specified by the user. Level set snakes segment all four objects, which is not the desired result because the real intent of the user is to segment only the upper left object.

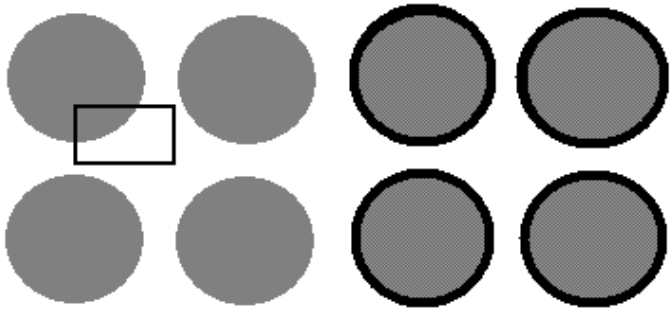


Figure 5.4: (Left) initial rectangle contour and four objects (Right) four objects are segmented by level set snakes.

To address this problem, we need to determine the spatial relationship between the initial contour and the object. Note that we explain this concept in 2D only for the purpose of illustration. The algorithm is implemented in 3D. Once we have obtained the candidate tumor region, we can calculate its "center." In this section, the center is calculated by the Valence Normalized Spatial Median algorithm presented in Chapter 3. Now we have the candidate tumor region and its center, no matter the region is convex or concave. The initial active surface *S* is then defined as:

$$(x-x_0)^2+(y-y_0)^2+(z-z_0)^2=r^2 \tag{5.24}$$

where (x_0, y_0, z_0) is the center of the candidate tumor region and radius *r* is a constant. To make sure that the sphere is inside the candidate tumor region, we require *r* to be smaller than the radius of the maximum inscribed sphere of the candidate tumor region at point (x_0, y_0, z_0). Thus, Equation (23) becomes:

$$\begin{cases} \phi(0,x,y,z)=S \\ \dfrac{\partial\phi}{\partial t}=F(K)\,|\nabla\phi| \end{cases} \tag{5.25}$$

Chan and Vese [53] pointed out the evolution of level set should not always rely on gradient (2D) or surface normal (3D). The basic idea of FVF is to add a directional component to the external force and keep the normal component. At a given point $B=(x_1, y_1, z_1)$ on the level set surface, there are two straight lines L_1 and L_2:

$$L_1: \frac{x-x_1}{l_1}=\frac{y-y_1}{m_1}=\frac{z-z_1}{n_1} \tag{5.26}$$

$$L_2: \frac{x-x_1}{l_2}=\frac{y-y_1}{m_2}=\frac{z-z_1}{n_2} \tag{5.27}$$

Where L_1 is the normal of the surface at point (x_1, y_1, z_1) and L_2 is the straight line determined by vector $\vec{AB}$ that start from center $A=(x_0, y_0, z_0)$ and points to $B=(x_1, y_1, z_1)$ (Figure 5.5).

Next, we extend the external energy function in [26] to 3D:

$$E_e(x,y,z) = \chi(f_x + \delta\gamma_x, f_y + \delta\gamma_y, f_z + \delta\gamma_z) \tag{5.28}$$

where χ is a normalization operator, $\delta = \pm 1$ (controlling the inward or outward direction, when the surface is "outside" or "inside" the candidate tumor region), and γ is the angle between L_1 and L_2, determined by:

$$\cos\gamma = \frac{l_1 l_2 + m_1 m_2 + n_1 n_2}{\sqrt{l_1^2 + m_1^2 + n_1^2}\sqrt{l_2^2 + m_2^2 + n_2^2}} \tag{5.29}$$

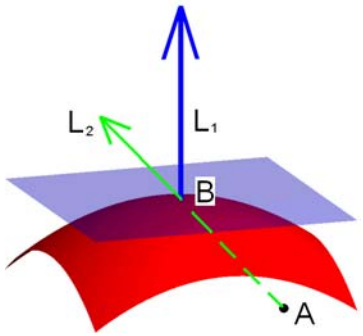


Figure 5.5: The red surface is the level set surface, the blue plane is the tangent plane to that surface, the blue arrow is the surface normal, the black dot is the center of the candidate tumor region, and the green arrow represents the directional component of the external energy. (Image adapted from Wikipedia.)

The external energy $E_e(x,y,z)$ has two components: a normal component and a directional component. The normal force is computed in a manner similar to the traditional snake [46] and GVF snake [54]. The novelty of FVF lies in the computation of the directional force. When the active surface is within the candidate tumor region, the directional force can push the surface towards the boundary of the tumor. When the surface is close to the boundary of the tumor, the normal force fits the surface to the tumor.

5.3 Experiments

5.3.1 Test Dataset

We test the proposed method with the SPL Brain Tumors Image Dataset [8, 9, 21]. It is a freely available brain tumor segmentation repository. It contains brain MR images (SPGR T1 POST GAD) of 10 patients with brain tumors (3 meningiomas, 3 low grade gliomas and 4 astrocytomas). Segmentation ground-truth, which was defined as the area of those brain tumor voxels in which at least three of four expert raters agreed regarding their identification [9], is also available in this repository. The image format is no-header, unsigned short 16-bit (byte order: MSB LSB). The resolution is 256x256x124, with pixel size of 0.9375 x 0.9375 mm, slice thickness of 1.5 mm, slice gap of 0.0 mm.

5.3.2 Brain Atlases

Two groups of brain atlases, the Talairach–Tournoux (TT) brain atlas [27, 28] and the ICBM atlases [24, 25] are often used to collect healthy or normal brain information.

The print TT atlas was constructed from a single brain specimen sectioned and photographed sagittally. Coronal and axial sections were subsequently interpolated manually. The Cerefy database [29] contains an extended and enhanced electronic version of the TT brain atlas. It can read or write DICOM 3 format. It is a free Java program including the atlas. However, the atlas itself is not freely accessible.

In this chapter, we use two ICBM atlases, the ICBM452 atlas [24] and the ICBM Tissue Probabilistic Atlases [25], which are freely accessible in the public domain.

The ICBM452 atlas [24] is a freely available brain atlas. It is an average of intensities and spatial positioning of T1-weighted MR images of normal adult brains. This atlas is not based on any single subject but is constructed from the average position, orientation and scale from a number of individual brains. The ICBM452 atlas is used to estimate a Normalized Gaussian Mixture Model (NGMM) by the Expectation-Maximization (EM) algorithm.

The ICBM Tissue Probabilistic Atlases [25] classified the ICBM452 atlas into gray matter (GM), white matter (WM), and cerebrospinal fluid (CSF). The GM, WM, and CSF maps were separated into different components. Each component was then averaged in atlas space across the subjects to create the probability fields for each tissue type. These fields represent the likelihood of finding GM, WM, or CSF at a specified position for a subject that has been registered to the atlas space. The ICBM Tissue Probabilistic Atlases are used to obtain the prior probabilities of GM, WM, and CSF.

5.3.3 Pre-processing

The Neuroimaging Informatics Tools and Resources Clearinghouse (NITRC) [30] provide links to a variety of brain MR image processing software. We choose to use MIPAV [22, 23] after careful investigation and comparison. MIPAV (Medical Image Processing, Analysis, and Visualization) is a Java application mainly for processing and analysis of brain MR images. It can run on Java-enabled systems such as Windows, UNIX, or Macintosh OS X. We also identified some other powerful tools, such as 3D Slicer [31], FSL [32], STAPLE [33], BioImage Suite [34], ITK-SNAP [35], and software suites provided by Asclepios [36] at INRIA, LONI [37]. They focus on different aspects of 3D neuro-imaging and demonstrate excellence in different applications. We choose MIPAV because it suits our application, the dataset, and computational resources.

In this chapter, the pre-processing stage has two steps: skull stripping and registration. The registration is performed with the non-rigid registration method in MIPAV. Our method is robust enough so that other pre-processing steps, such as noise reduction and inter-scan intensity standardization described in [17], become unnecessary.

SPL Brain Tumors Image Dataset [8, 9] contains the MR scans of patients' heads and structures nearby. To get the Volume Of Interest (VOI), *i.e.*, the brain, we must strip the skull and other non-brain structures and tissues. We use MIPAV to extract brain from MR images. Given the input MR images, MIPAV can automatically extract brains without user intervention. Figure 5.6 shows the

original MR image, the extracted brain, and the 3D volume rendering of the extracted brain of Patient #1 in the dataset.

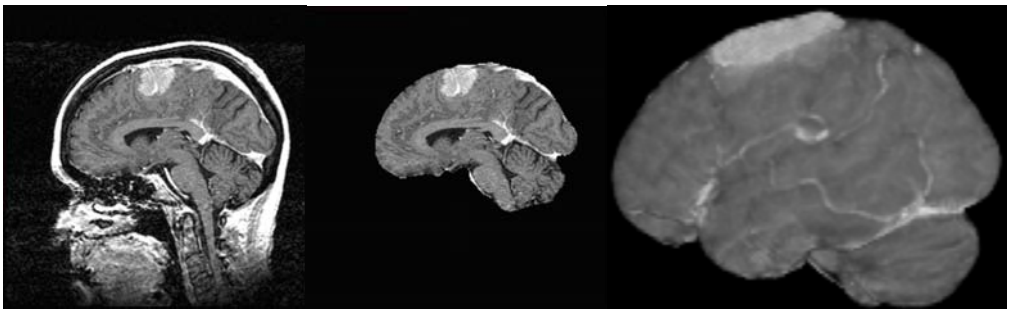


Figure 5.6: (Left) Original image (Middle) extracted brain (Right) 3D volume rendering of the extracted brain.

The extracted brain is then registered [38] to the ICBM atlas space with MIPAV. The basic idea of registration is to find a matrix T that transforms the extracted brain image to the atlas so that the cost function, which represents the quality of alignment between two images, is minimized. In this chapter, the transformation matrix T is non-rigid because the brain is non-rigid. Figure 5.7 shows the ICBM452 atlas, the registered brain and the 3D volume rendering of the registered brain of Patient #1.

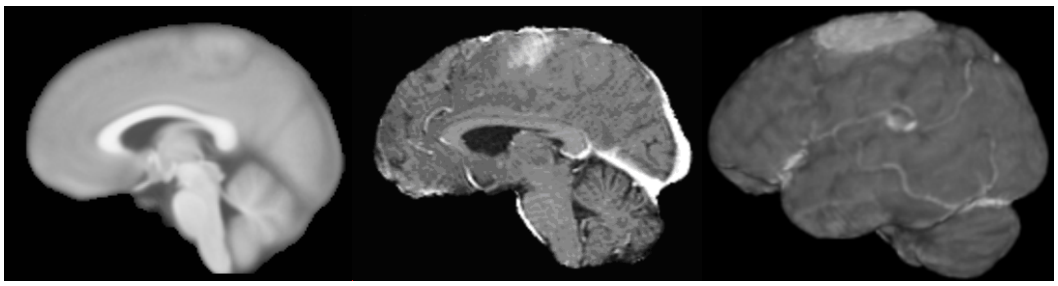


Figure 5.7: (Left) ICBM452 atlas (Middle) registered brain (Right) 3D volume rendering of the registered brain.

5.3.4 The Setting of Parameters

There are five parameters in our implementations: Ψ (in Equation 5.13), α (in Equation 5.17), β (in Equation 5.17), the kernel of dilation and the kernel of

erosion. Ψ is set to 127, α is set to 2.5, β is set to 0, the kernel of dilation is 3x3x3 - 6 connected, the kernel of erosion is 3x3x3 - 6 connected. The setting of parameters follows the general practices of image segmentation with active contour models [53-55].

5.3.5 Experimental Results

The Tanimoto Metric (*TM*) [55] is used for quantitative analysis. It was defined in the previous chapter. Similarly, the percentage of over-segmentation is defined as $OS = \frac{\|R_X - R_X \cap R_G\|}{\|R_X \cup R_G\|}$, the percentage of under-segmentation is define as $US = \frac{\|R_G - R_X \cap R_G\|}{\|R_X \cup R_G\|}$, where R_X is the region enclosed by the surface generated by the proposed method, R_G is the region of the ground-truth segmentation provided in the SPL Brain Tumors Image Dataset, and $\|\cdot\|$ is set cardinality (number of elements). Table 5.2 shows the test results.

Table 5.2: Results of the proposed method

Case	Tumor type	GBMM time	FVF time	Total time	*TM*	*OS*	*US*
1	meningioma	361 sec	51 sec	412 sec	0.88	0.10	0.02
2	meningioma	372 sec	51 sec	423 sec	0.83	0.07	0.10
3	meningioma	375 sec	52 sec	427 sec	0.57	0.23	0.20
4	low grade glioma	374 sec	50 sec	424 sec	0.66	0.21	0.13
5	astrocytoma	375 sec	51 sec	426 sec	0.22	0.78	0.01
6	low grade glioma	389 sec	52 sec	441 sec	0.53	0.43	0.03
7	astrocytoma	388 sec	51 sec	439 sec	0.67	0.01	0.33
8	astrocytoma	384 sec	53 sec	437 sec	0.57	0.30	0.13
9	astrocytoma	361 sec	56 sec	417 sec	0.30	0.64	0.06
10	low grade glioma	397 sec	52 sec	449 sec	0.70	0.18	0.12

The proposed method was run on a Pentium 4 (3GHz CPU, 2GB RAM) desktop computer with Windows XP (Version 2002, SP3) operating system. The resolution is 256x256x124 for each test case. The algorithms were implemented in MATLAB 7.5 (R2007b) and not optimized for speed. The current implementation shows that this method is a near real-time (about seven minutes) algorithm. This method can be implemented as a real-time (seconds) program

with more efficient programming languages such as C or C++ (at least ten times faster than MATLAB).

In Table 5.2, the GBMM time represents the CPU time for calculating the Gaussian Bayesian Brain Map (GBBM), the FVF time reveals the CPU time of executing the proposed 3D Flexible Vector Flow (FVF) algorithm to segment brain tumor. The total time is the sum of GBMM time and the FVF time, which shows the total time used by the proposed method. The pre-processing stage requires about 9 minutes, and the post-processing stage requires about 5 minutes. Thus, the total overhead time is about 14 minutes for each test case. In the pre-processing and post-processing stages, the default settings of MIPAV are applied so that those two stages are fully automatic just as other parts of the proposed method. Note that the Normalized Gaussian Mixture Model (NGMM) was trained off-line. The training time was about 25 minutes. Once the training was finished, the results were used repeatedly in each test case.

Figures 5.8-5.17 show the result of the patient #1-#10.

Utilizing multiple MRI protocols often provide easier and better segmentation [7]. Other factors that can simplify segmentation and increase accuracy include using 2D images, semi-automatic implementation, and testing fewer types of tumor. Our technique implements a fully automatic 3D segmentation for three types of brain tumors in T1 MRIs. As we have discussed in the introduction, a number of existing algorithms do not work in 3D. Some algorithms work in 3D but require multiple MRI protocols. Therefore, it is currently very difficult to compare our method with the existing ones on the SPL dataset [8, 9] which has only T1 MRIs. On the other hand, since other datasets that are used in the existing methods are not available due to proprietary and privacy reasons, it is very difficult to compare our method with the existing ones on those datasets. Nevertheless, we notice that the accuracy (0.22-0.88) of our method matches a recent work by Corso *et al.* [16], where the accuracy was in the range of 0.27-0.88 for segmentation of only one type of tumor glioblastoma multiforme (GBM).

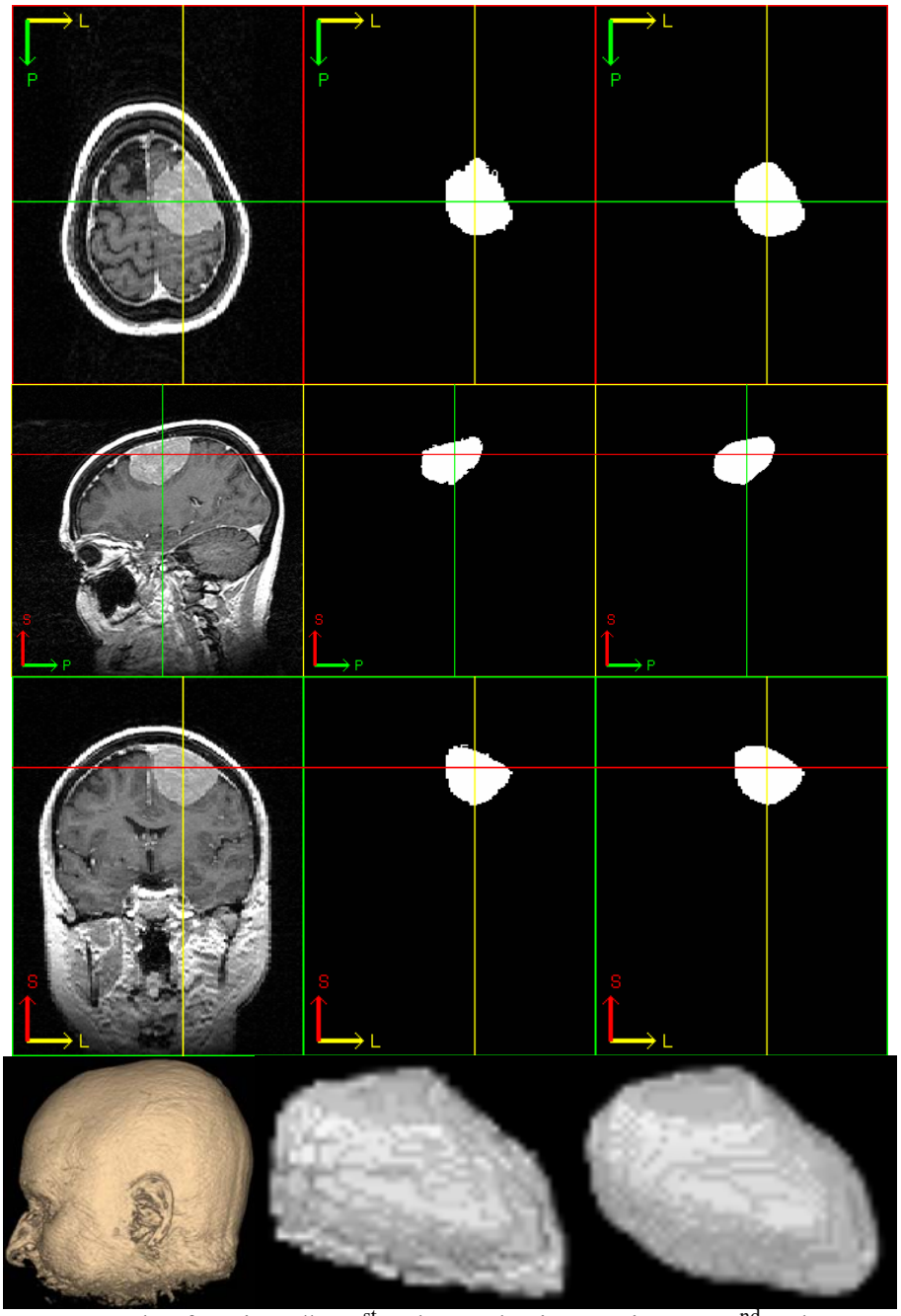


Figure 5.8: Result of Patient #1. 1st column: brain MR image, 2nd column: ground truth, 3rd column: brain tumor extracted by the proposed method. 1st row: axial view, 2nd row: sagittal view, 3rd row: coronal view, 4th row: volume rendering.

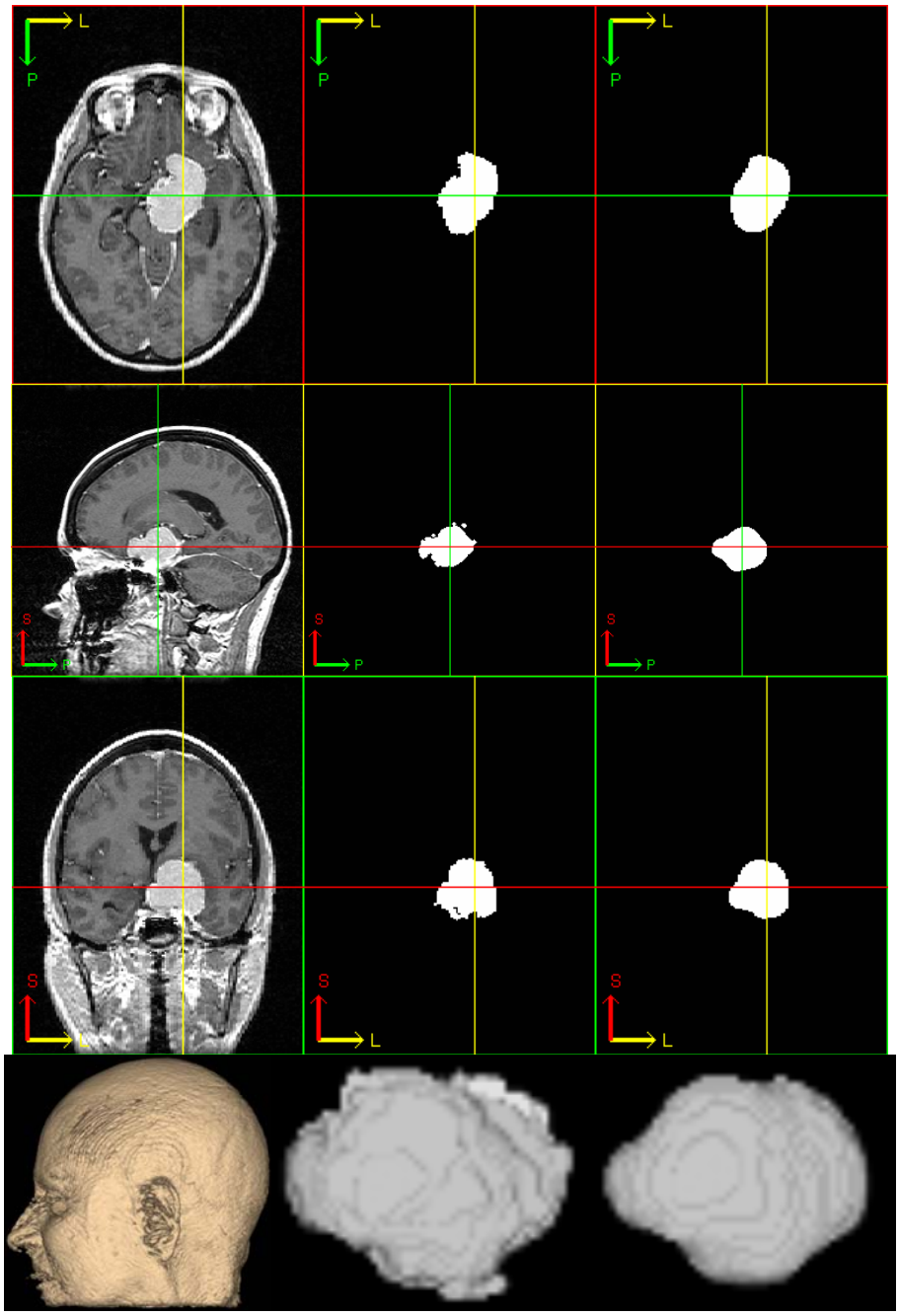


Figure 5.9: Result of Patient #2. 1st column: brain MR image, 2nd column: ground truth, 3rd column: brain tumor extracted by the proposed method. 1st row: axial view, 2nd row: sagittal view, 3rd row: coronal view, 4th row: volume rendering.

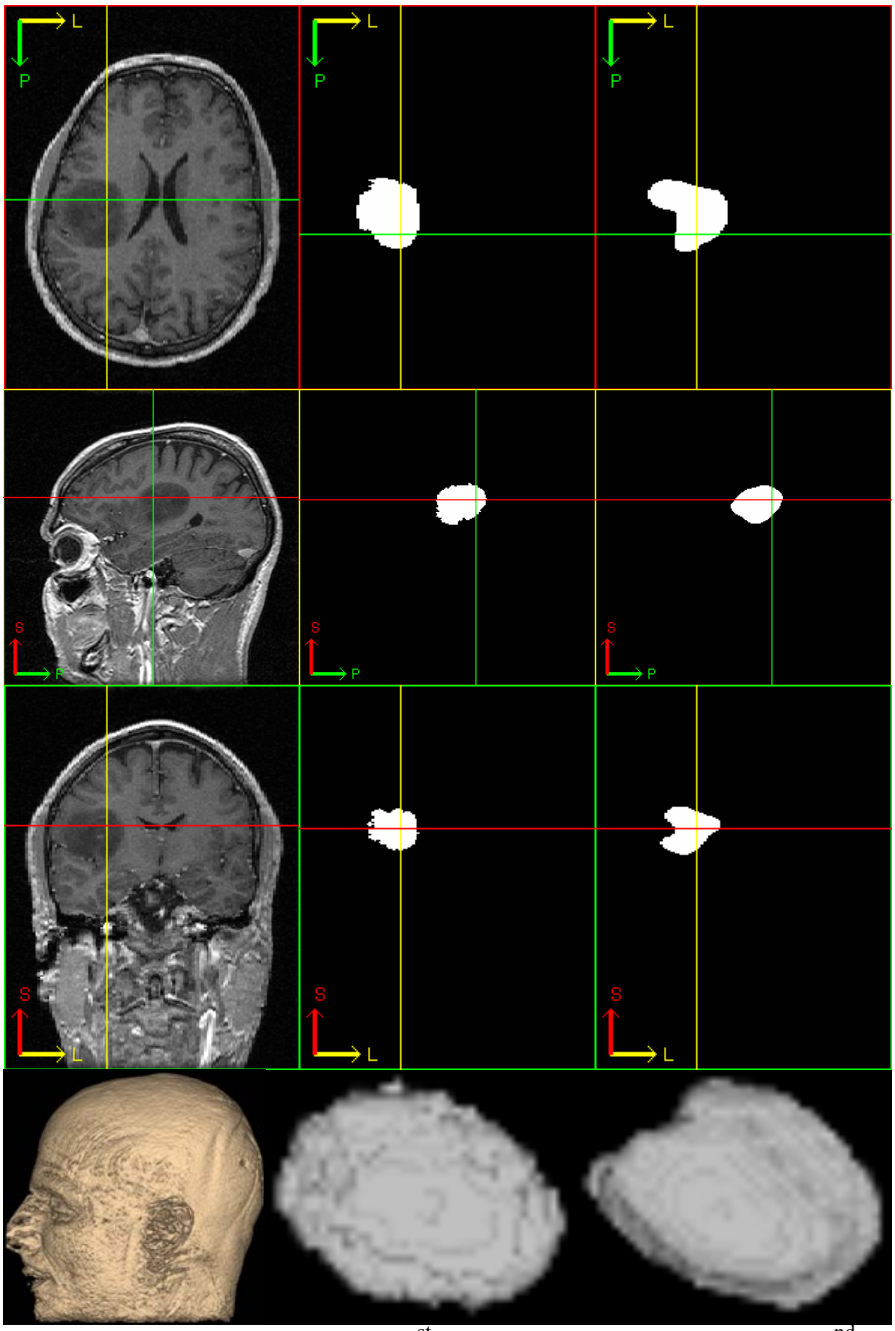


Figure 5.10: Result of Patient #3. 1st column: brain MR image, 2nd column: ground truth, 3rd column: brain tumor extracted by the proposed method. 1st row: axial view, 2nd row: sagittal view, 3rd row: coronal view, 4th row: volume rendering.

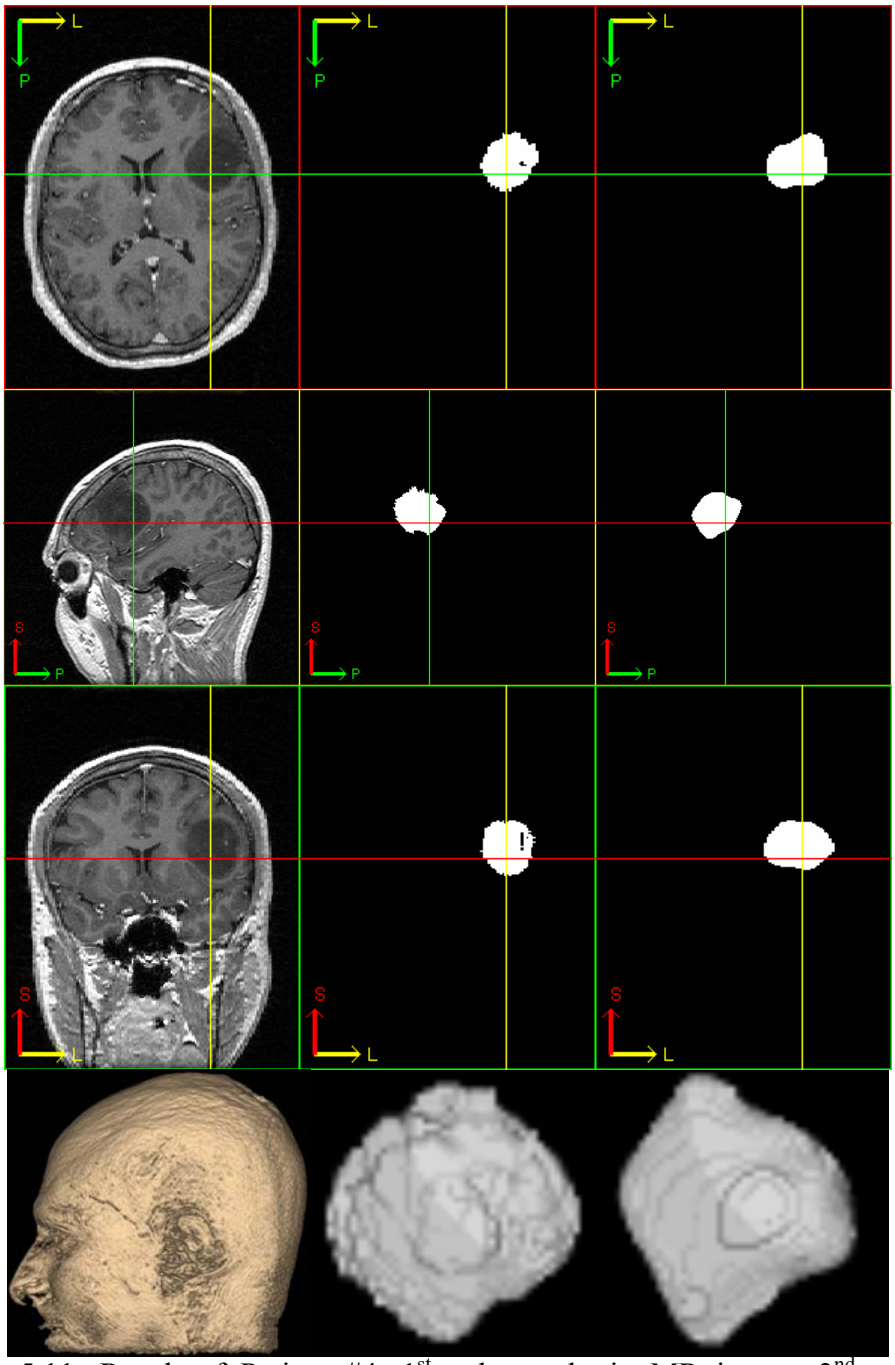


Figure 5.11: Result of Patient #4. 1st column: brain MR image, 2nd column: ground truth, 3rd column: brain tumor extracted by the proposed method. 1st row: axial view, 2nd row: sagittal view, 3rd row: coronal view, 4th row: volume rendering.

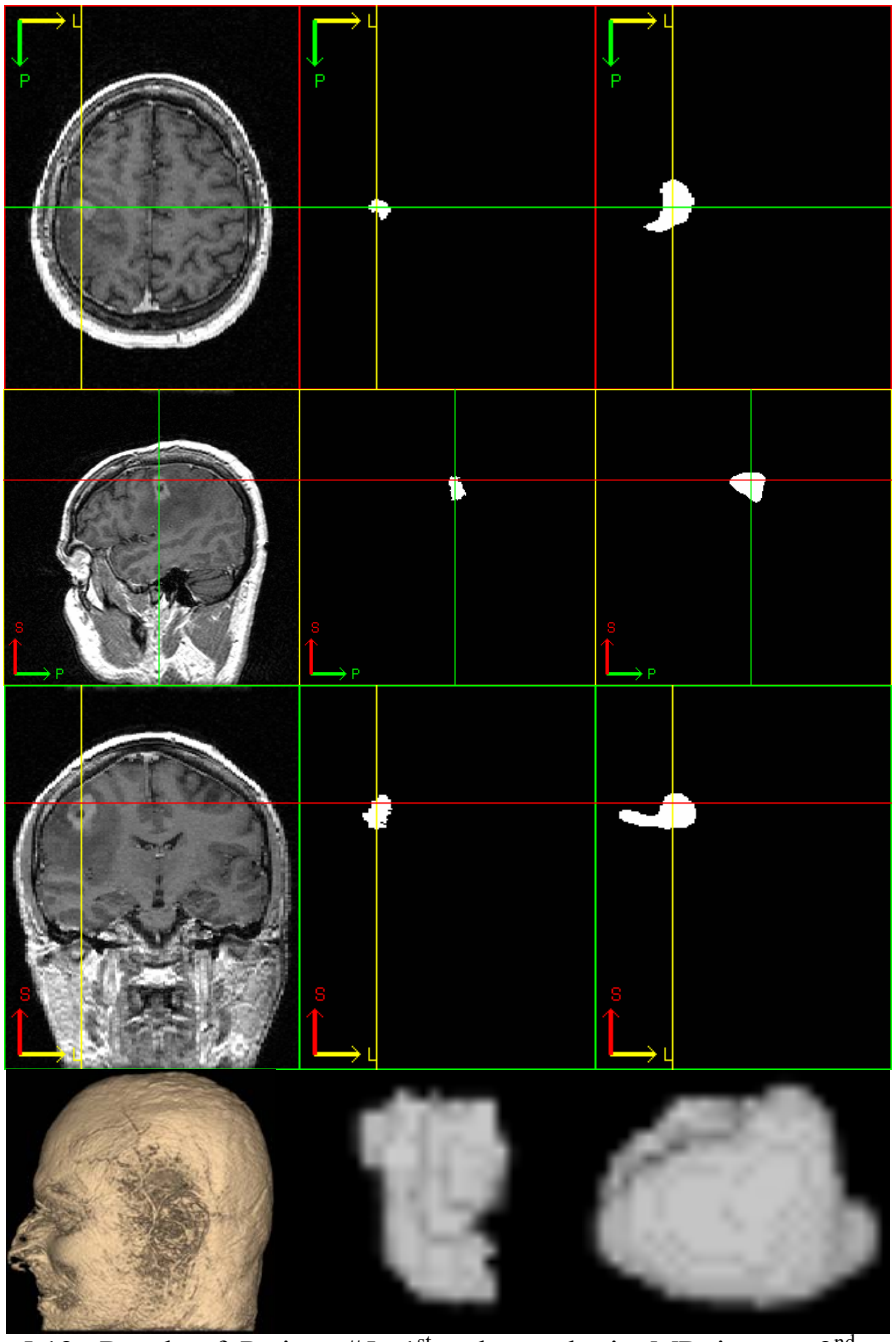


Figure 5.12: Result of Patient #5. 1st column: brain MR image, 2nd column: ground truth, 3rd column: brain tumor extracted by the proposed method. 1st row: axial view, 2nd row: sagittal view, 3rd row: coronal view, 4th row: volume rendering. The tumor region is very small and the intensity is inhomogeneous so that the segmentation accuracy (0.22) is very low.

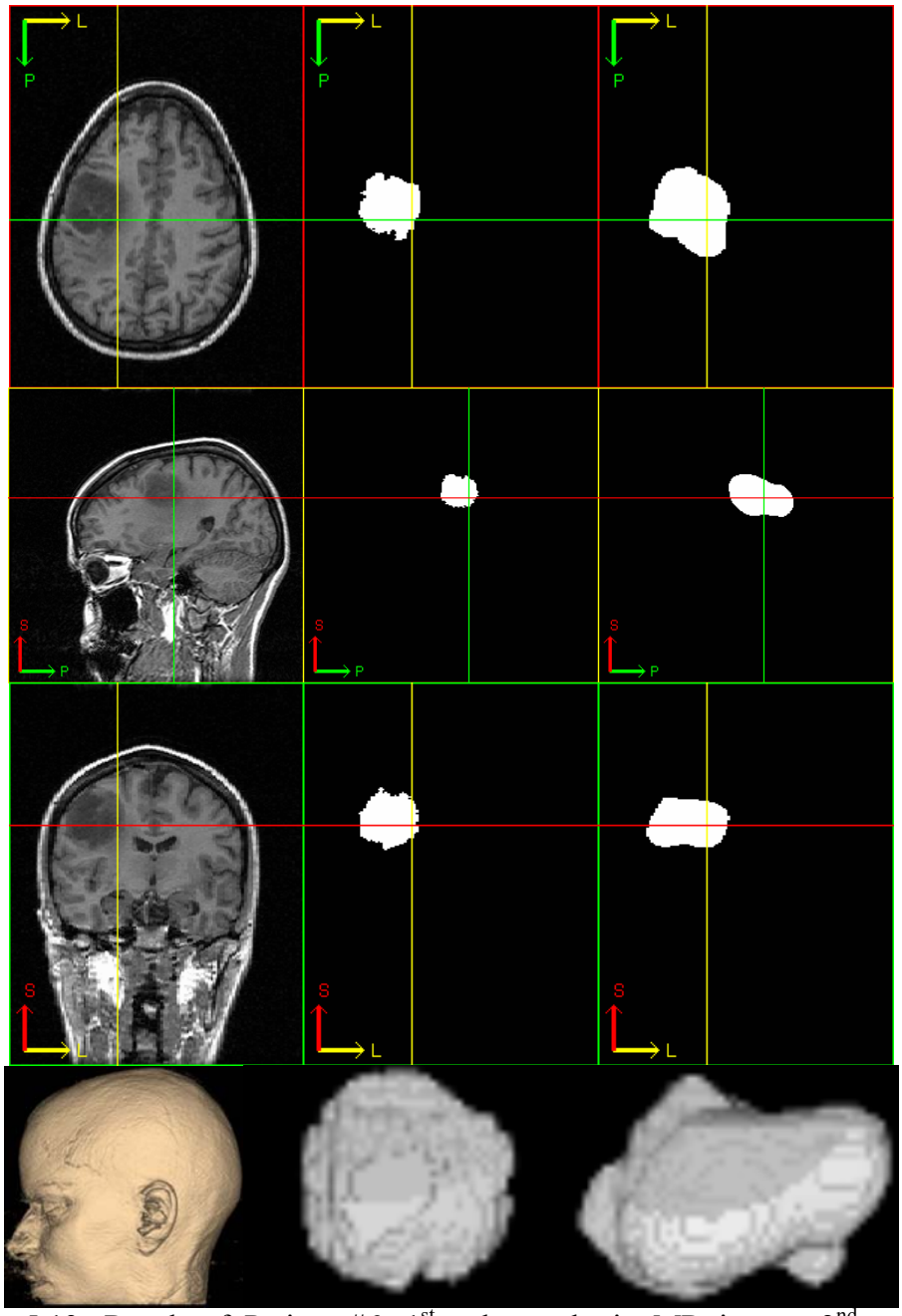


Figure 5.13: Result of Patient #6. 1st column: brain MR image, 2nd column: ground truth, 3rd column: brain tumor extracted by the proposed method. 1st row: axial view, 2nd row: sagittal view, 3rd row: coronal view, 4th row: volume rendering.

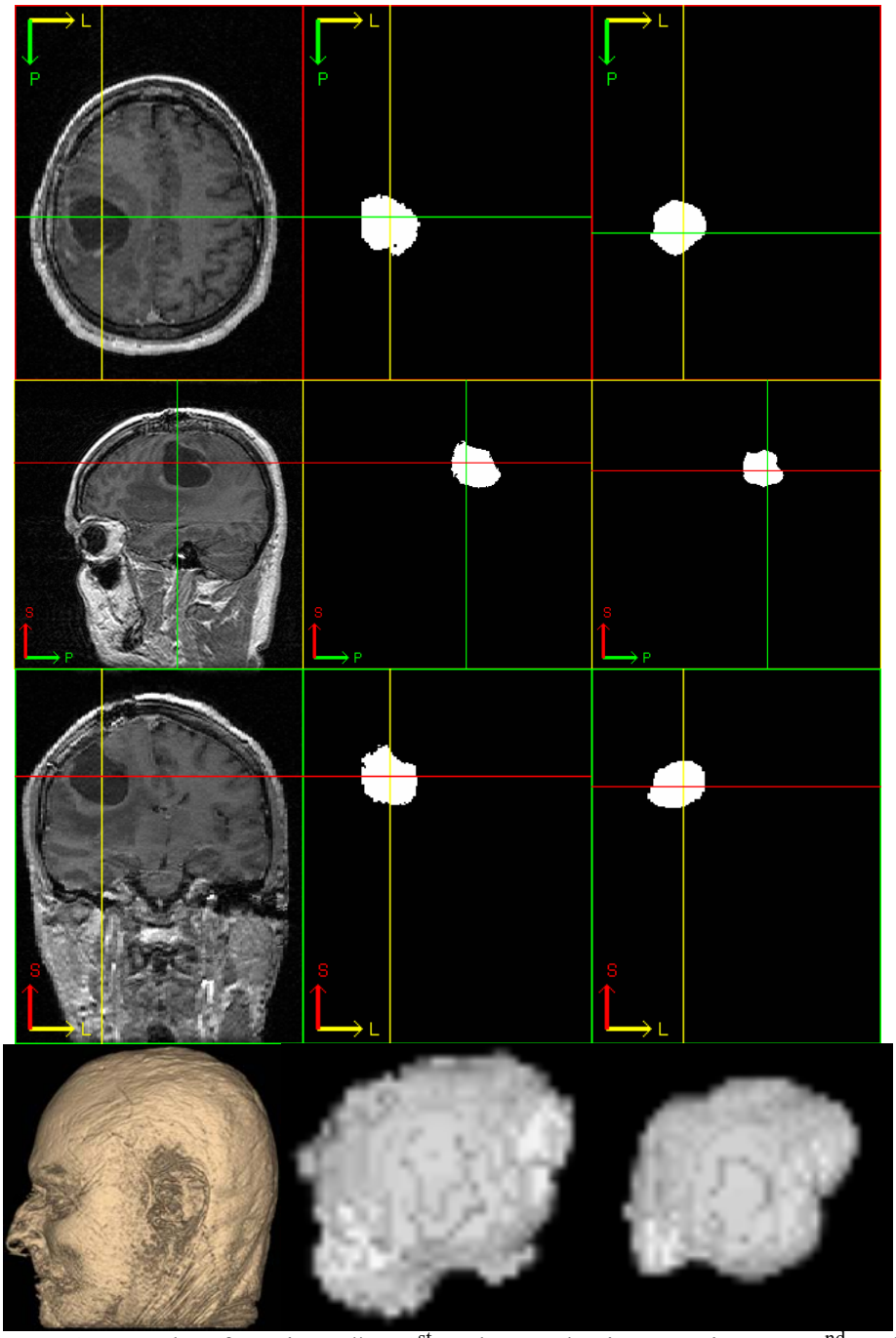


Figure 5.14: Result of Patient #7. 1st column: brain MR image, 2nd column: ground truth, 3rd column: brain tumor extracted by the proposed method. 1st row: axial view, 2nd row: sagittal view, 3rd row: coronal view, 4th row: volume rendering.

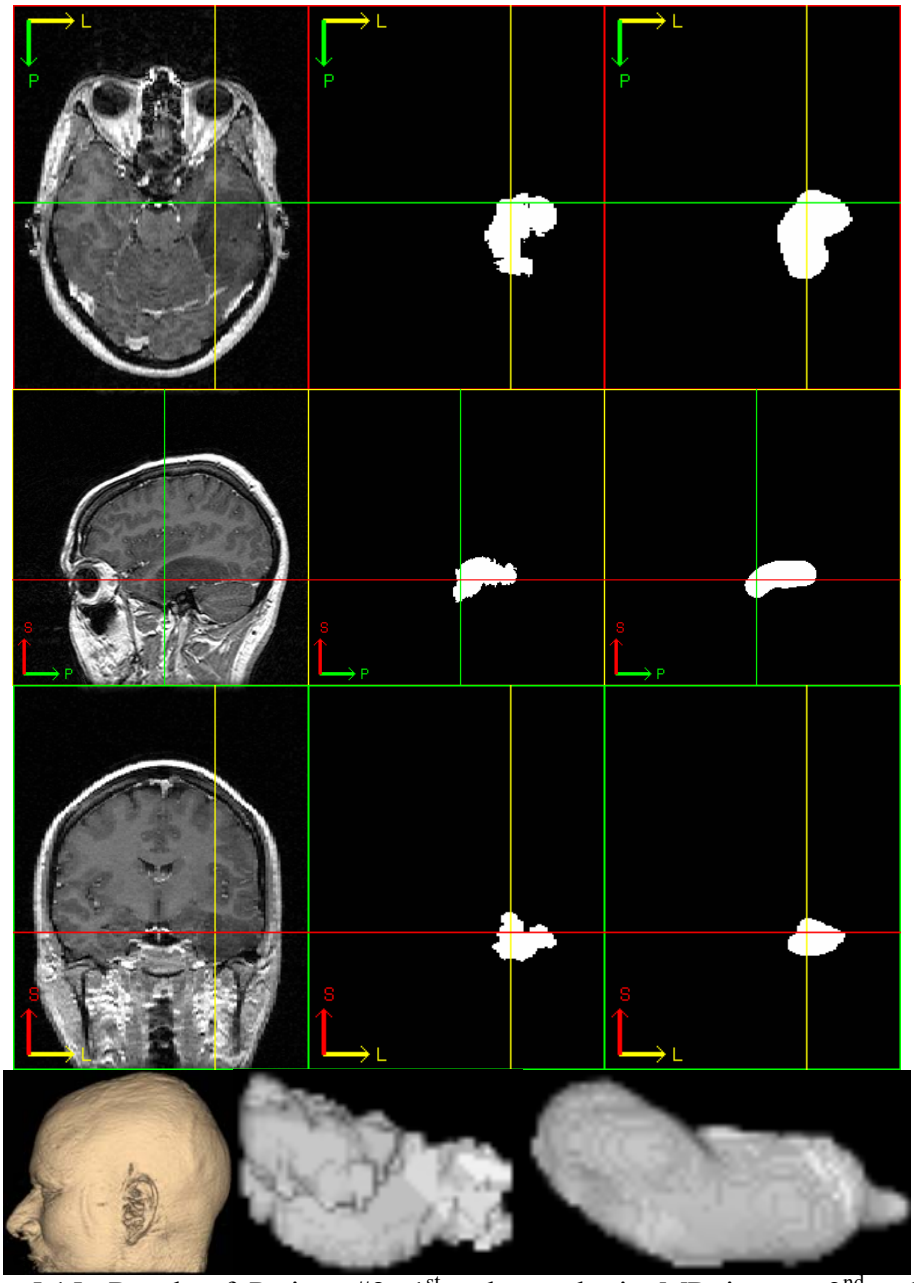


Figure 5.15: Result of Patient #8. 1st column: brain MR image, 2nd column: ground truth, 3rd column: brain tumor extracted by the proposed method. 1st row: axial view, 2nd row: sagittal view, 3rd row: coronal view, 4th row: volume rendering.

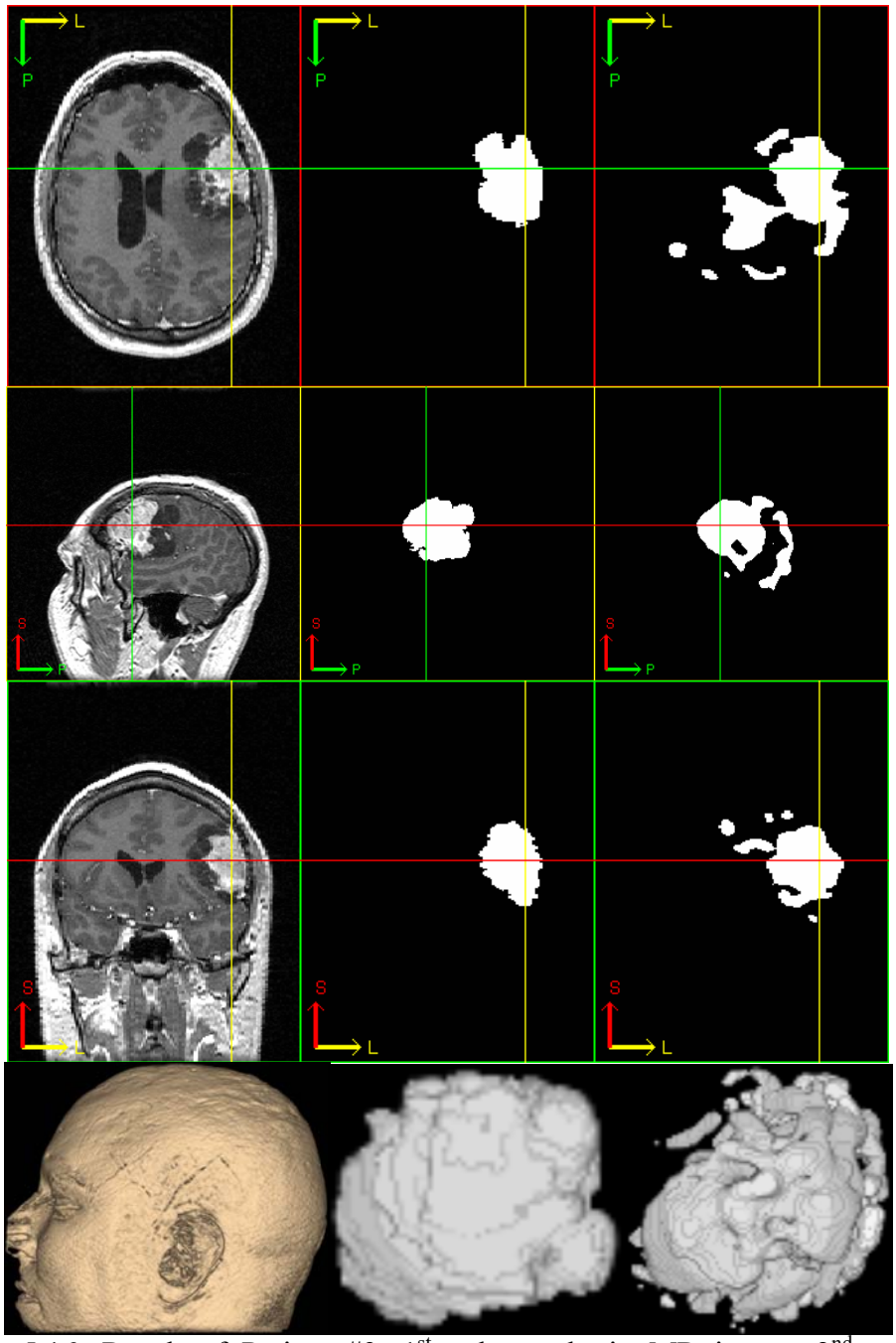


Figure 5.16: Result of Patient #9. 1st column: brain MR image, 2nd column: ground truth, 3rd column: brain tumor extracted by the proposed method. 1st row: axial view, 2nd row: sagittal view, 3rd row: coronal view, 4th row: volume rendering. The tumor region is spongy and largely inhomogeneous so that the segmentation accuracy (0.30) is low.

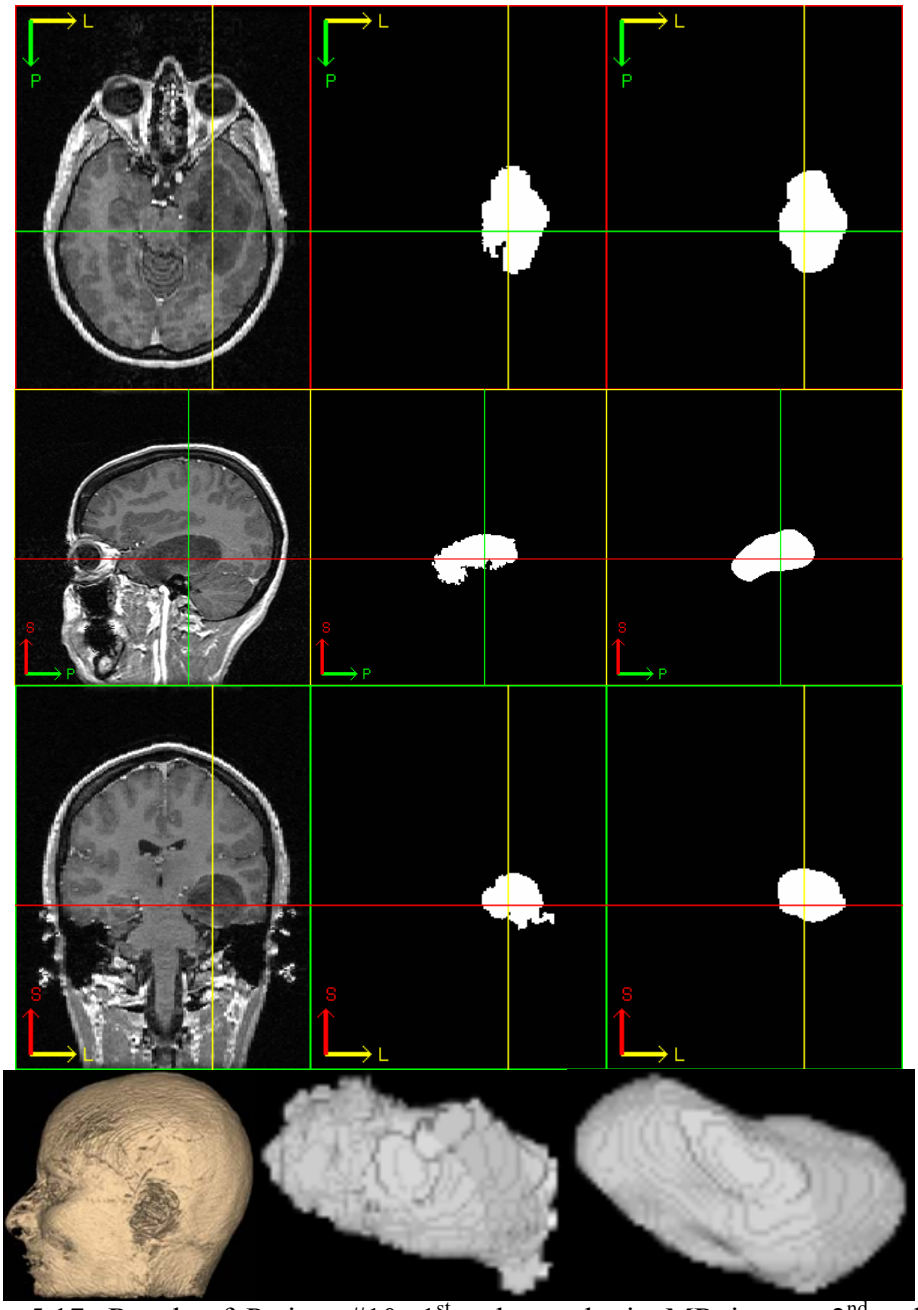


Figure 5.17: Result of Patient #10. 1st column: brain MR image, 2nd column: ground truth, 3rd column: brain tumor extracted by the proposed method. 1st row: axial view, 2nd row: sagittal view, 3rd row: coronal view, 4th row: volume rendering.

5.4 Conclusions

A new brain tumor segmentation method was presented and validated in this paper. The method is able to segment brain tumors fully automatically by taking advantages of the proposed Normalized Gaussian Mixture Model (NGMM) and 3D Flexible Vector Flow (FVF) algorithms. Our technique can be utilized to generate segmented brain tumor images that display clinically important neuroanatomic and neuropathologic information.

In some test cases (e.g., case #1 and #2), the accuracy of our method is satisfactory. However, due to the complexity of automatic brain tumor segmentation and other reasons (such as unavailability of T2 images), the accuracy of our technique is not satisfactory in other test cases (e.g., case #5 and #9). Therefore, without further improvement and validation, this method cannot be directly applied to clinical cases because of the lack of precision.

This technique has five strengths: 1) fully automatic; 2) able to perform in 3D; 3) requires only one MR modality (T1); 4) able to segment multiple types of brain tumor; 5) work in near real-time (minutes). This method can be implemented as a real-time (seconds) program with more efficient programming languages such as C or C++ (which are at least ten times faster than current implementation with MATLAB).

The major limitation of this method is that the accuracy of segmentation of astrocytoma is low. There are three reasons responsible for the low segmentation accuracy. First, we use software MIPAV [22, 23] to register the atlas (average of healthy brains) and brain tumor MR images. It is well known that registration of MR images with and without tumor is generally very difficult. The registration algorithm of MIPAV works well for meningioma and low grade glioma but not for astrocytoma. Second, it is very challenging to segment brain tumors without shape information and relative position to other structures. Last but not the least, we do not distinguish brain tumor and surrounding edema. In future work, we plan to investigate MR image registration algorithms, incorporate shape information and relative position to other structures, and distinguish brain tumor and surrounding edema, to improve the accuracy.

Bibliography

[1] K. Brindle, "New approaches for imaging tumour responses to treatment", *Nature Reviews Cancer* vol 8(2), pp. 94-107, Feb. 2008.

[2] L. M. Fletcher-Heath, L. O. Hall, D. B. Goldgof, and F. R. Murtagh, "Automatic segmentation of non-enhancing brain tumors in magnetic resonance images", *Artif. Intell. Med.*, vol. 21, pp. 43–63, 2001.

[3] Y. Zhu and H. Yan, "Computerized tumor boundary detection using a hopfield neural network", *IEEE Trans. Med. Imag.*, vol. 16(1), pp.55–67, Feb. 1997.

[4] M. C. Clark, L. O. Hall, D. B. Goldgof, R. Velthuizen, R. Murtagh, and M. S. Silbiger, "Automatic tumor segmentation using knowledge-based techniques", *IEEE Trans. Med. Imaging*, vol. 17(2), pp. 187–201, Apr. 1998.

[5] J. G. Smirniotopoulos, "The new WHO classification of brain tumors", *Neuroimaging Clinics North America*, vol. 9, no. 4, pp. 595–613, Nov. 1999.

[6] M. R. Patel and V. Tse, "Diagnosis and staging of brain tumors", *Seminars Roentgenology*, vol. 39, no. 3, pp. 347–360, 2004.

[7] S. Vinitski, C. F. Gonzalez, R. Knobler, D. Andrews, T. Iwanaga, and M. Curtis, "Fast tissue segmentation based on a 4D feature map in characterization of intracranial lesions", *J. Magn. Reson. Imag.*, vol. 9, no. 6, pp. 768–776, 1999.

[8] S. K. Warfield, M. Kaus, F. A. Jolesz, and R. Kikinis, Adaptive, "Template Moderated, Spatially Varying Statistical Classification", *Med. Image Anal.*, Vol 4(1): 43-55, 2000.

[9] M. Kaus, S. K. Warfield, A. Nabavi, P. M. Black, F. A. Jolesz, and R. Kikinis. "Automated Segmentation of MRI of Brain Tumors", *Radiology*. 218(2):586-91, 2001.

[10] M. Prastawa, E. Bullitt, N. Bullitt, K. V. Leemput, and G. Gerig, "Automatic brain tumor segmentation by subject specific modification of atlas priors", *Acad. Radiol.*, vol. 10, pp. 1341–1348, Dec. 2003.

[11] M. Prastawa, E. Bullitt, S. Ho, and G. Gerig, "A brain tumor segmentation framework based on outlier detection", *Med. Image Anal.*, vol. 8, no. 3, pp. 275–283, Sep. 2004.

[12] J. Zhang, K. Ma, M. H. Er, and V. Chong, "Tumor segmentation from magnetic resonance imaging by learning via one-class support vector machine", *International Workshop on Advanced Image Technology (IWAIT)*, 2004.

[13] J. Liu, J. K. Udupa, D. Odhner, D. Hackney, and G. Moonis, "A system for brain tumor volume estimation via MR imaging and fuzzy connectedness", *Comput. Med. Imaging Graphics*, vol. 29, no. 1, pp. 21–34, 2005.

[14] J. Wang, Q. Li, T. Hirai, S. Katsuragawa, F. Li, and K. Doi, "An Accurate Segmentation Method for Volumetry of Brain Tumor in 3D MRI", *Proc. of SPIE*, Vol. 6914 69144D-1, 2008.

[15] S. Dube, J. J. Corso, A. Yuille, T. F. Cloughesy, S. El-Saden, and U. Sinha, "Hierarchical Segmentation of Malignant Gliomas via Integrated Contextual Filter Response", *SPIE Medical Imaging Symposium*, vol. 6914 (3), pp. 69143Y.1-69143Y.6, 2008.

[16] J. J. Corso, E. Sharon, S. Dube, S. El-Saden, U. Sinha, and A. Yuille, "Efficient Multilevel Brain Tumor Segmentation With Integrated Bayesian Model Classification", *IEEE TRANSACTIONS ON MEDICAL IMAGING*, VOL. 27, NO. 5, MAY 2008.

[17] M. Wels, G. Carneiro, A. Aplas, M. Huber, J. Hornegger, and D. Comaniciu, "A Discriminative Model-Constrained Graph Cuts Approach to Fully Automated Pediatric Brain Tumor Segmentation in 3-D MRI", *MICCAI*, Part I, LNCS 5241, pp. 67–75, 2008.

[18] W. E. Phillips, R. P. Velthuizen, S. Phupanich, L. O. Hall, L. P. Clarke, and M. L. Silbiger, "Application of fuzzy c-means segmentation technique for tissue differentiation in MR images of a hemorrhagic glioblastoma multiforme", *Magn. Reson. Imag.*, vol. 13, no. 2, pp. 277–290, 1995.

[19] N. B. Karayiannis and P. I. Pai, "Segmentation of magnetic resonance images using fuzzy algorithms for learning vector quantization", *IEEE Trans. Med. Imaging*, vol 18(2), pp. 172–180, Feb. 1999.

[20] S. Ho, E. Bullitt, and G. Gerig, "Level set evolution with region competition: Automatic 3-D segmentation of brain tumors", in *Proc. Int. Conf. Pattern Recognition*, vol. I, pp. 532–535, 2002.

[21] http://www.spl.harvard.edu/publications/item/view/1180, retrieved in March 2009.

[22] P. L. Bazin, D. L. Pham, W. Gandler, and M. McAuliffe, "Free Software Tools for Atlas-based Volumetric Neuroimage Analysis", *Progress in Biomedical Optics and Imaging - Proceedings of SPIE* 5747 (III), art. no. 212, pp. 1824-1833.

[23] http://mipav.cit.nih.gov/, retrieved in April 2009.

[24] http://www.loni.ucla.edu/ICBM/Downloads/Downloads_452T1.shtml, retrieved in March 2009.

[25] http://www.loni.ucla.edu/ICBM/Downloads/Downloads_ICBMprobabilistic.shtml, retrieved in March 2009.

[26] **T. Wang**, I. Cheng and A. Basu, "Fluid Vector Flow and Applications in Brain Tumor Segmentation", *IEEE Trans. on Biomedical Engineering*, Vol. 56(3), pages 781-789, 2009.

[27] J. Talairach and P. Tournoux, *Co-Planar Stereotaxic Atlas of the Human Brain*, Thieme, 1988.

[28] J. Lancaster, J. Summerlin, L. Rainey, C. Freitas, and P. Fox, "The talairach daemon, a database server for talairach atlas labels", *Neuroimage* **5**(4), 1997.

[29] W. L. Nowinski and D. Belov, "The cerefy neuroradiology atlas: a talairach-tournoux atlas-based tool for analysis of neuroimages available over the internet", *NeuroImage* 20, pp. 50–57, 2003.

[30] http://www.nitrc.org/, retrieved in April 2009.

[31] http://www.slicer.org, retrieved in March 2009.

[32] http://www.fmrib.ox.ac.uk/fsl, retrieved in March 2009.

[33] http://crl.med.harvard.edu/software/STAPLE, retrieved in March 2009.

[34] http://bioimagesuite.org, retrieved in March 2009.

[35] www.itksnap.org, retrieved in March 2009.

[36] http://www-sop.inria.fr/asclepios, retrieved in March 2009.

[37] http://www.loni.ucla.edu, retrieved in March 2009.

[38] M. Chen, T. Kanade, D. Pomerleau, J. Schneider, "3-D Deformable Registration of Medical Images Using a Statistical Atlas", *MICCAI*, LNCS 1679, pp. 621-630, 1999.

[39] A. P. Dempster, N. M. Laird, D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm", *J. of the Royal Stat. Society*. Series B (Methodological) 39 (1): 1–38, 1977.

[40] R. Neal, G. E. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants". *Learning in Graphical Models* (Cambridge, MA: MIT Press): 355–368. ISBN 0262600323, 1999.

[41] T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning*. New York: Springer. pp. 236–243. ISBN 0-387-95284-5, 2001.

[42] M. Jamshidian, R. I. Jennrich, "Acceleration of the EM Algorithm by using Quasi-Newton Methods". *Journal of the Royal Statistical Society*: Series B (Statistical Methodology) 59 (2): 569–587, 1997.

[43] J. B. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations", in *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*. 1: 281–297, University of California Press, 1967.

[44] P. Brucker, "On the complexity of clustering problems", *Optimization and operations research*, pp. 45–54, Lecture Notes in Economics and Mathematical Systems 157, Berlin: Springer, 1978.

[45] P. Drineas, A. Frieze, R. Kannan, S. Vempala, V. Vinay, "Clustering large graphs via the singular value decomposition", *Machine Learning* 56 (1): 9–33, 2004.

[46] M. Kass, A. Witkin, and D. Terzopoulus, "Snakes: Active contour model", *Intl. J. of Computer Vision*, vol. 1(4), pp. 321-331, 1988.

[47] R. Whitaker. "A level-set approach to 3D reconstruction from range data". *Int. J. of Comp. Vision*, Vol. 29(3), pp. 203–231, 1998.

[48] S.Osher and J.Sethian, "Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations". *J. of Computational Physics*, Vol 79(1), pp. 12–49, 1988.

[49] D.Adalstein, and J.A.Sethian, "A fast level set method for propagating interfaces". *Journal of Computational Physics*, Vol 118(2), pp. 269–277, 1995.

[50] J.A. Sethian, *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Material Sciences*, Cambridge University Press, 1996.

[51] J. C. Masse and J. F Plante, "A Monte Carlo study of the accuracy and robustness of ten bivariate location estimators", *Comput. Statistics & Data Analysis*, vol. 42, pp. 1-26, 2003.

[52] **T. Wang** and I. Cheng, "Generation of Unit-width curve skeletons based on Valence Driven Spatial Median (VDSM)", *Int. Sym. on Visual Comput.*, LNCS 5358, pp. 1061-1070, 2008.

[53] T. Chan and L. Vese. "Active contours without edges". *IEEE Transactions on Image Processing*, 10(2):266–277, February 2001.

[54] C. Xu and J. L. Prince, "Snakes, shapes, and gradient vector flow", *IEEE Trans. on Image Processing*, pp. 359-369, 1998.

[55] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, USA: Academic Press, p. 366, 1999.

[56] D. Cobzas, N. Birkbeck, M. Schmidt, M. Jagersand, and A. Murtha, "3D variational brain tumor segmentation using a high dimensional feature set", Mathematical Methods in Biomedical Image Analysis (MMBIA 2007), in conjunction with ICCV.

Chapter 6 Conclusion

This thesis addresses two important and interesting research problems for object representation and analysis: skeletonization and segmentation. Skeletonization algorithm extracts the "centre-lines" of an object and uses them to efficiently represent the object. It has many applications such as object matching and retrieval. Segmentation algorithm locates the target object or Region Of Interest (ROI) from images. It has many applications such as medical image analysis.

These two research problems are not independent but related to each other. Our 3D skeletonization algorithm [5] takes a 3D binary image (equilateral 3D grid) and generates unit-width curve skeletons. A 3D binary image can be converted from a 3D mesh by voxelization. The key idea of the algorithm is volumetric processing, which is also the key idea of the 3D segmentation algorithm [2]. Therefore, algorithms and methods developed for one problem may be utilized for solving another. For instance, the Valence Normalized Spatial Median (VNSM) algorithm [5] that initially proposed to generate unit-width curve skeleton was used in the segmentation algorithm [1] to determine the centre of a region.

6.1 Skeletonization

Skeletonization algorithms should have some desired properties such as centeredness, connectivity preservation, robustness to noise, thinness, etc.

The well-known Ma and Sonka's 3D skeletonization algorithm [9], if not the only, is one of the first fully parallel 3D thinning algorithms. It has higher efficiency than most other skeletonization methods. However, we found it cannot preserve connectivity. A solution [3] was given in first skeletonization study of this thesis for connectivity preservation.

However, neither the original algorithm [9] nor the modified version [3] can guarantee to generate unit-width curve skeleton, which is highly desirable by many applications such as the example of 3D matching and retrieval presented in [4]. We modified the spatial median method by adding the valence constraint to

obtain unit-width (thinness) curve skeleton with better visual centeredness than the spatial median approach. Therefore, two important properties are addressed in the second skeletonization study [5]: thinness and visual centeredness.

For our application [4], *i.e.*, 3D matching and retrieval, thinness is essential because the chain code generation algorithm only works on unit-width lines (curve skeletons). A new metric, Thinness Metric (*TM*) was proposed and used to measure the improvement in thinness. *TM* is a normalized metric with values between zero and one. If *TM* is zero, the skeleton is unit-width thin; otherwise, it is not thin. If *TM* is one, all the points on the skeleton are crowded points. The TM score of the proposed method is always zero, which indicates the curve skeletons generated by our method are always thin.

Centeredness, on the other hand, has always been a controversial topic [10]. First of all, centeredness is only well-defined in symmetric models. Secondly, centeredness is conflicting with robustness to noise and smoothness. Figures 6.1 and 6.2 illustrate these two concerns.

As shown in Figure 6.2, the curve skeleton is very sensitive to noise. Centeredness constrains the curve skeletons to the medial lines, which are extremely sensitive to boundary perturbations.

In real world, given that few models are perfectly symmetric and noise always exists, centeredness is not often required or desired.

However, skeletons are often defined as “center lines” or “medial axes”. Therefore, visual centeredness, *i.e.*, skeletons should not lie on the boundary of the object, is often desirable. However, the centre point estimated by existing centre estimators such as arithmetic mean or spatial median may lie outside or on the boundary of a concave region. In the second skeletonization study, a so-called Valence Normalized Spatial Median (VNSM) algorithm is proposed to locate the centre point of a region. The key concept is to normalize the spatial median by the valence of a point (vertex) so that the boundary points which have smaller valences than interior points will not be chose as the centre of a region, no matter the region is convex or concave.

There are some other application specific properties, such as smoothness, for skeletonization algorithm. Smoothness is sometimes required in the application of virtual navigation, which uses the curve skeleton as a camera translation path. This path should be as smooth as possible to avoid abrupt changes in the displayed image. However, for our application, 3D matching and retrieval, smoothness is not required or desired.

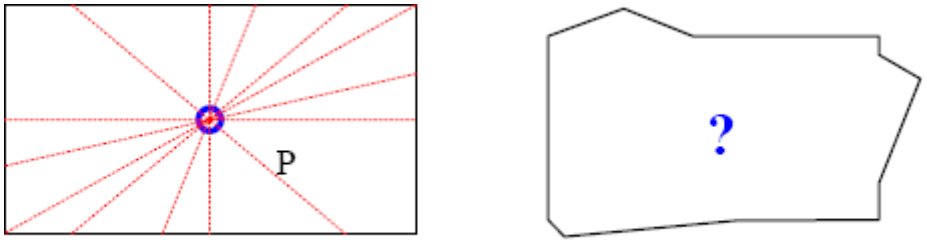


Figure 6.1: Centeredness of an isolated point in 2D. (a) a point perfectly centered within a symmetric figure is at equal distance from the boundary of the figure in all directions. b) a point cannot be perfectly centered within a non-symmetric figure. (Images courtesy of Cornea [10].)

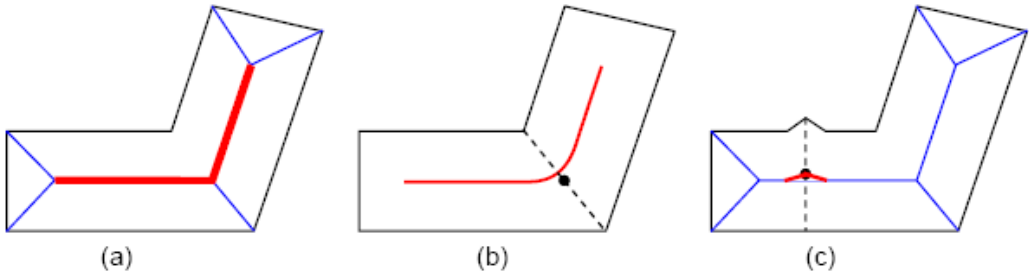


Figure 6.2: Centeredness vs. robustness and smoothness. A curve-skeleton (in red) as a subset of the medial axis/surface is perfectly centered within the figure (a). A smoother curve skeleton, which is not perfectly centered in the "elbow" region (b). A perfectly centered skeleton cannot remain smooth in the presence of noise (c). (Images courtesy of Cornea [10].)

The skeletonization research has gained significant momentum [3, 5, 9, 10] in recent years. Many research works have emerged to apply skeletons in a number of applications [4, 6-10]. However, in each specific application, the set of desired

properties must be carefully defined to choose a proper skeletonization method. Otherwise, the expected results may not be achieved.

Traditionally, skeletonization algorithms focus on extracting one skeleton from one object. A new and innovative way to generate skeleton is to consider "Groupwise" skeletonization [13]. However, a detailed discussion on that topic is beyond the scope of this thesis.

6.2 Segmentation

This thesis also presents two studies in segmentation that advanced the state-of-the-art research. The first segmentation study [2] presents a new approach named Flexible Vector Flow (FVF) to address a few problems of other active contour models such as insufficient capture range and poor convergence for concavities. This approach was applied to brain tumor segmentation in two dimensional (2D) space. The second segmentation study [1] extends the 2D FVF algorithm to three-dimension (3D) and utilizes it to automatically segment brain tumors in 3D.

Chan and Vese [11] pointed out the evolution of active contour (2D) or active surface (3D) should not always rely on gradient (2D) or surface normal (3D). This inspired us to explore new component to drive the evolution of active contours and active surfaces, which leads to the new FVF algorithm. The basic idea of FVF is to add a directional component to the external force while keep the gradient or normal component. This idea is straightforward and the experiments show that it is also efficient and reasonably accurate.

The segmentation studies used human brains to explore new brain tumor segmentation approaches. Another interesting and relevant research topic is human brain segmentation. The purpose of human brain segmentation is to segment human brains into a number of Regions Of Interests (ROIs). The ROIs can be further processed for diagnosis support system or construction of brain atlases. Gousias *et al.* [12] proposed an automatic brain segmentation method to segment brain MRIs of 2-year-olds into 83 ROIs (shown in Figure 6.3). This work may lead to the construction of brain atlases for infants and children. As we have discussed in Chapter 5, brain atlases played an important role in our brain tumor

segmentation method. However, only adult brain atlases are publicly available currently. Once the infants and children brain atlases become available, our brain tumor method [1] can be adopted and applied to detect brain abnormalities such as Intra-Ventricular Hemorrhage (IVH), which is often found in pre-term babies.

As discussed in Chapter 3, unit-width curve skeletons have been utilized for 3D segmentation. A possible future research is to perform human brain segmentation by taking advantage of our skeletonization algorithm [3, 5]. Some unit-width curve skeletons of human brains are shown in Figure 6.4.

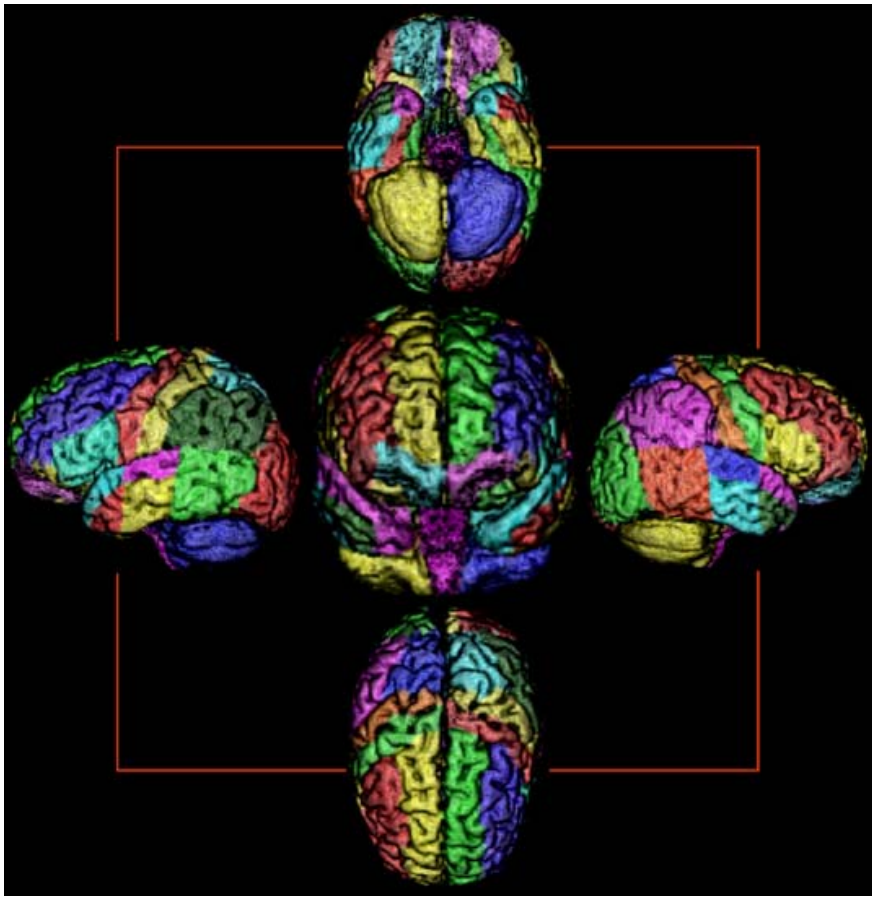


Figure 6.3: Example of brain segmentation. Different ROIs are colour-coded [14].

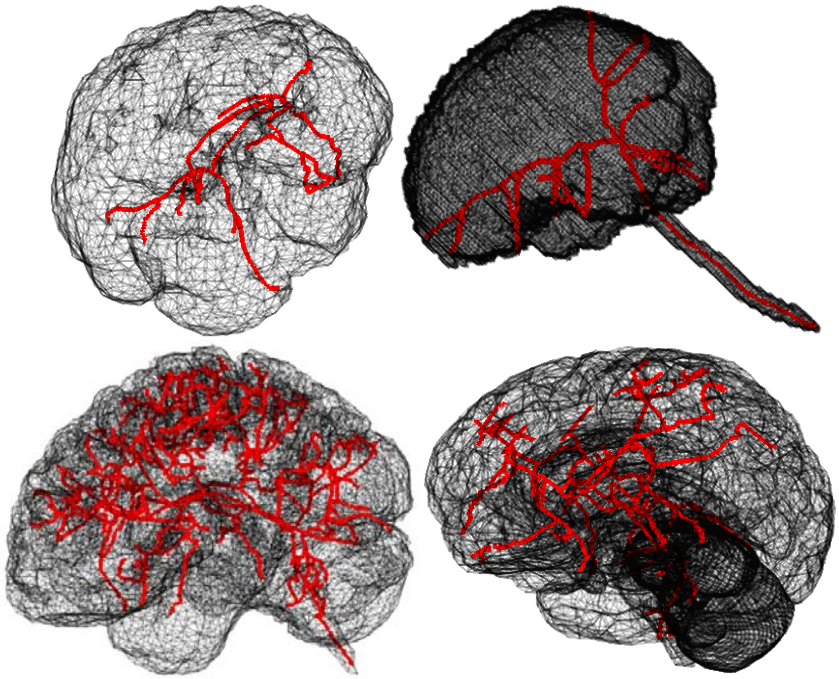


Figure 6.4: Skeletons of human brains.

Brain tumor segmentation has been investigated intensively in the last a few decades. Unfortunately, no method has been clinically proved. Therefore, no method can be directly applied to clinical cases. Further improvement and validation must be performed for existed and emerging methods.

6.3 Publications

This thesis is mainly based on the following publications.

In Preparation

1. **T. Wang**, I. Cheng and A. Basu, Fully Automatic Brain Tumor Segmentation using a Normalized Gaussian Bayesian Classifier and 3D Flexible Vector Flow, submitted for publication.

Refereed Journal Papers

2. **T. Wang**, I. Cheng and A. Basu, Fluid Vector Flow and Applications in Brain Tumor Segmentation, IEEE Transactions on Biomedical Engineering, Vol. 56(3), pages 781-789, 2009.
3. **T. Wang** and A. Basu, A note on "A fully parallel 3D thinning algorithm and its applications", Vol. 28(4), pages 501-506, Pattern Recognition Letters, 2007.

Refereed Conference Papers

4. **T. Wang**, I. Cheng, V. Lopez, E. Bribiesca and A. Basu, Valence Normalized Spatial Median for Skeletonization and Matching, Search in 3D and Video workshop (S3DV), in conjunction with IEEE International Conference on Computer Vision (ICCV) 2009.
5. **T. Wang** and I. Cheng, Generation of Unit-width curve skeletons based on Valence Driven Spatial Median (VDSM), International Symposium on Visual Computing (ISVC), LNCS 5358, pages 1061-1070, 2008.
6. **T. Wang** and A. Basu, Iterative Estimation of 3D Transformations for Object Alignment, International Symposium on Visual Computing (ISVC), LNCS 4291, pages 212-221, 2006.
7. **T. Wang** and A. Basu, Automatic Estimation of 3D Transformations using Skeletons for Object Alignment, IAPR/IEEE International Conference on Pattern Recognition (ICPR), pages 51-54, 2006.

Refereed Poster Presentation

8. V. Lopez, I. Cheng, E. Bribiesca, **T. Wang** and A. Basu, Twist-and-Stretch: A Shape Dissimilarity Measure based on 3D Chain Codes, ACM SIGGRAPH Asia Research Poster, 2008.

Bibliography

[1] **T. Wang**, I. Cheng and A. Basu, "Fully Automatic Brain Tumor Segmentation using a Normalized Gaussian Bayesian Classifier and 3D Fluid Vector Flow", *submitted for publication*.

[2] **T. Wang**, I. Cheng and A. Basu, "Fluid Vector Flow and Applications in Brain Tumor Segmentation", *IEEE Transactions on Biomedical Engineering*, Vol. 56(3), pages 781-789, 2009.

[3] **T. Wang** and A. Basu, "A note on 'A fully parallel 3D thinning algorithm and its applications' ", Vol. 28(4), pages 501-506, *Pattern Recognition Letters*, 2007.

[4] **T. Wang**, I. Cheng, V. Lopez, E. Bribiesca and A. Basu, "Valence Normalized Spatial Median for Skeletonization and Matching", *Search in 3D and Video workshop (S3DV), in conjunction with IEEE International Conference on Computer Vision (ICCV)*, 2009.

[5] **T. Wang** and I. Cheng, "Generation of Unit-width curve skeletons based on Valence Driven Spatial Median (VDSM)", *International Symposium on Visual Computing (ISVC), LNCS 5358*, pages 1061-1070, 2008.

[6] **T. Wang** and A. Basu, "Iterative Estimation of 3D Transformations for Object Alignment", *International Symposium on Visual Computing (ISVC), LNCS 4291*, pages 212-221, 2006.

[7] **T. Wang** and A. Basu, "Automatic Estimation of 3D Transformations using Skeletons for Object Alignment", *IAPR/IEEE International Conference on Pattern Recognition (ICPR)*, pages 51-54, 2006.

[8] V. Lopez, I. Cheng, E. Bribiesca, **T. Wang** and A. Basu, "Twist-and-Stretch: A Shape Dissimilarity Measure based on 3D Chain Codes", *ACM SIGGRAPH Asia Research Poster*, 2008.

[9] C. M. Ma, M. Sonka, "A fully parallel 3D thinning algorithm and its applications", *Computer Vision and Image Understanding*, vol. 64 (3), pp. 420-433, 1996.

[10] N. D. Cornea. "Curve-Skeletons: Properties, Computation And Applications", *Ph.D. Thesis*, *Rutgers University*, 2007.

[11] T. Chan and L. Vese. "Active contours without edges". *IEEE Transactions on Image Processing*, vol. 10(2), pp. 266–277, 2001.

[12] I. S. Gousias, D. Rueckert, R. A. Heckemann,L. E. Dyet, J. P. Boardman, A. D. Edwards, and A. Hammers, "Automatic segmentation of brain MRIs of 2-year-olds into 83 regions of interest", *NeuroImage*, vol. 40 pp.672–684, 2008.

[13] A. Ward and G. Hamarneh. "GMAT: The Groupwise Medial Axis Transform for Fuzzy Skeletonization and Intelligent Pruning". IEEE TPAMI 2009.