

# Framelet-Based Algorithm for Segmentation of Tubular Structures

Xiaohao Cai<sup>1</sup>, Raymond H. Chan<sup>1</sup>, Serena Morigi<sup>2</sup>, and Fiorella Sgallari<sup>2</sup>

<sup>1</sup> Department of Mathematics,  
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong  
{xhcai,rchan}@math.cuhk.edu.hk

<sup>2</sup> Department of Mathematics-CIRAM, University of Bologna, Bologna, Italy  
{morigi,sgallari}@dm.unibo.it

**Abstract.** Framelets have been used successfully in various problems in image processing, including inpainting, impulse noise removal, super-resolution image restoration, etc. Segmentation is the process of identifying object outlines within images. There are quite a few efficient algorithms for segmentation that depend on the partial differential equation modeling. In this paper, we apply the framelet-based approach to identify tube-like structures such as blood vessels in medical images. Our method iteratively refines a region that encloses the possible boundary or surface of the vessels. In each iteration, we apply the framelet-based algorithm to denoise and smooth the possible boundary and sharpen the region. Numerical experiments of real 2D/3D images demonstrate that the proposed method is very efficient and outperforms other existing methods.

## 1 Introduction

In this paper, we consider the segmentation problem of branching tubular objects from 2D and 3D images. This kind of problem arises in several application fields, for example, extracting roads in aerial photography, and anatomical surfaces of tubular structures like blood vessels in Magnetic Resonance Angiography (MRA) images. Because of the necessity to obtain as much fine details as possible in real time, automatic, robust and efficient methods are needed.

There are several vessel segmentation algorithms that are based on deformable models, see [21] for an extended review. Because the explicit deformable model representation is usually impractical, level set techniques to evolve a deformable model have been introduced, and they provide implicit representation of a deformable model. However, the level set segmentation approach is computationally more expensive as it needs to cover the entire domain of interest, which is generally one dimension higher than the original one. Interested readers are referred to recent literature on the level set segmentation strategy for tubular structures [18,20,24].

A new model for active contours to detect objects in a given image based on techniques of curve evolution, Mumford-Shah functional and level sets was

proposed in [10]. A generalization of the active contour without edges model was proposed in [23] for object detection using logic operations. This logic framework suffers from the same limits as in the active contour model and is not suitable for detecting tubular structures.

In [16], a geometric deformable model for the segmentation of tubular-like structures was proposed. The model is characterized mainly by two components: the mean curvature flow and the directionality of the tubular structures. The major advantage of this technique is the ability to segment twisted, convoluted and occluded structures without user interactions; and it can follow the branching of different layers, from thinner to larger structures. The dependence on the grid resolution chosen to solve the discretized partial differential equation (PDE) model is still an open problem. The authors in [16] have also applied a variant of the proposed PDE model to the challenging problem of composed segmentation in [17].

There are some work on texture classification and segmentation using wavelets or wavelet frames [25,1]. Framelet-based approach is a versatile and effective tool for many different applications in image processing, see [4,7,8,3]. Recently, the authors in [13] proposed to combine the framelet-based image restoration model of [5] and the total variation based segmentation model of [10,9,2] to do segmentation. In this paper, we also derive a segmentation algorithm that uses the framelet-based approach. However our method is not based on minimizing any variational model and hence it is different from the method in [13]. In fact, our algorithm gradually updates an interval that contains pixel values of possible boundary pixels. Like the method in [16], our method also has the ability to segment twisted, convoluted and occluded structures. In addition, our method is very effective in denoising and can extract more details from the given image.

The rest of the paper is organized as follows. In Section 2, we recall some basic facts about tight frames and framelet-based algorithms. Our segmentation algorithm is given in Section 3. Section 4 discusses how to find an interval that contains pixel values of possible boundary pixels. In Section 5 we test our algorithm on various real 2D and 3D images. Comparisons with other methods will be also given. Conclusions are given in Section 6.

## 2 Framelet-Based Algorithm

In this section, we briefly introduce the framelet-based algorithm. For theories of tight frames and framelets, we refer the readers to [11] for more details. In order to apply the framelet-based algorithm, one only needs to know the filters corresponding to the framelets. For the framelets derived from the piecewise linear B-spline, the corresponding filters are:

$$h_0 = \frac{1}{4}[1, 2, 1], \quad h_1 = \frac{\sqrt{2}}{4}[1, 0, -1], \quad h_2 = \frac{1}{4}[-1, 2, -1], \quad (1)$$

see [22]. The framelet coefficients of any given vector  $\mathbf{v}$  corresponding to filter  $h_i$  can be obtained by convolving  $h_i$  with  $\mathbf{v}$ . In matrix terms, we can construct,

for each filter, its corresponding filter matrix which is just the Toeplitz matrix with diagonals given by the filter coefficients, e.g.  $H_0 = \frac{1}{4}\text{tridiag}[1, 2, 1]$ . Then the 1D framelet forward transform is given by

$$A = \begin{bmatrix} H_0 \\ H_1 \\ H_2 \end{bmatrix}. \quad (2)$$

To apply the framelet transform onto  $\mathbf{v}$  is equivalent to computing  $A\mathbf{v}$ , and  $H_i\mathbf{v}$  gives the framelet coefficients corresponding to the filter  $h_i$ ,  $i = 1, 2, 3$ .

The  $d$ -dimensional framelet system is constructed by tensor products from the 1D framelets, see [14]. For example, in 2D, there are nine framelets given by  $h_{ij} \equiv h_i^T \otimes h_j$  for  $i, j = 1, 2, 3$ , where  $h_i$  is given in (1). For any 2D image  $f$ , the framelet coefficients with respect to  $h_{ij}$  are obtained by convolving  $h_{ij}$  with  $f$ . In 3D, there are twenty-seven filters and the framelet coefficients can also be obtained by convolutions. Let  $A$  represent the corresponding framelet forward transform matrix (cf. (2)). In 2D case,  $A$  will be a stack of nine block-Toeplitz-Toeplitz-block matrices, see [4]. In this notation, given any image  $f$ , the matrix-vector product  $A \cdot \text{vec}(f)$  gives all the framelet coefficients. Here  $\text{vec}(f)$  denotes the vector obtained by concatenating the columns of  $f$ .

All framelet transforms have a very important property, the “*perfect reconstruction property*”:  $A^T A = I$ , the identity matrix, see [22]. Unlike the wavelets, in general,  $AA^T \neq I$ . The framelet-based algorithms, as given in [8, 43], are of the following generic form:

$$\mathbf{f}^{(i+\frac{1}{2})} = \mathcal{U}(\mathbf{f}^{(i)}), \quad (3)$$

$$\mathbf{f}^{(i+1)} = A^T \mathcal{T}_\lambda(A\mathbf{f}^{(i+\frac{1}{2})}), \quad i = 1, 2, \dots \quad (4)$$

Here  $\mathbf{f}^{(i)}$  is an approximate solution,  $\mathcal{U}$  is a problem-dependent operator, and  $\mathcal{T}_\lambda(\cdot)$  is the soft-thresholding operator defined as follows. Given vectors  $\mathbf{v} = [v_1, \dots, v_n]^T$  and  $\lambda = [\lambda_1, \dots, \lambda_n]^T$ ,  $\mathcal{T}_\lambda(\mathbf{v}) \equiv [t_{\lambda_1}(v_1), \dots, t_{\lambda_n}(v_n)]^T$ , where

$$t_{\lambda_k}(v_k) \equiv \begin{cases} \text{sgn}(v_k)(|v_k| - \lambda_k), & \text{if } |v_k| > \lambda_k, \\ 0, & \text{if } |v_k| \leq \lambda_k. \end{cases} \quad (5)$$

For how to choose  $\lambda_k$ , see [15].

Algorithm (4) is usually called the *isotropic* framelet-based algorithm. This is because the thresholding operator  $\mathcal{T}_\lambda$  is applied on all the framelet coefficients  $A\mathbf{f}^{(i+\frac{1}{2})}$  in (4). In [7], the *anisotropic* framelet-based algorithm was proposed. The main idea is that the filter  $h_1$  in (1) is the central-difference apart from a scalar multiple. Hence the corresponding framelet coefficients are related to the gradient  $\nabla f$  of the image  $f$ . One should therefore rotate these coefficients along the tangential and normal direction, and threshold only the components along the tangential direction, see [26]. For the coefficients corresponding to other filters, we threshold as in the isotropic framelet-based algorithm. In [7], it was shown that the anisotropic thresholding scheme can give better restoration than the isotropic one, and can follow edges more closely. Later in the numerical tests, we have tried both thresholding schemes, and found that anisotropic thresholding can give the tubular structures better.

### 3 Framelet-Based Algorithm for Segmentation

The technology of MRA imaging is based on detection of signals from flowing blood and suppression of signals from other static tissues, so that the blood vessels appear as high intensity regions in the image, see Fig. 1(a). The structures to be segmented are vessels of variable diameters which are close to each other. Partial occlusions and intersections make the segmentation very challenging. Moreover, the real image can be affected by speckle noise. In general in medical images, speckle noise and weak edges make it difficult to identify the structures in the image. Fortunately, the MRA images also contain some properties that can be used to construct our algorithm. From Fig. 1(a), we see that the pixels near the boundary of the vessels are not exactly of one value, but they are in some range, whereas the values of the pixels in other parts are far from this range. Thus the main idea of our algorithm is to approximate this range accurately. We will obtain the range iteratively by a framelet-based algorithm. The main steps are as follows. Suppose in the beginning of the  $i$ th iteration, we are given an approximate image  $f^{(i)}$ , and an approximate range  $[\alpha_i, \beta_i]$  for  $\alpha_i \leq \beta_i$  which contains the pixel values of all the possible boundary pixels. Then we (i) use the range to threshold the image into three parts—below, inside, and above the range; (ii) denoise and smooth the inside part by the framelet-based algorithm to get a new image  $f^{(i+1)}$ ; and (iii) refine the range to  $[\alpha_{i+1}, \beta_{i+1}]$  by using  $f^{(i+1)}$ . We stop when  $f^{(i+1)}$  becomes a binary image. In the followings, we elaborate each of the steps. Without loss of generality, we assume all images have dynamic range in  $[0, 1]$ .

**Step (i): Thresholding the Image into Three Parts.** Using the range  $[\alpha_i, \beta_i] \subseteq [0, 1]$ , we can separate the image  $f^{(i)}$  into three parts—below, inside, and above the range, see Fig. 1(b). To emphasize the boundary, we threshold those pixel values that are smaller than  $\alpha_i$  to 0, those larger than  $\beta_i$  to 1, and those in between, we stretch them between 0 and 1 using a simple linear contrast stretch, see [19]. More precisely, let  $\Omega$  be the index set of all the pixels in the image,  $f_j^{(i)}$  be the pixel value of pixel  $j$  in image  $f^{(i)}$ , and

$$\begin{aligned} M_i &= \max\{f_j^{(i)} \mid \alpha_i \leq f_j^{(i)} \leq \beta_i, j \in \Omega\}, \\ m_i &= \min\{f_j^{(i)} \mid \alpha_i \leq f_j^{(i)} \leq \beta_i, j \in \Omega\}. \end{aligned}$$

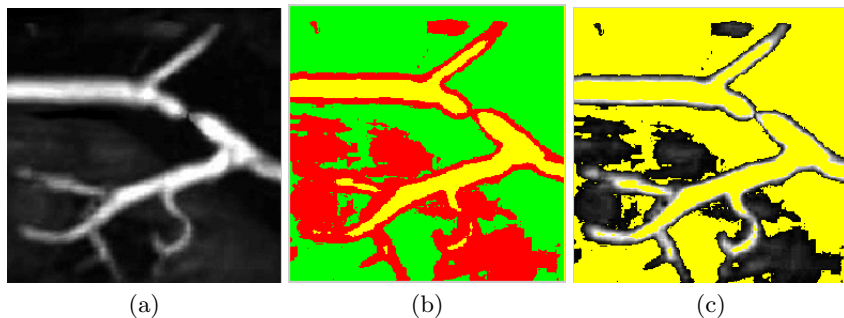
Then we define

$$f_j^{(i+\frac{1}{2})} = \begin{cases} 0, & \text{if } f_j^{(i)} \leq \alpha_i, \\ \frac{f_j^{(i)} - m_i}{M_i - m_i}, & \alpha_i \leq f_j^{(i)} \leq \beta_i, \\ 1, & \text{if } \beta_i \leq f_j^{(i)}, \end{cases} \quad \text{for all } j \in \Omega. \quad (6)$$

Fig. 1(c) shows the threshold and stretched image from Fig. 1(b). In the following, we write (6) simply as  $f^{(i+\frac{1}{2})} = \mathcal{U}(f^{(i)})$  (cf. (3)), and we denote

$$\Lambda^{(i)} = \{j \mid m_i < f_j^{(i)} < M_i, j \in \Omega\}, \quad (7)$$

the index set for pixels with values inside the range  $(m_i, M_i)$ , i.e., the index set for pixels of  $f^{(i+\frac{1}{2})}$  with values neither 0 nor 1. Our next step is to denoise and smooth the image  $f^{(i+\frac{1}{2})}$  on  $\Lambda^{(i)}$ .



**Fig. 1.** (a) Given image. (b) Three parts of the given image (green–below, red–in between, and yellow–above). (c) Threshold and stretched image by (6) (yellow pixels are with value 0 or 1).

**Step (ii): Framelet-Based Iteration.** To denoise and smooth the image  $f^{(i+\frac{1}{2})}$  on  $\Lambda^{(i)}$ , we apply the framelet-based iteration (4) on  $\Lambda^{(i)}$ . More precisely, if  $j \notin \Lambda^{(i)}$ , then we set  $f_j^{(i+1)} = f_j^{(i+\frac{1}{2})}$ ; otherwise, we use (4) to get  $f_j^{(i+1)}$ . To write it out clearly, let  $\mathbf{f}^{(i+\frac{1}{2})} = \text{vec}(f^{(i+\frac{1}{2})})$ , and  $P^{(i)}$  be the diagonal matrix where the diagonal entry is 1 if the corresponding index is in  $\Lambda^{(i)}$ , and 0 otherwise. Then

$$\mathbf{f}^{(i+1)} \equiv (I - P^{(i)})\mathbf{f}^{(i+\frac{1}{2})} + P^{(i)}A^T\mathcal{T}_\lambda(A\mathbf{f}^{(i+\frac{1}{2})}). \quad (8)$$

By reordering the entries of the vector  $\mathbf{f}^{(i+1)}$  into columns, we obtain the image  $f^{(i+1)}$ . Note that the effect of (8) is to denoise and smooth the image on  $\Lambda^{(i)}$ , see (4). Since the pixel values of all pixels outside  $\Lambda^{(i)}$  are either 0 or 1, the cost of matrix-vector multiplications in (8), such as  $A\mathbf{f}^{(i+\frac{1}{2})}$ , can be reduced significantly by taking advantage of this.

**Step (iii): Refining the Range.** The process of finding the new range  $[\alpha_{i+1}, \beta_{i+1}]$  from  $f^{(i+1)}$  is very similar to the process of finding the initial interval  $[\alpha_0, \beta_0]$  from the given image. We postpone it till the next section. We will see that  $[\alpha_{i+1}, \beta_{i+1}] \subsetneq [0, 1]$  for all  $i \geq 0$ . This point guarantees the convergence of our method, see Theorem II.

**Stopping Criterion.** We stop the iteration when all the pixels of  $f^{(i+\frac{1}{2})}$  are either of value 0 or 1, or equivalently when  $|\Lambda^{(i)}| = 0$ . For the binary image  $f^{(i+\frac{1}{2})}$ , all the pixels with value 1 constitute the tubular structures. In the numerical tests, we use the matlab command “`contour`” and “`isosurface`” respectively to obtain the boundary of  $f^{(i+\frac{1}{2})}$  in 2D and 3D respectively.

## 4 Initializing and Refining the Range

In this section, we discuss how to find  $[\alpha_i, \beta_i]$  given  $f^{(i)}$ . When  $i = 0$ , the initial guess  $f^{(0)}$  is chosen to be the given image. Recall that  $[\alpha_i, \beta_i]$  is an interval containing the pixel values of the possible boundary pixels. Our idea of finding it is as follows: (i) find the average  $\mu^{(i)}$  of the pixel values of the possible boundary pixels; and (ii) determine a suitable interval  $[\alpha_i, \beta_i] \subsetneq [0, 1]$  that contains  $\mu^{(i)}$ .

For  $i = 0$ , since we do not have any knowledge of where the boundary will possibly be, we use the gradient of  $f^{(0)}$  to find it. Define the gradient image  $g$  of  $f^{(0)}$  as:

$$g_j = \left[ \sum_{\ell=1}^d (\partial_{x_\ell} f_j^{(0)})^2 \right]^{1/2}, \quad \text{for all } j \in \Omega,$$

where  $\partial_{x_\ell}$  is the forward-difference in the  $x_\ell$ -direction, and  $d = 2$  for 2D and  $d = 3$  for 3D. Our first approximation of the boundary is composed of those pixels where  $g_j > \epsilon$  for a given  $\epsilon$ . (In our numerical tests, we choose  $\epsilon \in [10^{-3}, 10^{-1}]$ .) Thus let  $\Gamma = \{j \mid g_j > \epsilon, j \in \Omega\}$ , the index set of those pixels; and let  $\mu_\Gamma$  be the average of  $f^{(0)}$  on  $\Gamma$ , i.e.

$$\mu_\Gamma = \frac{1}{|\Gamma|} \sum_{j \in \Gamma} f_j^{(0)},$$

where  $|\Gamma|$  is the cardinality of  $\Gamma$ . Obviously the smaller the  $\epsilon$  is, the larger cardinality of  $|\Gamma|$  will be.

Naturally, those pixels in  $\Gamma$  can be separated into two parts by  $\mu_\Gamma$ : one part contains the pixels near to the tubulars (yellow part of Fig. [1\(b\)](#)), and the other part is near to the background (green part of Fig. [1\(b\)](#)). More precisely, we define

$$\Gamma_+ = \{j \mid f_j > \mu_\Gamma, j \in \Gamma\} \quad \text{and} \quad \Gamma_- = \{j \mid f_j < \mu_\Gamma, j \in \Gamma\}.$$

Let  $\mu_+$  and  $\mu_-$  be the averages of  $f^{(0)}$  on  $\Gamma_+$  and  $\Gamma_-$  respectively. Note that  $\mu_+$  (and respectively  $\mu_-$ ) is the average of those possible boundary pixels that are close to the tubulars (and respectively close to the background). We use them to compute  $\mu^{(0)}$ , the average of the possible boundary pixels. Define

$$\Lambda^{(-1)} = \{j \mid \mu_- < f_j^{(0)} < \mu_+, j \in \Omega\}.$$

Then

$$\mu^{(0)} = \frac{1}{|\Lambda^{(-1)}|} \sum_{j \in \Lambda^{(-1)}} f_j^{(0)}. \quad (9)$$

For  $i \geq 0$ , we define  $\mu^{(i+1)}$  similarly as in [\(9\)](#):

$$\mu^{(i+1)} = \frac{1}{|\Lambda^{(i)}|} \sum_{j \in \Lambda^{(i)}} f_j^{(i+1)}, \quad (10)$$

where  $\Lambda^{(i)}$  is given by [\(7\)](#) and  $f^{(i+1)}$  is given by [\(8\)](#).

Finally we discuss how to choose the interval  $[\alpha_i, \beta_i]$  that contains  $\mu^{(i)}$  for  $i \geq 0$ . Our idea is first to compute coarse interval  $[\alpha_i^L, \beta_i^H]$  that contains  $\mu^{(i)}$  by

$$\alpha_i^L = \frac{1}{|\{j : f_j^{(i)} \leq \mu^{(i)}\}|} \sum_{\{j : f_j^{(i)} \leq \mu^{(i)}\}} f_j^{(i)}, \quad \beta_i^H = \frac{1}{|\{j : f_j^{(i)} \geq \mu^{(i)}\}|} \sum_{\{j : f_j^{(i)} \geq \mu^{(i)}\}} f_j^{(i)},$$

where  $j \in \Lambda^{(i-1)}$  for  $i \geq 1$  and  $j \in \Omega$  for  $i = 0$ . From the above formulas, we can see that  $[\alpha_i^L, \beta_i^H]$  will never be  $[0, 1]$  if the given image is not a binary image. (If  $\alpha_i^L = \beta_i^H$ , then all remaining pixels have the same pixel value. Hence we set them all to 1 and the image is thus a binary image, and the algorithm stops.)

Next we compute, for all  $\alpha \in [\alpha_i^L, \beta_i^H]$  ( $\alpha_i^L \neq \beta_i^H$ ),

$$c(\alpha) \equiv \frac{1}{|\{j : f_j^{(i)} \geq \alpha\}|} \sum_{\{j : f_j^{(i)} \geq \alpha\}} f_j^{(i)} - \frac{1}{|\{j : f_j^{(i)} \leq \alpha\}|} \sum_{\{j : f_j^{(i)} \leq \alpha\}} f_j^{(i)},$$

where  $j \in \Omega$ . Let the range of  $c(\alpha)$  be  $[c_m, c_M]$  and  $\ell = c_M - c_m$ . Then

$$\alpha_i \equiv \min\{\alpha \in [\alpha_i^L, \beta_i^H] \mid c(\alpha) = c(\mu^{(i)}) - \gamma\ell\}, \quad (11)$$

$$\beta_i \equiv \max\{\alpha \in [\alpha_i^L, \beta_i^H] \mid c(\alpha) = c(\mu^{(i)}) + \gamma\ell\}. \quad (12)$$

Here  $\gamma \in (0, 1/2)$  is a parameter that controls the length of the interval  $[\alpha_i, \beta_i]$ . Clearly the larger the interval is, the more pixels are to be considered as possible boundary pixels. So the smaller the  $\gamma$  is, the faster convergence of our method will have. In the numerical tests, we choose  $\gamma = 1/5$ . We give the full algorithm below and show that it always converges to a binary image.

---

**Algorithm:** Framelet-based algorithm for segmentation

---

1. Initialize: set  $f^{(0)} = f$ ,  $\mu^{(0)}$  by (9), and  $[\alpha_0, \beta_0]$  by (11) and (12).
  2. Do  $i = 0, 1, \dots$ , until stopped
    - (a) Compute  $f^{(i+\frac{1}{2})} = \mathcal{U}(f^{(i)})$  by (6).
    - (b) Stop if  $f^{(i+\frac{1}{2})}$  is a binary image.
    - (c) Update  $f^{(i+\frac{1}{2})}$  to  $f^{(i+1)}$  by (8).
    - (d) Update  $\mu^{(i+1)}$  by (10), and then  $[\alpha_{i+1}, \beta_{i+1}]$  by (11) and (12).
  3. Extract the boundary from the binary image  $f^{(i+\frac{1}{2})}$ .
- 

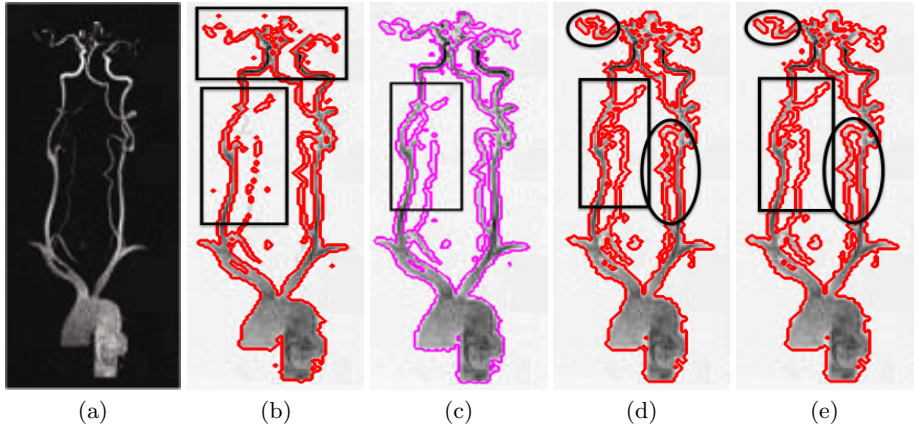
**Theorem 1.** *Our framelet-based algorithm will converge to a binary image.*

*Proof.* Obviously, we just need to prove that  $|\Lambda^{(i)}| = 0$  at some finite step  $i$ , see (6) and (7). If the given image  $f^{(0)}$  is a binary image, we are done. Without loss of generality, we assume that  $f^{(0)}$  is not a binary image. Given  $\Lambda^{(i-1)}$  defined by (7) for any  $i \geq 1$ , note that the pixel values of those pixels not in  $\Lambda^{(i-1)}$  will not be changed by (8), i.e., they will stay at either 0 or 1. Then  $[\alpha_i, \beta_i]$  will be obtained by (11) and (12), where  $[\alpha_i, \beta_i] \subseteq [\alpha_i^L, \beta_i^H] \subsetneq [0, 1]$ . Since  $[m_i, M_i] \subseteq [\alpha_i, \beta_i]$ , we have  $m_i \neq 0$  or  $M_i \neq 1$ . By (6) and (7), the pixels satisfying  $f^{(i)} \leq m_i$  or  $f^{(i)} \geq M_i$  are set to 0 or 1 respectively. Thus, there will be at least one pixel in  $\Lambda^{(i-1)}$  with value neither 0 nor 1 that is set to 0 or 1 by (6). Hence  $|\Lambda^{(i)}| < |\Lambda^{(i-1)}|$ , where  $|\cdot|$  denotes the cardinality of the set. Since  $|\Lambda^{(0)}|$  is finite, there must exist some  $i$  such that  $|\Lambda^{(i)}| = 0$ .  $\square$

Finally, let us estimate the computation cost of our method for a given image with  $n$  pixels: (i) the cost of computing each  $\mu^{(i)}$  and  $[\alpha_i, \beta_i]$  is  $O(n)$ , see (9)–(12) respectively; and (ii) the cost of steps (a) and (c) in the above algorithm is  $O(n)$ , see (6) and (8). Hence the cost of our method is  $O(n)$  per iteration. We remark that our algorithm usually converges within a few iterations, see the numerical results in the next section.

## 5 Numerical Examples

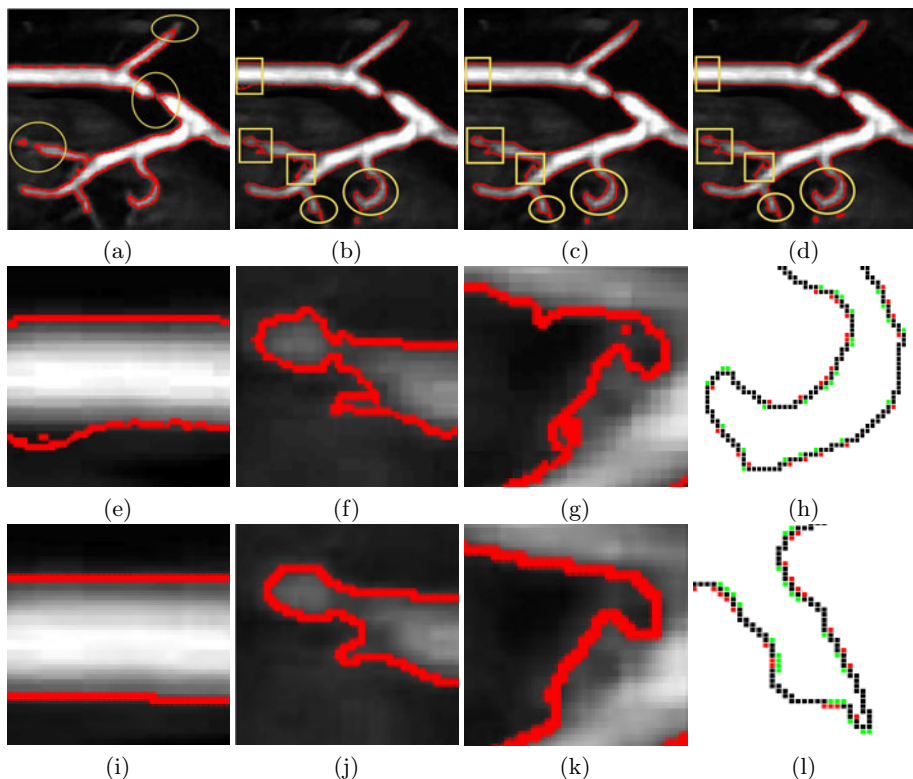
In this section, we test our proposed framelet-based segmentation method on three 2D/3D real images. All the data are obtained from [16] and [17]. We use the piece-wise linear filters given in (11) with only the first level, i.e. no downsampling. We tried both isotropic and anisotropic thresholding schemes, see the discussion at the end of Section 2. The thresholding parameters  $\lambda_k$  used in (5) are chosen to be  $\lambda_k \equiv 2^{-1/2}$  for isotropic thresholding. For anisotropic thresholding,  $\lambda_k = 0.1 \times 2^{-1/2}$  for the components along the tangential direction and  $\lambda_k = 2^{-1/2}$  for other coefficients.



**Fig. 2.** Carotid vascular system segmentation. (a) Given image. (b) and (c) Results by the methods in [10] and [16] respectively. (d) and (e) Results by our method with isotropic and anisotropic thresholding schemes respectively.

*Example 1.* The test image is a  $182 \times 62$  MRA image of a carotid vascular system, see Fig. 2(a). The results by our method using isotropic and anisotropic thresholding are given by Fig. 2(d) and (e) respectively. With the parameters  $\gamma = 1/5$  and  $\epsilon = 1.6 \times 10^{-2}$ , our method converges in 6 iterations for both thresholding schemes. The first and second rows of Table 1 give  $|\Lambda^{(i)}|$  at each iteration, from which, we can see that only very few pixels (comparing with  $|\Omega| = 182 \times 62 = 11,284$ ) need to be classified after 3 iterations. For the purpose of



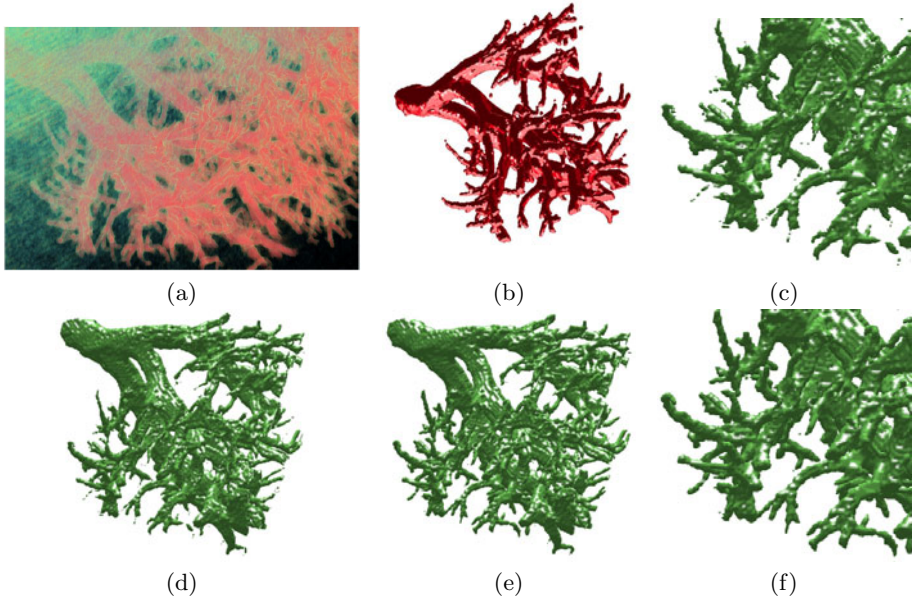


**Fig. 3.** Kidney vascular system segmentation. (a) and (b) Results by the methods in [10] and [16] respectively. (c) and (d) Results by our method with isotropic and anisotropic thresholding respectively. (e)–(g) and (i)–(k) Zoomed in rectangular parts of (b) and (c) respectively. (h) and (l) Superimposed boundaries inside the ellipses of (c) and (d) (red—in (c), green—in (d), and black—intersection of (c) and (d)).

**Table 1.** Cardinality of  $\Lambda^{(i)}$  at each iteration of the three examples

	$n =  \Omega $	$ \Lambda^{(0)} $	$ \Lambda^{(1)} $	$ \Lambda^{(2)} $	$ \Lambda^{(3)} $	$ \Lambda^{(4)} $	$ \Lambda^{(5)} $	$ \Lambda^{(6)} $	$ \Lambda^{(7)} $	$ \Lambda^{(8)} $	$ \Lambda^{(9)} $
Fig. 2(d)	11284	2374	307	83	23	7	1	0	-	-	-
Fig. 2(e)	11284	2374	233	48	13	5	1	0	-	-	-
Fig. 3(c)	66049	8314	1834	565	137	29	18	4	0	-	-
Fig. 3(d)	66049	8314	1557	406	95	19	5	1	0	-	-
Fig. 4(d)	8120601	104329	21333	5460	1430	326	70	9	3	1	0
Fig. 4(e)	8120601	104329	20020	4984	1260	299	72	19	6	0	-

comparison, we also give the results by the methods in [10] and [16] respectively, see Fig. 2(b) and (c). Clearly, the result of Fig. 2(b) is not satisfactory since the tubulars obtained are disconnected. By comparing the parts inside the rectangles in Fig. 2 (c) with those in Fig. 2(d) and (e), we see that our method can extract



**Fig. 4.** Segmentation of the kidney volume data set. (a) Given CTA image. (b) Result by the method in [17]. (d) and (e) Results by our method with isotropic and anisotropic thresholding respectively. (c) and (f) Zoomed in the bottom-left corners of (d) and (e) respectively.

more details than the method in [16]. Finally, the parts inside the ellipses of Fig. 2(d) and (e) demonstrate that the anisotropic thresholding keeps the edge better than isotropic thresholding.

*Example 2.* The test image is a  $257 \times 257$  MRA image of a kidney vascular system as shown in Fig. 1(a). This example shows the ability of our method to reconstruct structures which present small occlusions along the coherence direction. With the parameters  $\gamma = 1/5$  and  $\epsilon = 5 \times 10^{-3}$ , our method converges in 7 iterations for both thresholding schemes. The third and fourth rows in Table 1 give  $|\Lambda^{(i)}|$  at each iteration. The result of Fig. 3(a) by the method in [10] is not good since it can not connect the small occlusions along the coherence direction, while this can be done by our method and the method in [16], see Fig. 3(b), (c) and (d). Furthermore, our method is better than the method in [16] by comparing the rectangular parts of Fig. 3(b) with those in Fig. 3(c) and (d), since our method can detect smoother edges. More precisely, see Fig. 3(e)–(g) and (i)–(k), which are the results of zooming in the rectangular parts of Fig. 3(b) and (c) respectively. This also shows that our method is very effective in denoising. In order to compare Fig. 3(c) with (d) explicitly, we superimpose the boundaries of them, see Fig. 3(h) and (l). Clearly, the boundary of the tubulars is tighter and more pixels at the tips are obtained by anisotropic thresholding.

*Example 3.* This is a 3D example where we extracted a volumetric data set of size  $201 \times 201 \times 201$  from a  $436 \times 436 \times 540$  CTA (Computed Tomographic Angiography) image of the kidney vasculature system, see Fig. 4(a). With the parameters  $\gamma = 1/5$  and  $\epsilon = 6 \times 10^{-2}$ , our method converges in 9 and 8 iterations for isotropic and anisotropic thresholding respectively. The last two rows in Table II show  $|A^{(i)}|$  at each iteration. By comparing our method with the method in [17], the results show that our method can give many more details, see Fig. 4(b), (d) and (e). The zoomed in bottom-left corners of Fig. 4(d) and (e) clearly give the difference of the results of our method by isotropic and anisotropic thresholding, see Fig. 4(c) and (f). We see that more pixels at the tips of the tubular structures are detected and the tubular structures are connected better by the anisotropic thresholding than by the isotropic thresholding.

## 6 Conclusions and Future Work

In this paper, we introduced a new segmentation method based on the framelet-based approach. The numerical results demonstrate the ability of our method for segmenting tubular structures. The method can be implemented fast and give very accurate, smooth boundaries or surfaces. In addition, since the pixel values of more and more pixels will be set to either 0 or 1 during the iteration, by taking advantage of this, one can construct a sparse data structure to accelerate the method. Moreover, one can use different tight frame systems such as those from contourlets and curvelets [12,6] to better capture the boundary. These are directions we will explore in the future.

## References

1. Arivazhagan, S., Ganesan, L.: Texture segmentation using wavelet transform. *Pattern Recognition Letters* 24, 3197–3203 (2003)
2. Bresson, X., Esedoglu, S., Vanderghelynst, P., Thiran, J., Osher, S.: Fast global minimization of the active contour/snake model. *J. Math. Imaging Vision* 28, 151–167 (2007)
3. Cai, J.F., Chan, R.H., Shen, L.X., Shen, Z.W.: Simultaneously inpainting in image and transformed domains. *Numer. Math.* 112, 509–533 (2009)
4. Cai, J.F., Chan, R.H., Shen, Z.W.: A framelet-based image inpainting algorithm. *Appl. Comput. Harmon. Anal.* 24, 131–149 (2008)
5. Cai, J.F., Osher, S., Shen, Z.W.: Split Bregman methods and frame based image restoration. *Multiscale Modeling and Simulation* 8, 337–369 (2009)
6. Candès, E., Demanet, L., Donoho, D., Ying, L.: Fast discrete curvelet transforms. *Multiscale Modeling and Simulation* 5, 861–899 (2006)
7. Chan, R.H., Setzer, S., Steidl, G.: Inpainting by flexible Haar-wavelet shrinkage. *SIAM J. Imaging Sci.* 1, 273–293 (2008)
8. Chan, R.H., Chan, T.F., Shen, L.X., Shen, Z.W.: Wavelet algorithms for high-resolution image reconstruction. *SIAM J. Sci. Comput.* 24, 1408–1432 (2003)
9. Chan, T.F., Esedoglu, S., Nikolova, M.: Algorithms for finding global minimizers of image segmentation and denoising models. Technical Report 54, UCLA (2004)

10. Chan, T.F., Vese, L.A.: Active contours without edges. *IEEE Trans. Image Process.* 10, 266–277 (2001)
11. Daubechies, I.: Ten lectures on wavelets. Lecture Notes, vol. CBMS-NSF(61). SIAM, Philadelphia (1992)
12. Do, M.N., Vetterli, M.: The contourlet transform: an efficient directional multiresolution image representation. *IEEE Trans. Image Process.* 14, 2091–2106 (2004)
13. Dong, B., Chien, A., Shen, Z.W.: Frame based segmentation for medical images. Technical Report 22, UCLA (2010)
14. Dong, B., Shen, Z.W.: MRA based wavelet frames and applications. IAS Lecture Notes Series, Summer Program on The Mathematics of Image Processing, Park City Mathematics Institute (2010)
15. Donoho, D.L.: De-noising by soft-thresholding. *IEEE Trans. Inform. Theory* 41, 613–627 (1995)
16. Franchini, E., Morigi, S., Sgallari, F.: Segmentation of 3D tubular structures by a PDE-based anisotropic diffusion model. In: Dæhlen, M., Floater, M., Lyche, T., Merrien, J.-L., Mørken, K., Schumaker, L.L. (eds.) *MMCS 2008. LNCS*, vol. 5862, pp. 224–241. Springer, Heidelberg (2010)
17. Franchini, E., Morigi, S., Sgallari, F.: Composited segmentation of tubular structures by an anisotropic PDE model. In: Tai, X.-C., Mørken, K., Lysaker, M., Lie, K.-A. (eds.) *SSVM 2009. LNCS*, vol. 5567, pp. 75–86. Springer, Heidelberg (2009)
18. Gooya, A., Liao, H., et al.: A variational method for geometric regularization of vascular segmentation in medical images. *IEEE Trans. Image Process.* 17, 1295–1312 (2008)
19. Gonzales, R.C., Woods, R.E.: *Digital Image Processing*, 3rd edn. Prentice Hall, Englewood Cliffs (2008)
20. Hassan, H., Farag, A.A.: Cerebrovascular segmentation for MRA data using levels set. *International Congress Series*, vol. 1256, pp. 246–252 (2003)
21. Kirbas, C., Quek, F.: A review of vessel extraction techniques and algorithms. *ACM Computing Surveys* 36, 81–121 (2004)
22. Ron, A., Shen, Z.W.: Affine Systems in  $L^2(\mathbb{R}^d)$ : The Analysis of the Analysis Operator. *J. Funct. Anal.* 148, 408–447 (1997)
23. Sandberg, B., Chan, T.F.: A logic framework for active contours on multi-channel images. *J. Vis. Commun. Image R.* 16, 333–358 (2005)
24. Scherl, H., et al.: Semi automatic level set segmentation and stenosis quantification of internal carotid artery in 3D CTA data sets. *Medical Image Analysis* 11, 21–34 (2007)
25. Unser, M.: Texture classification and segmentation using wavelet frames. *IEEE Trans. Image Process.* 4, 1549–1560 (1995)
26. Weickert, J.: *Anisotropic Diffusion in Image Processing*. Teubner, Stuttgart (1998)