Efficient Geometrical Potential Force Computation for Deformable Model Segmentation

Igor Sazonov[1], Xianghua Xie[2], and Pemural Nithiarasu[1]

[1]College of Engineering, Swansea University, Singleton Park, Swansea, UK SA2 8PP
[2]Department of Computer Science, Swansea University, Singleton Park, Swansea, UK SA2 8PP
{i.sazonov,csehab,x.xie,i.sazonov,p.nithiarasu}@swansea.ac.uk

Abstract. Segmentation in high dimensional space, e.g. 4D, often requires decomposition of the space and sequential data process, for instance space followed by time. In [1], the authors presented a deformable model that can be generalized into arbitrary dimensions. However, its direct implementation is computationally prohibitive. The more efficient method proposed by the same authors has significant overhead on computer memory, which is not desirable for high dimensional data processing. In this work, we propose a novel approach to formulate the computation to achieve memory efficiency, as well as improving computational efficiency. Numerical studies on synthetic data and preliminary results on real world data suggest that the proposed method has a great potential in biomedical applications where data is often inherently high dimensional.

1 Introduction

Among many others, deformable modeling is a popular approach to image segmentation, e.g. [2–4]. Conventional techniques suffer from weak edge, image noise and convergence issues. For instance, in [2] a constant pressure force is necessary in order to improve its capture range, resulting in monotonic expanding or shrinking of the mode that is problematic. There have been numerous work reported in the literature to improve the performance of both image gradient based methods, such as [5–7], and region based approaches, e.g. [8]. In [1], Yeo *et al.* proposed a 3D deformable model that is based on a hypothesised geometrical interactions between image gradient vectors and embedding level set surface normal vectors. It is shown that the geometrical potential force (GPF) is robust towards noise interference, weak edges, and exhibits invariant convergence capabilities such that the model can be initialized across object boundary and converge to deep concavities and propagate through narrow passages to recover complex geometries, that are conventionally difficult for image gradient based deformable modeling techniques. The authors also showed its theoretical relationship to the 2D Magnetostatic Active Contour (MAC) model [7] , which

is inspired by a physical analogy. The MAC model can be considered a special case of GPF in 2D, whereas GPF can be more conveniently extended to higher dimensional applications.

The computation of the GPF comprises two stages. At the first stage, the so-called geometrical potential (GP) $G(x, y, z)$ is computed through the convolution of the image gradient and the kernel $\mathbf{K}$:

$$G(\mathbf{x}) = \sum_{\mathbf{x}' \in \Omega} \nabla I(\mathbf{x}') \cdot \mathbf{K}(\mathbf{x} - \mathbf{x}'), \qquad \mathbf{K}(\mathbf{x}) = \begin{cases} \mathbf{x} / \|\mathbf{x}\|^{n+1}, & \mathbf{x} \neq \mathbf{0} \\ \mathbf{0}, & \mathbf{x} = \mathbf{0} \end{cases} \tag{1}$$

where $\mathbf{x} = [x, y, z]^T$ is the vector of coordinates of the image grid-points (voxel centres), $I(\mathbf{x})$ is the greyscale image, ∇I is its gradient, Ω is the image domain, dot denotes the scalar product of two vector functions (∇I and kernel $\mathbf{K}(\mathbf{x})$), and n is the image dimension ($n = 3$ for 3D images).

At the second stage, the derived geometrical potential is then integrated into the deformable surface evolution under the level set framework. The active surface, $S(t)$, is embedded in the level set function, $\Phi(t, \mathbf{x})$: $S(t) = \{\mathbf{x}, \Phi(t, \mathbf{x}) = 0\}$, and its deformation is achieved by solving the following PDE proposed and developed in [9–11] and related to the energy minimization approach:

$$\partial\Phi/\partial t = \alpha\, g\kappa\|\nabla\Phi\| - (1-\alpha)\, \mathbf{F}\nabla\Phi \tag{2}$$

where α is a weighting parameter, $g(\mathbf{x}) = 1/(1+\|\nabla I\|^2)$ is the stopping function, $\kappa(t, \mathbf{x}) = \nabla\hat{\mathbf{n}}$ denotes the curvature of isosurfaces of Φ, $\hat{\mathbf{n}}(t, \mathbf{x})$ is the unit vector normal to isosurfaces of Φ, and $\mathbf{F}(t, \mathbf{x}) = G\,\hat{\mathbf{n}}$ is the GPF that acts as the external force.

Direct calculation of the geometrical potential G is computationally expensive, particularly in 3D. However, Eq. (1) can be computed as a convolution of two functions. Hence a natural approach is to apply the fast Fourier transform (FFT) to compute the convolution, which is described in [1].

However, a significant drawback of using the FFT based computation as proposed in [1] is that it requires lots of computer memory for a large number of intermediate arrays of the same size as the initial image I. That is, it needs to compute and store 3 components of the image gradient $\nabla I = [I_x, I_y, I_z]^T$ and twice more for the real and imaginary part of their Fourier image, also 3 components of the kernel $\mathbf{K}$ and twice more for the Fourier image. Thus, it requires about 20 times more than the direct method, which can be problematic when dealing with volumetric data or extending this method to 4D, i.e. dynamic volumetric data. Dedicated memory management may become necessary and even crucial. Memory economic and computationally efficient method to evaluate the GP is thus desirable.

In this paper, we propose to compute spectrum of the kernel by an analytical formula so that there is no need to store components of the vector kernel and the real or imaginary part of its spectrum. We also change the vector form of the integrand into a scalar form to achieve further efficiency. The proposed methods are valuated on both numerical examples and real world 3D data.

2 Analytical Formula for Kernel's Spectrum

One of the possible approaches to reduce memory usage is to use an analytical formula for the kernel spatial spectrum rather than kernel's formula (1) in the x-space. To derive an analytical formula for the kernel Fourier image, it is useful to consider the computation of G in the continuous infinite 3D Euclidian space. In this case the kernel should be described by a generalized function (distribution) (see, e.g. [12]):

$$G(\mathbf{x}) = \int_{\mathbf{x}' \in \mathbb{R}^3} \nabla I(\mathbf{x}') \cdot \mathbf{K}(\mathbf{x} - \mathbf{x}') \, \mathrm{d}^3\mathbf{x}, \qquad \mathbf{K}(\mathbf{x}) = P.V. \frac{\mathbf{x}}{\|\mathbf{x}\|^{n+1}} \tag{3}$$

where $P.V.$ denotes *principal value*, i.e. integral in (3) diverging when $\mathbf{x}' \to \mathbf{x}$, should be treated as the limit

$$G(\mathbf{x}) = \lim_{\varepsilon \to 0^+} \int_{\|\mathbf{x}' - \mathbf{x}\| > \varepsilon} \nabla I(\mathbf{x}') \cdot \frac{\mathbf{x} - \mathbf{x}'}{\|\mathbf{x} - \mathbf{x}'\|^{n+1}} \, \mathrm{d}^3\mathbf{x}' \tag{4}$$

Performing the Fourier transform

$$\tilde{\mathbf{K}}(\mathbf{k}) = \mathcal{F}[\mathbf{K}](\mathbf{k}) = \int \mathbf{K}(\mathbf{x}) e^{\mathrm{i}\mathbf{k}\mathbf{x}} \, \mathrm{d}^3\mathbf{x}, \qquad \mathrm{i} = \sqrt{-1} \tag{5}$$

we can show that that the spectrum depends only on direction of wavevector $\mathbf{k}$ and is independent of its magnitude

$$\tilde{\mathbf{K}}(\mathbf{k}) = -\mathrm{i}\pi^2 \frac{\mathbf{k}}{\|\mathbf{k}\|}. \tag{6}$$

Comparing spectrum $\tilde{\mathbf{K}}(\mathbf{k})$ computed analytically via Eq. (6) and that computed by performing FFT for the kernel calculated in the x-space by (1) (see Figure 1(left)), we see that near the origin they have close values. However, the spectrum computed via the FFT decays when any component of the wavevector grows. Moreover, it vanishes when any component of the wavevector reaches its maximum value which is determined by the grid size in the correspondent direction: $k_{i,\max} = \pi / h_i$ where h_1, h_2, h_3 are voxel sizes in x,y,z direction, respectively. Therefore, to obtain the G-function close to that computed by FFT based method, spectrum (6) should be multiplied by a function $f(\mathbf{k})$ which equals 1 in the origin and smoothly decays when $k_i \to k_{i,\max}$. As numerical computation shown later, a good approximation of a 3D spectrum can be formulated as

$$\tilde{\mathbf{K}}(\mathbf{k}) = \mathrm{i}\pi^2 \frac{\mathbf{k}}{\|\mathbf{k}\|} f(\mathbf{k}), \qquad f(\mathbf{k}) = (1 - \|\mathbf{k}'\| + V(\mathbf{k})) \tag{7}$$

where

$$V = \frac{(\xi \|\mathbf{k}'\| - 1)^2}{(\xi + \xi \|\mathbf{k}'\| - 2)\,\xi}, \quad \mathbf{k}' = \left[\frac{k_1}{k_{1,\max}}, \frac{k_2}{k_{2,\max}}, \frac{k_3}{k_{3,\max}} \right]^T, \quad \xi = \max_{i=1,2,3} |k'_i|.$$

This makes the computation much more memory economic; however, we still have to compute the FFT for components of ∇I and then multiply every element of the arrays of the kernel spectrum computed directly for every element.

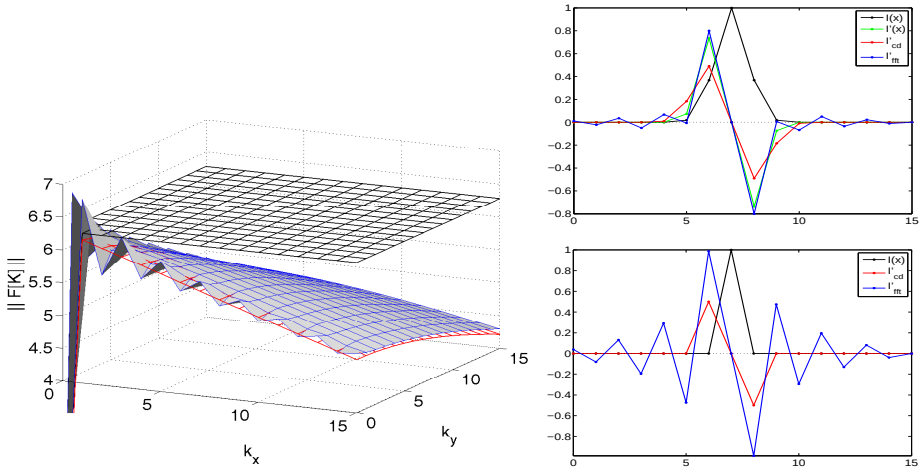

Fig. 1. Left: Absolute value of spectrum $\|\mathrm{Im}\,\tilde{\mathbf{K}}\|$ in the 128×128 2D domain (only first 16 positive wave components are shown) computed via Eq. (6) (black), computed by FFT from kernel evaluated in the x-space (blue) and approximated by Eq. (7) (red). Right: Function $I(x)$ (black), its exact derivative (green, on the top only), its derivative computed by central-difference (red), the same—through FFT (blue). Top: $I(x) = \exp\{-(x-7)^2\}$, bottom: $I(x) = \delta(x-7)$.

3 Use of a Scalar Kernel

Alternatively, we may rearrange the intergrand shown in Eqn. (3) as a product of scalar function and a scalar kernel, instead of a dot product between vectors. To derive the correspondent formula in x-space, we again temporally consider continuous infinite space in which initial integral takes the form given in (3). We then reforumate (3) as

$$G(\mathbf{x}) = \int_{\mathbf{x}' \in \mathbb{R}^3} I(\mathbf{x}') \cdot \nabla \mathbf{K}(\mathbf{x} - \mathbf{x}')\, \mathrm{d}^3\mathbf{x} \tag{8}$$

Thus, we only have to deal with the scalar kernel which is the divergence of the vector kernel $\mathbf{K}$. In the discretized finite domain Eqn. (8) can be approximated as

$$G(\mathbf{x}) = \sum_{\mathbf{x}' \in \Omega} I(\mathbf{x}') \cdot K(\mathbf{x} - \mathbf{x}'), \qquad K(\mathbf{x}) = \nabla \mathbf{K}(\mathbf{x}) \tag{9}$$

where the best way to calculate $\nabla \mathbf{K}$ is to compute vector kernel $\mathbf{K}$ and compute the spatial derivatives by central differences.

4 Combined Approach

However, we may combine the above two methods together to achieve even more efficient computation. In the k-space, the calculation of the geometrical potential

spectrum, $\tilde{G}(\mathbf{k})$, is read as

$$\tilde{G} = \left(\mathrm{i}\mathbf{k}\tilde{I}\right) \cdot \tilde{\mathbf{K}} = \tilde{I}\left(\mathrm{i}\mathbf{k} \cdot \tilde{\mathbf{K}}\right) = \tilde{I}\tilde{K} \tag{10}$$

where $\tilde{K}(\mathbf{k})$ is spectrum of the scalar kernel ($K = \nabla\mathbf{K}$), factor $\mathrm{i}\mathbf{k}$ in the k-space corresponds to the nabla (∇) operator in the infinite continuous x-space. Because we are dealing with discretized images with noise, the computation of the gradient through multiplication by $\mathrm{i}\mathbf{k}$ in the k-space can result in undesired sensitivity to noise. Derivative of a function on a finite uniform grid can be approximated by forward, backward or central differences, but also can be computed through the direct and inverse FFT. The latter method gives very high accuracy for smooth functions (periodic or decaying fast toward the grid borders).

For example, for a 1D function $I(x) = \exp\{-(x-7)^2\}$ set on $x = \{0, 1, \ldots, 15\}$ the error of derivative computed by the central differences is 0.24 whereas the error of derivative computed trough FFT is only 0.08 as seen in Figure 1(right-top). But if the function is not smooth (for example, contains delta-correlated noise) the situation is quite opposite. Consider, as an example, a discrete implementation of Dirac's delta $\delta(x-7)$. Then the derivative computed by the central differences gives a reasonable approximation of $\delta'(x-7)$ with a three point support, whereas the FFT method gives an oscillating result, as depicted in Figure 1(bottom right).

Thus, for image segmentation when noise is common in presence it is more appropriate to use central differences approximation than the FFT method. Fortunately though, the Fourier transform can be used to compute the central differences as well. Recall that in a continuous infinite space the derivative can be expressed as a convolution with $\delta'(x)$

$$\partial I/\partial x = I * (\delta'(x)) = \int_{-\infty}^{+\infty} I(x')\,\delta'(x-x')\,\mathrm{d}x', \tag{11}$$

Computing this derivative by use of the Fourier transform, we should recall its spectrum $\mathcal{F}[\delta'(x)] = \mathrm{i}k$. The central differences can be computed analogously as a convolution with the function $\frac{1}{2h}\big(\delta(x+h) - \delta(x-h)\big)$ having spectrum

$$\mathcal{F}\big[\frac{1}{2h}\big(\delta(x+h) - \delta(x-h)\big)\big] = \frac{\mathrm{i}}{h}\sin(kh). \tag{12}$$

which tends to $\mathrm{i}k$ when $h \to 0$.

In 3D case, spectrum of the gradient operator, $\mathrm{i}\mathbf{k}$, should be substituted by vector

$$\mathbf{g}(\mathbf{k}, \mathbf{h}) = \left[\frac{\mathrm{i}\sin k_1 h_1}{h_1}, \frac{\mathrm{i}\sin k_2 h_2}{h_2}, \frac{\mathrm{i}\sin k_3 h_3}{h_3}\right]^T \tag{13}$$

Then Eqn. (10) should be transformed to

$$\tilde{G} = \tilde{I} \times \left(\mathbf{g}(\mathbf{k}, \mathbf{h}) \cdot \tilde{\mathbf{K}}\right). \tag{14}$$

Thus, for this combined approach we perform FFT on the image $I(\mathbf{x})$; then for every element of the obtained arrays we calculate the scalar kernel spectrum by employing Eqn. (7) for the vector kernel and Eqn. (13) for the modified nabla operator in the k-space; finally we carry out the inverse Fourier transform. It requires memory space 4 times less than that for the initial image $I(\mathbf{x})$: I, $\mathrm{Re}\,\tilde{I}$, $\mathrm{Im}\,\tilde{I}$, G.

5 3D Numerical Examples

To compare different methods for computation of the geometrical potential, an artificial 3D star-like gray-scale image is created shown in Figure 2(right). Its dimension is $64 \times 64 \times 32$ pixel: this relatively small size image is chosen for the sake of convenience in visualizing the results. To understand the noise interference, the 3D data is then added with 5% Gaussian noise.

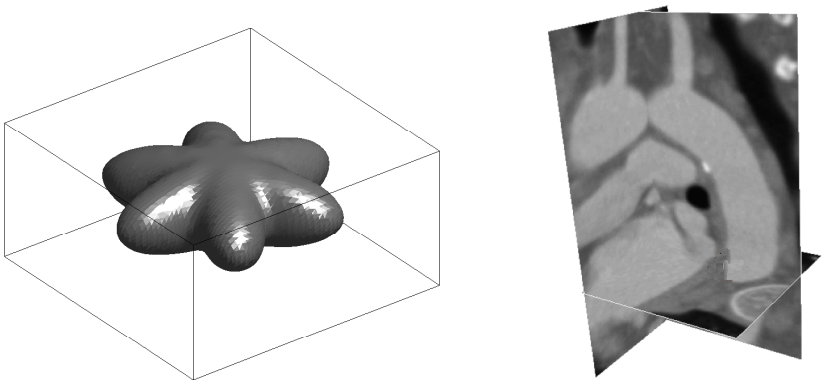

Fig. 2. Left:
Isosurface of the 3D image (without added Gaussian noise). Right: an example of 3D scan of a human aorta.

Figure 3(left) shows the mid-slice along the z-axis. Note, the zero-crossings in the geometrical potential are in effect indicating the locations where the deformable model will converge, since on either side of the zero crossing the deformable model will converge towards zero-crossings. Hence, in the numerical studies, we examine the accuracy of the zero-crossings of different methods compared to the object boundary (groundtruth). The colored contours in Figure 3(left) indicate the results from different methods. Also the black curve shows the isoline for $I(x, y, z_m) = I_m$, i.e. result of segmentation performed by thresholding [13]: the middle value $I_m = \frac{1}{2}(I_{\max} + I_{\min})$ is used as the threshold.

All the lines are very close to each other, which suggests that the proposed methods are close approximation to the direct method. Plots of geometrical potential $G(x, y_m, z_m)$ along the x-coordinates is shown in Figure 3(right), where

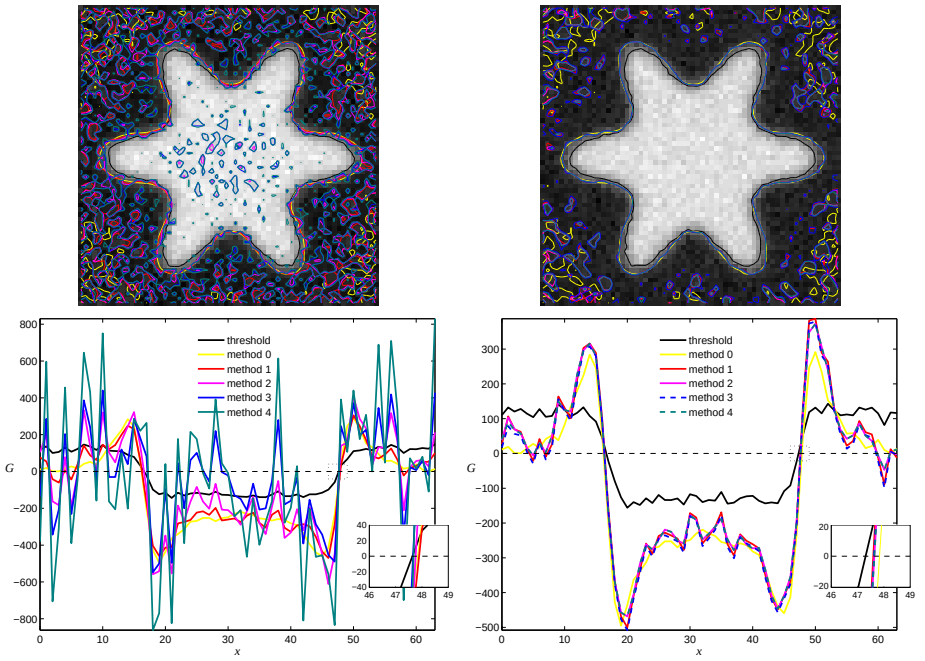

Fig. 3. Top:
A slice of the 3D image; the colored curved indicate isolines of $G = 0$. Bottom: The G variation along the x direction through the center of the 3D image computed by the different methods explained in the legend. Method 0 is direct computation of the geometric potential; method 1 is the FFT based implementation of method 0; method 2 is using analytical formula for kernel's spectrum; method 3 using the scalar kernel alone; and method 4 is combining methods 2 and 3. Left: methods 2–4 without corrections, right: methods 2–4 with corrections (7) and (14).

$y_m = \frac{1}{2}(y_{\min}+y_{\max})$. The curve $I_m - I(x, y_m)$ is plotted in black. It shows that the difference zero crossing is small. The rectangular region indicated by the dotted line is zoomed and depicted at the right border of the plot. The difference is in sub-pixel level.

The direct computation is less susceptible to noise, but it is too slow to be practical. The proposed methods produce very similar result to that using FFT computation as proposed in [7, 1]. However, the proposed methods, particularly the combined approach, are far more memory efficient.

The CPU time of all the methods can be found in Table 1. Note, the combined approach (method 4) uses 4 times less memory than the FFT based computation used in [1]. The experiment was carried out on Linux, Intel(R) Xeon 3.00GHz, RAM 4G. A typical 3D scan of 512^3 voxels can only be pratically processed by method 4 and it requires 8 min of the CPU time and 3G of memory.

To demonstrate the effectiveness of the proposed combined approach, we show an example of segmenting a human aorta from a 3D CT dataset. The

Table 1. CPU time and memory comparison for a 256^3 image. Method 0 is direct computation of the geometric potential; method 1 is the FFT based implementation of method 0; method 2 is using analytical formula for kernel's spectrum; method 3 using the scalar kernel alone; and method 4 is combining methods 2 and 3.

	method 0	method 1	method 2	method 3	method 4
CPU time	$\sim$ 7days	91s	55s	42s	30s
Memory required	0.6G	1.8G	1.0G	0.6G	0.4G

testing data and the results are shown in Figures 2(left) and 4. The initial surface is a sphere placed inside the lower part of the aorta and the model is able to propagate efficiently and converge accurately.

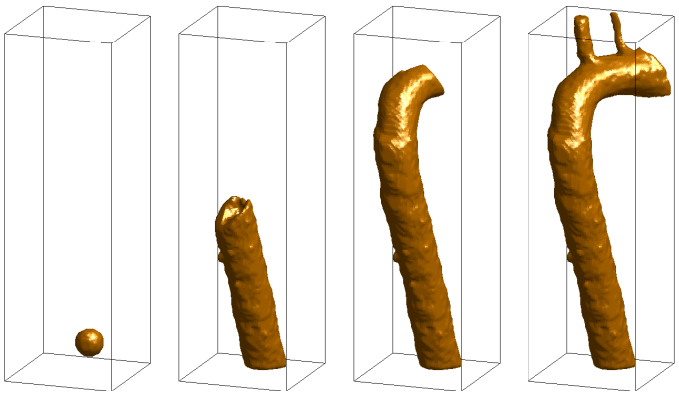

Fig. 4. An example of segmenting human aorta in 3D CT shown in Fig. 2(right) using the combined approach. From left: initial surface, intermediate stages, and final converged result.

6 4D Numerical Examples

Note that all the equations derived for the proposed methods can be readily generalised to 4D medical scans (dynamic volumetric data). We should treat the coordinate vector as $\mathbf{x} = \{x, y, z, t\}$, use the 4D wavenumber vector $\mathbf{k}$ with k_4-component treated as the frequency, and substitute $n = 4$ into the correspondent formulae for the kernel in Eqns. (1) and (3).

Here, we present a numerical study that is similar to that in the 3D case, but using a dynamic 3D shape. We vary the ray length shape parameters of the 3D star-like harmonic object periodically in time with the maximum near the middle of the cycle. The ray length parameter evolution is given as $[\frac{1}{2}(1 + \cos(2\pi(t - t_m - \frac{1}{3})/N_t))]^{1.5}$ where $N_t = 16, t_m = N_t/2$. The image dimension is

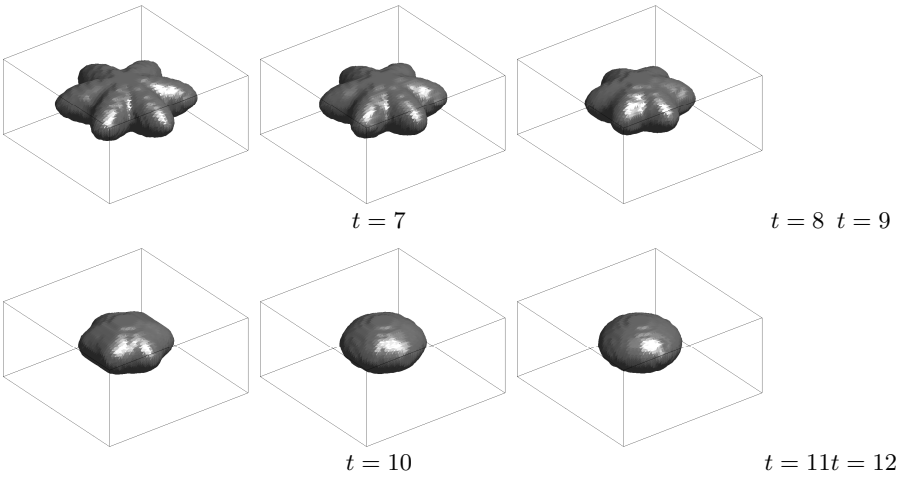

Fig. 5. Object shape at instances of 7 to12.

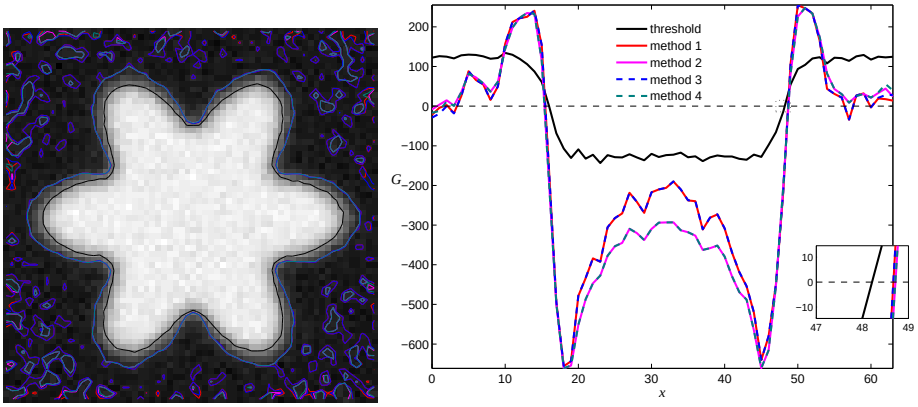

Fig. 6. Left: A slice of the 4D image; the colored curved indicate isolines of $G = 0$. Right: The G variation along the x direction through the center of the 3D image computed by the different methods explained in the legend. Method 1 is the FFT based implementation of direct computation; method 2 is using analytical formula for kernel's spectrum; method 3 using the scalar kernel alone; and method 4 is combining methods 2 and 3.

$64 \times 64 \times 32 \times 16$. Thus the image contains 16 3D images, some of which are shown in Figure 5. Similarly, Gaussian noise is also added to the dynamic shape.

Figure 6(left) shows a slice of the image at instant $t = t_m = 7$ (the maximal length of the star-rays) and $z = z_m$. Here one can find colored contours $G(x, y, z_m, t_m) = 0$ with the geometrical potential computed by the different methods implemented in 4D. Spatial zero-crossings of geometrical potential: $G(x, y_m, z_m, t_m)$ where y_m, z_m, t_m are plotted in Figure 6(right). Note, the di-

rect method is not shown as it takes prohibitive amount of time to compute the geometrical potential. There is no discernible difference among methods with improved computational efficiency. However, the proposed combined approach requires significantly less memory. This is particularly advantageous in dealing with 4D dataset.

7 Conclusion

We proposed several computationally efficient and memory economic methods to evaluate the geometrical potential in the GPF model [1]. The approach which combines analytical kernel spectrum and scalar kernel conversion provides most satisfactory results. The methods were evaluated on 3D and 4D synthetic datasets, as well as 3D real world data. This preliminary work provided promising results which suggest that the proposed method has a great potential in efficient deformable modelling in high dimensional space without decomposing the space into a sequential order.

References

1. Yeo, S.Y., Xie, X., Sazonov, I., Nithiarasu, P.: Geometrically induced force interaction for three-dimensional deformable models. IEEE T-IP **20**(5) (2011) 1373–1387
2. Malladi, R., Sethian, J.A., Vemuri, B.C.: Shape modelling with front propagation: A level set approach. IEEE T-PAMI **17**(2) (1995) 158–175
3. Whitaker, R.: Modeling deformable surfaces with level sets. IEEE Computer Graphics and App. **24**(5) (2004) 6–9
4. Xie, X.: Active contouring based on gradient vector interaction and constrained level set diffusion. IEEE T-IP **19**(1) (2010) 154–164
5. Xu, C., Prince, J.L.: Snakes, shapes, and gradient vector flow. IEEE T-IP **7**(3) (1998) 359–369
6. Xiang, Y., Chung, A., Ye, J.: A new active contour method based on elastic interaction. In: IEEE CVPR. (2005) 452–457
7. Xie, X., Mirmehdi, M.: MAC: Magnetostatic active contour model. IEEE T-PAMI **30**(4) (2008) 632–647
8. Chan, T., Vese, L.: Active contours without edges. IEEE T-IP **10**(2) (2001) 266–277
9. Caselles, V., Kimmel, R., Sapiro, G.: Geodesic active contour. IJCV **22**(1) (1997) 61–79
10. Paragios, N., Deriche, R.: Geodesic active regions and level set methods for supervised texture segmentation. IJCV **46**(3) (2002) 223–247
11. Parigios, N., Mellina-Gottardo, O., Ramesh, V.: Gradient vector flow geometric active contours. IEEE T-PAMI **26**(3) (2004) 402–407
12. Vladimirov, V.S.: Methods of the Theory of Generalized Functions. Taylor & Francis (2002)
13. Smith, C.M., Smith, J., Williams, S.K., Rodriguez, J.J., Hoying, J.B.: Automatic thresholding of three-dimensional microvascular structures from confocal microscopy images. J. Microscopy **225**(3) (2007) 244–257