Chapter 14
Segmentation and Blood Flow Simulations of Patient-Specific Heart Data

Dimitris Metaxas, Scott Kulp, Mingchen Gao, Shaoting Zhang, Zhen Qian, and Leon Axel

Abstract In this chapter, we present a fully automatic and accurate segmentation framework for 2D cardiac tagged MR images, a semiautomatic method for 3D segmentation from CT data, and the results of blood flow simulation using these highly detailed models. The 2D segmentation system consists of a semiautomatic segmentation framework to obtain the training contours, and a learning-based framework that is trained by the semiautomatic results, and achieves fully automatic and accurate segmentation.

We then present a method to simulate and visualize blood flow through the human heart, using the reconstructed 4D motion of the endocardial surface of the left ventricle as boundary conditions. The reconstruction captures the motion of the full 3D surfaces of the complex features, such as the papillary muscles and the ventricular trabeculae. We use visualizations of the flow field to view the interactions between the blood and the trabeculae in far more detail than has been achieved previously, which promises to give a better understanding of cardiac flow. Finally, we use our simulation results to compare the blood flow within one healthy heart and two diseased hearts.

D. Metaxas (✉) • S. Kulp • M. Gao • S. Zhang
CBIM, Rutgers University, New Brunswick, NJ, USA
e-mail: dnm@cs.rutgers.edu; sckulp@cs.rutgers.edu; minggao@cs.rutgers.edu; shaoting@cs.rutgers.edu

Z. Qian
Piedmont Heart Institute, Atlanta, GA, USA
e-mail: Zhen.Qian@piedmont.org

L. Axel
NYU School of Medicine, New York, NY, USA
e-mail: Leon.Axel@nyumc.org

M. Garbey et al. (eds.), *Computational Surgery and Dual Training: Computing, Robotics and Imaging*, DOI 10.1007/978-1-4614-8648-0_14,

Keywords Patient specific simulation • Magnetic resonance images • Image segmentation • Hemodynamic • Heart flow • Flow visualization • Metamorphs segmentation • Shape model • Adaboost learning • Deformable model • Valves deformation • Heart disease • Ejection fraction

14.1 Cardiac Tagged MRI Segmentation

Tagged cardiac magnetic resonance imaging (MRI) is a well-known technique for noninvasively visualizing the detailed motion of myocardium throughout the heart cycle. This technique has the potential of early diagnosis and quantitative analysis of various kinds of heart diseases and malfunction. This technique generates a set of equally spaced parallel tagging planes within the myocardium as temporary markers at end-diastole by spatial modulation of magnetization. Imaging planes are perpendicular to the tagging planes, so that the tags appear as parallel dark stripes in MR images and deform with the underlying myocardium during the cardiac cycle in vivo, which gives motion information of the myocardium normal to the tagging stripes. See Fig. 14.1a–c for some examples. However, before it can be used in routine clinical evaluations, an imperative but challenging task is to automatically find the boundaries of the epicardium and the endocardium.

Segmentation in tagged MRI is difficult for several reasons. First, the boundaries are often obscured or corrupted by the nearby tagging lines, which makes the conventional edge-based segmentation method infeasible. Second, tagged MRI tends to increase the intensity contrast between the tagged and un-tagged tissues at the price of lowering the contrast between the myocardium and the blood. At the same time, the intensity of the myocardium and blood vary during the cardiac cycle due to the tag fading in the myocardium and being flushed away in the blood. Third, due to the short acquisition time, the tagged MR images have a relatively high level of noise. These factors make conventional edge or region-based segmentation techniques impractical. The last important reason is that, from the clinicians' point of view, or for the purpose of 3D modeling, *accurate* segmentation based solely on the MR image is usually not possible. For instance, for conventional clinical practice, the endocardial boundary should exclude the papillary muscles for the

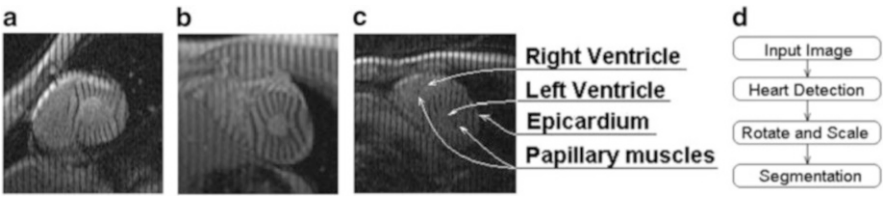

Fig. 14.1 (**a–c**) Some examples of tagged cardiac MRI images. The task of segmentation is to find the boundaries of the epicardium and endocardium (including the LV and RV and excluding the papillary muscles). (**d**) The framework of our segmentation method

purpose of easier analysis. However, in the MR images, the papillary muscles are often apparently connected with the endocardium and cannot be separated if only the image information is used. Thus, manual correction, or prior statistical shape knowledge is usually needed to improve the segmentation results.

There have been some efforts to achieve tagged MRI segmentation. In [21], grayscale morphological operations were used to find non-tagged blood-filled regions. Then they used thresholding and active contour methods to find the boundaries. In [11], a learning method with a coupled shape and intensity statistical model was proposed. In [21], morphological operations are sensitive to the complex image appearance and high level image noise, and the active contour method tends to get irregular shapes without a strong prior shape model. In [11], their intensity statistical model cannot capture the complex local texture features, which leads to inaccurate image forces.

In this chapter, in order to address the difficulties stated above, we propose a novel and fully automatic segmentation method based on three learning frameworks: (1) An active shape model (ASM) is used as the prior heart shape model. (2) A set of confidence-rated local boundary criteria are learned by Adaboost, a popular learning scheme, at landmark points of the shape model, using the appearance features in the nearby local regions. These criteria give the probability of the local region's center point being on the boundary and force their corresponding landmark points to move toward the direction of the highest probability regions. (3) An Adaboost detection method is used to initialize the segmentation's location, orientation and scale. The second component is the most essential contribution of our method. We abandon the usual edge or region-based methods because of the complicated boundary and region appearance in the tagged MRI. It is not feasible to designate one or a few edge or region rules to solve the complicated segmentation task. Instead, we try to use all possible information, such as the edges, the ridges, and the breaking points of tagging lines, to form a *complex rule*. It is apparent that at different locations on the heart boundary, this *complex rule* must be different, and our confidence in the *complex rule* varies too. It is impractical to manually set up each of these *complex rules* and weigh their confidence ratings. Therefore, we implement Adaboost to learn a set of rules and confidence ratings at each landmark point on the shape model. The first and the second frameworks are tightly coupled. The shape model deforms under the forces from Framework 2 while controlled and smoothed by Framework 1. To achieve fully automatic segmentation, in Framework 3 the detection method automatically provides an approximate position and size of the heart to initialize the segmentation step. See Fig. 14.1d for a complete illustration of the frameworks.

Before we implement the learning-based framework, we need to generate a large amount of accurately segmented contours used as the training data. A full set of conventional spatiotemporal (4D) tagged MRI consists of more than one thousand images. Segmenting every image manually and individually is a very

time-consuming and inefficient process. Therefore, we developed a semiautomatic segmentation system that spatiotemporally propagates and requires minimal manual interaction.

14.1.1 Training Data Generated From A Semiautomatic Segmentation System

Training data are usually obtained by manual delineation with a proper user interface. Semiautomatic method can vastly lower manual workload and improve the accuracy and robustness. To address the difficulty added by tagging lines, before the segmentation process, a tunable Gabor filter bank technique is first applied to remove the tagging lines and enhance the tag-patterned region [18]. Because the tag patterns in the blood are flushed out very soon after the initial tagging modulation, this tag removal technique actually enhances the blood-myocardium contrast and facilitates the following myocardium segmentation.

The 2D Gabor filter is basically a 2D Gaussian multiplied by a complex 2D sinusoid [5]. At time 1 of the tagged MR imaging process, when the tagging lines are initially straight and equally spaced, we set the initial parameters of the Gabor filter to equal the frequencies of the image's first harmonic peaks in the spectral domain. During a heart beat cycle, the tagging lines move along with the underlying myocardium, and the spacings and orientations of them change accordingly. We modify the parameters of the Gabor filter to fit these deformed tag patterns. The original un-tuned Gabor filter and the modified Gabor filters make up a tunable Gabor filter bank. The input tagged MR images are convolved with the tunable Gabor filters in the Fourier domain. Then the tag patterned region will result in high filtering response, and enhance the blood-myocardium contrast and facilitate myocardium segmentation. As shown in Fig. 14.2, the de-tagged images in mid-systolic phase make the boundary segmentation tasks easier.

At each pixel in the input image, we apply the tunable Gabor filter bank and find a set of optimal filter parameters that maximize the Gabor filter response. From the optimal parameters, we can learn the current pixel's relative distance with respect to the nearby tagging lines, and approximately the displacement of the underlying tissue over time. For conventional short axis (SA) tagged MRI sequences, we have two sets of data whose tagging lines are initially perpendicular to each other. When we combined them, we get the 2D deformation of the myocardium. Therefore, we only need to do myocardium segmentation at one time, then it can be temporally propagated to the neighboring time frames. Spatial propagation of the heart wall boundaries is a bit difficult due to the complex heart geometry and the topological changes of the boundaries at different positions of the heart. Our solution is segmenting a few key slices first, which represent the topologies of the rest of the slices. Then we let the key frames propagate to the remaining slices.

The semiautomatic segmentation step is based on a newly proposed deformable model, which we call "Metamorphs" [12]. The key advantage of the Metamorph

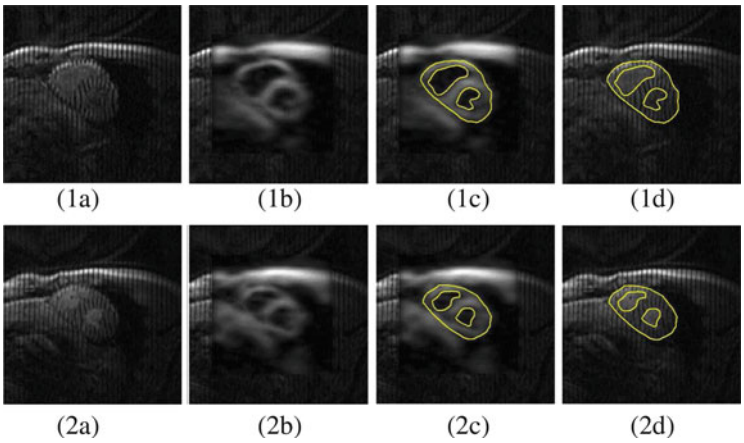

Fig. 14.2 Metamorphs segmentation on de-tagged images. (**1**) Segmentation at time 7, slice position 7. (**2**) Segmentation at time 7, slice position 10. (**a**) Original image. (**b**) Image with tags removed by Gabor filtering. (**c**) Cardiac contours segmented by Metamorphs on detagged image. (**d**) Contours projected on the original image

model is that it integrates both shape and interior texture and its dynamics are derived coherently from both boundary and region information in a common variational framework. These properties of Metamorphs make it more robust to image noise and artifacts than traditional shape-only deformable models.

The model deformations are efficiently parameterized using a space warping technique, the cubic B-spline-based free form deformations (FFD) [1, 26]. The essence of FFD is to deform an object by manipulating a regular control lattice F overlaid on its volumetric embedding space. In this paper, we consider an incremental free form deformations (IFFD) formulation using the cubic B-spline basis [10]. The interior intensity statistics of the models are captured using non-parametric kernel-based approximations, which can represent complex multi-modal distributions. Using this nonparametric approximation, the intensity distribution of the model interior gets updated automatically while the model deforms. When finding object boundaries in images, the dynamics of the Metamorph models are derived from an energy functional consisting of both edge/boundary energy terms and intensity/region energy terms.

We used Metamorph models to segment heart boundaries in tagged MR images. In Fig. 14.2, we show the Left Ventricle, Right Ventricle, and Epicardium segmentation using Metamorphs on de-tagged MR images. By having the tagging lines removed using Gabor filtering, a Metamorph model can get close to the heart wall boundary more rapidly. Then the model can be further refined on the original tagged image. The Metamorph model evolution is computationally efficient, due to our use of the nonparametric texture representation and FFD parameterization of the model deformations. For all the examples shown, the segmentation process takes <200 ms to converge on a 2 Ghz PC station.

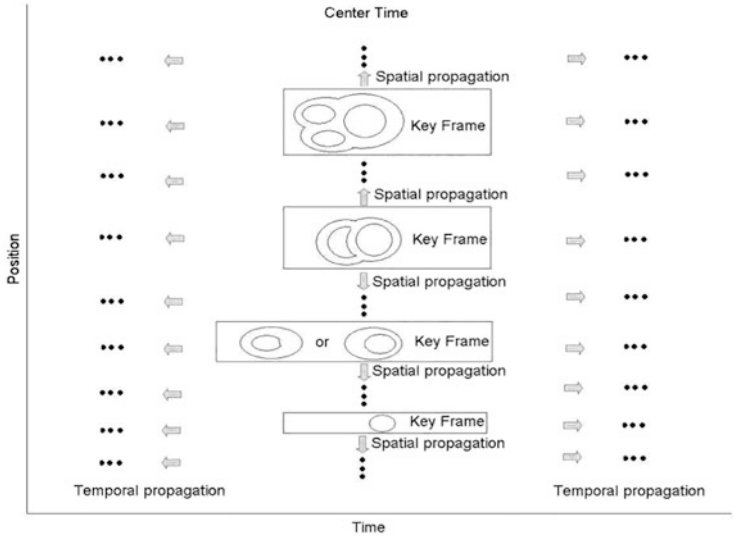

Fig. 14.3 The framework of our automated segmentation in 4D spatiotemporal MRI-tagged images. We start at a center time when the *tag lines* are flushed away in the blood area while they remain clear in the myocardium. Boundary segmentation is done in several key frames on the de-tagged images before the boundary contours are spatially propagated to the other positions. Then at each position, the boundaries are temporally propagated to other times

14.1.1.1 Integration and the Semiautomatic System

We integrate the above two major techniques, the tunable Gabor filter bank and the Metamorphs segmentation, to construct our spatiotemporal integrated MR analysis system. By using the two techniques in a complementary manner, exploiting specific domain knowledge about the heart anatomy and temporal characteristics of the tagged MR images, we can achieve efficient, robust segmentation with minimal user interaction. The algorithm consists of the following main steps (Fig. 14.3).

1. Tag removal for images at the mid-systolic phase. Given a 4D spatiotemporal tagged MR image data set of the heart, we start by filtering using a tunable Gabor filter bank on images of a 3D volume that corresponds to a particular time in the middle of the systolic phase, which we term *center time*. For a typical data set in which the systolic phase is divided into 13 time intervals, we apply the Gabor filtering on images at time 7, when tag patterns in the endocardium are flushed out by blood but tag lines in the myocardium are clearly visible.
2. Metamorphs segmentation using the de-tagged images. Given the de-tagged Gabor response images at time 7, we use Metamorphs to segment the cardiac contours including the epicardium, the LV and RV endocardium. The Metamorph models can be initialized far-away from the object boundary and efficiently converge to an optimal solution. For each image, we first segment the LV and

RV endocardium. To do this, the user initializes a circular model by clicking one point (the seed point) inside the object of interest, then the surrounding region intensity statistics and the gradient information automatically drive the model to converge to the endocardium boundaries. We then automatically initialize a Metamorphs model for the epicardial contour by merging the endocardial contours and expanding the interior volume according to myocardium thickness statistics. The model is then allowed to evolve and converge to the epicardium boundary.

3. Spatial propagation at the mid-systolic center time. At the mid-systolic phase, we do the segmentation at several key frames which represent the topologies of the rest of the frames, then let the segmented contours propagate to their nearby frames. In short axis cardiac MR images, from the apex to the base, the topology of the boundaries goes through the following variations: (1) one epicardium; (2) one epicardium and one LV endocardium (in some cases of the RV hypertrophy patients, one epicardium and one RV endocardium are also possible); (3) one epicardium, one LV endocardium and one RV endocardium; (4) one epicardium, one LV endocardium and two RV endocardium. The key frames consist of one center frame of the third topology and three transition frames. This spatial propagation actually provides a quick initialization method (rather than manually clicking the seed points as mentioned in step 2) for the rest of the non-key frames from the key frames.
4. Boundary tracking using tunable Gabor filters over time. Once we have segmented the cardiac contours at time 7, we keep tracking the motion of the myocardium and the segmented contours over time. This temporal propagation of the cardiac contours significantly reduces segmentation manual workload, since it enables us to do supervised segmentation at only one time, then fully automated segmentation of the complete 4D data set can be achieved. It also improves segmentation accuracy because we capture the overall trend in heart deformation more accurately by taking into account the temporal connection between segmented boundaries.
5. Boundary refinement by manual. In practice, we provide the manual correction option to doctors during the whole segmentation process to ensure satisfiable results.

This 4D segmentation system is developed in a Matlab 6.5 GUI environment. The user need to load in the raw MRI data of the short axis and long axis volumes first (Fig. 14.4-1a). Then the user is allowed to examine the whole data sets, which consist of two short axes and one long axis, and determine the slice index of the center time (Fig. 14.4-1b, 1c). The tag removal step is done on the 3D volume at the center time (Fig. 14.4-2a). Then the user has an option to determine the indices of the key frames and do Metamorphs segmentation on these key frames (Fig. 14.4-2b). The segmented contours are propagated spatially (optional) and then temporally. Practically the spatial propagation step is optional because for most clinical analysis one typical slice is enough unless a fully 4D model is required. Manual interaction is always available during the whole segmentation and propagation process to make corrections in time (Fig. 14.4-2c).

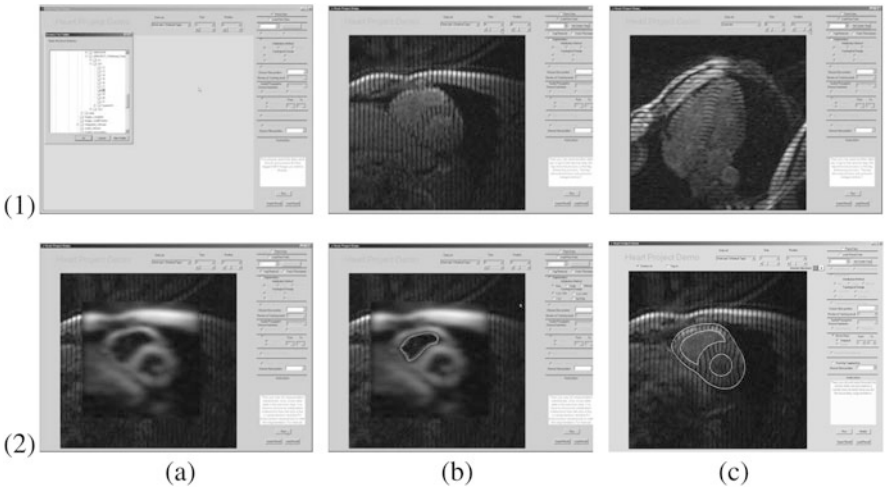

Fig. 14.4 Screen snapshots of our segmentation and tracking system. (**1a**) Read in the SA and LA volumes. (**1b,1c**) Examine the data sets. (**2a**) De-tagged image at the center time. (**2b**) Metamorphs segmentation based on de-tagged images. (**2c**) Segmentation results. The papillary muscle is excluded from the myocardium by manual interaction

14.1.2 Segmentation Based on ASM and Local Appearance Features Learning Using Adaboost

After we collected enough amount of training data from the previous system, we are able to perform learning methods to extract prior knowledge, such as a statistical shape prior, and image local features, to direct the future fully automatic segmentation. There has been some previous research on ASM segmentation methods based on local features modeling. In [9], a statistical analysis was performed, which used sequential feature forward and backward selection to find the set of optimal local features. In [13], an EM algorithm was used to select Gabor wavelet-based local features. These two methods tried to select a small number of features, which is impractical to represent complicated local textures such as in tagged MRI. In [16], a simple Adaboost learning method was proposed to find the optimal edge features. This method didn't make full use of the local textures and didn't differentiate each landmark point's confidence level. In our method, similarly using Adaboost, our main contributions are: the ASM deforms based on a more *complex* and robust rule, which is learned from the local appearance, not only of the edges, but also ridges and tagging line breakpoints. In this way we get a better representation of the local appearance of the tagged MRI. At the same time, we derive the confidence rating of each landmark point from their Adaboost testing error rates and use these confidence ratings to weigh the image forces on each landmark point. In this way the global shape is affected more by the *more confident* points and we eliminate the possible error forces generated from the *less confident* points.

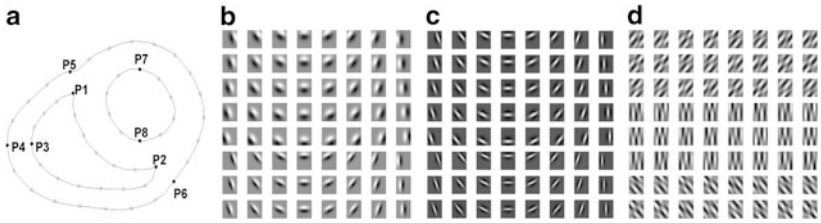

Fig. 14.5 (**a**) shows the automatic method used to place the landmark points. (**b–d**) Are the sample sets of feature filters: (**b**) are the derivatives of Gaussian used for edge detection, (**c**) are the second derivatives of Gaussian used for ridge detection, and (**d**) are the half-reversed Gabor filters used for tag line breakpoint detection

14.1.2.1 ASM, The Shape Model

Since the shape of the mid portion of the heart in short axis (SA) images is consistent and topologically fixed (one left ventricle (LV) and one right ventricle (RV)), it is reasonable to implement an ASM [2] to represent the desired boundary contours.

We choose training data from SA images with different tagging line orientations, such as (0° and 90°, or −45° and 45°), and slightly different tag spacings. Each data set included images acquired at phases through systole into early diastole, and at positions along the axis of the LV, from near the apex to near the base, but without topological changes. Segmented contours were centered and scaled to a uniform size. Landmark points were placed automatically by finding key points with specific geometric characteristics. As shown in Fig. 14.5a, the black points are the key points, which were determined by the curvatures and positions along the contours. For instance, $P1$ and $P2$ are the highest curvature points of the RV; $P7$ and $P8$ are on opposite sides of the center axis of the LV. Then, fixed numbers of other points are equally placed in between. In this way, the landmark points were registered to the corresponding locations on the contours. Here, we used 50 points to represent the shape.

For each set of contours, the 50 landmark points (x_i, y_i) were reshaped to form a shape vector $X = (x_1, x_2, \ldots, x_{50}, y_1, y_2, \ldots, y_{50})^T$. Then Principal Component Analysis was applied and the modes of shape variation were found. Any heart shape can be approximately modeled by $X = \bar{X} + Pb$, where $\bar{X}$ is the mean shape vector, P is the matrix of shape variations, and b is the vector of shape parameters weighting the shape variations.

After we find the image forces at each landmark point, as in Sect. 14.1.2.2, the ASM evolves iteratively. In each iteration, the model deforms under the influence of the image forces to a new location; the image forces are then calculated at the new locations before the next iteration.

14.1.2.2 Segmentation Via Learning Boundary Criteria Using Adaboost

To capture the local appearance characteristics, we designed three different kinds of steerable filters. We use the derivatives of a 2D Gaussian to capture the edges, we use the second order derivatives of a 2D Gaussian to capture the ridges, and we use half-reversed 2D Gabor filters [3] to capture the tagging line breakpoints.

Assume $G = G((x-x_0)\cos(\theta),(y-y_0)\sin(\theta),\sigma_x,\sigma_y)$ is an asymmetric 2D Gaussian, with effective widths σ_x and σ_y, a translation of (x_0,y_0) and a rotation of θ. We set the derivative of G to have the same orientation as G: $G' = G_x\cos(\theta) + G_y\sin(\theta)$.

The second derivative of a Gaussian can be approximated as the difference of two Gaussians with different σ. We fix σ_x as the long axis of the 2D Gaussians, and set $\sigma_{y2} > \sigma_{y1}$. Thus, $G'' = G(\sigma_{y1}) - G(\sigma_{y2})$.

In the previous two equations, we set $x_0 = 0$, and tune y_0, θ, σ_x, σ_y, σ_{y1}, and σ_{y2} to generate the desired filters.

The half-reversed 2D Gabor filters are defined as a 2D sine wave multiplied with the 2D derivative of a Gaussian:

$$F = G'(x,y)\cdot\mathbb{R}\{e^{-j[\phi+2\pi(Ux+Vy)]}\} \tag{14.1}$$

where G' is the derivative of a 2D Gaussian. U and V are the frequencies of the 2D sine wave, $\psi = \arctan(V/U)$ is the orientation angle of the sine wave, and ϕ is the phase shift. We set $x_0 = 0$, $\sigma_x = \sigma_y = \sigma$, $-45^\circ \le \psi - \theta \le 45^\circ$, and tune y_0, θ, σ, ϕ, U and V to generate the desired filters.

For a 15×15 sized window, we designed 1,840 filters in total. See Fig. 14.5b–d for some sample filters.

In the learning subsection, each training image is scaled proportionally to the scaling of its contours. At each landmark point of the contours, a small window (15×15) around it was cut out as a positive appearance training sample for this particular landmark point. Then along the normal of the contour, on each side of the point, we cut out two 15×15-sized windows as negative appearance training samples for this particular landmark point. Thus for each training image, at a particular landmark point, we got one positive sample and four negative samples (shown in Fig. 14.6a). We also randomly selected a few common negative samples outside the heart or inside the blood area, which are suitable for every landmark point. For image contrast consistency, every sample was histogram equalized.

The function of the Adaboost algorithm [7, 24] is to classify the positive training samples from the negative ones by selecting a small number of important features from a huge potential feature set and creating a weighted combination of them to use as an accurate strong classifier. During the boosting process, each iteration selects one feature from the potential features pool and combines it with the existing classifier that was obtained in the previous iterations. After many iterations, the combination of the selected important features can become a strong classifier with

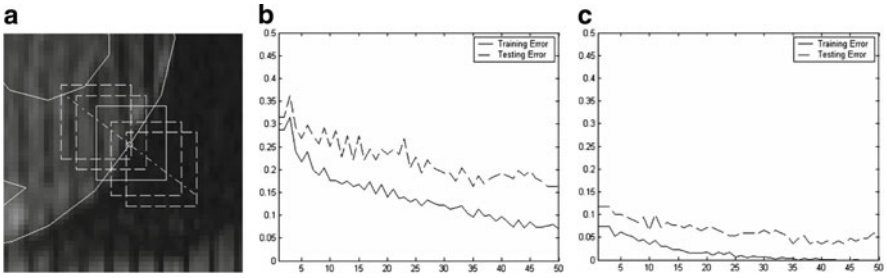

Fig. 14.6 (**a**) Shows the method of setting the training data. The *solid box* is the positive sample around the landmark points. The four *dashed line* boxes along the normal are the negative samples. This way of setting the negative samples is chosen to make the classifier more adaptive to the particular landmark position. (**b**) and (**c**) show the training error (*solid lines*) and testing error (*dash lines*) of two landmark points versus Adaboost iteration times. (**b**) is a point on the LV, (**c**) is a point on the Epi. Note how the training and testing error decrease as Adaboost iterates. Also note the testing error of (**b**) is higher than (**c**): we are more confident of landmark point (**c**)'s classification result

high accuracy. The output of the strong classifier is the weighted summation of the outputs of each of its each selected features, or, the weak classifiers: $F = \Sigma_t \alpha_t h_t(x)$, where α are the weights of weak classifiers, and h are the outputs of the weak classifiers.

We call F the boundary criterion. When $F > 0$, Adaboost classifies the point as being on the boundary. When $F < 0$, the point is classified as off the boundary. Even when the strong classifier consists of a large number of individual features, Adaboost encounters relatively few overfitting problems [25]. We divided the whole sample set into one training set and one testing set. The function of the testing set is critical. It gives a performance measure and a confidence level that tells us how much we should trust its classification result. Figure 14.6b, c shows the learning error curve versus the boosting iteration numbers at two selected landmark points. Note that every landmark point i has its own α, h, and F_i.

In the segmentation stage, we first select an initial location and scale, and then overlay the mean shape $\bar{X}$, which is obtained from ASM, onto the task image. In Sect. 14.1.3 we describe an automatic initialization method.

At a selected landmark point i on the shape model, we select several equally spaced points along the normal of the contour on both sides of i, and use their F values to examine the corresponding windows centered on these points. In [25], a logistic function was suggested to estimate the relative boundary probabilities:

$$Pr(y = +1|x) = \frac{e^{F(x)}}{e^{F(x)} + e^{-F(x)}} \tag{14.2}$$

We find a point j whose test window has the highest probability of being on the heart boundary. Thus an image force $\mathbf{f}$ should push the current landmark point i toward j.

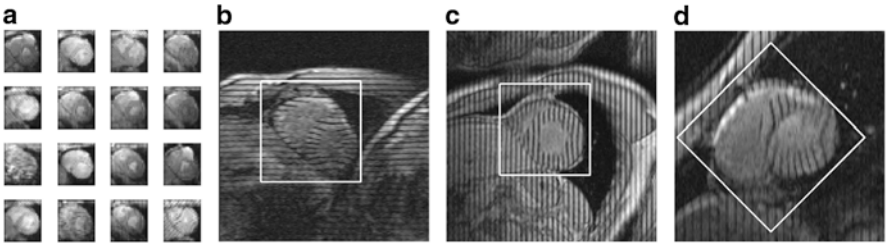

Fig. 14.7 (**a**) shows a few samples of the training data. (**b**), (**c**), and (**d**) are three detection results. For image (**d**), the image was rotated by a set of discrete angles before the detection, and the final detection is of the highest probability among all the discrete angles tested

Recall that, as discussed in the previous subsubsection, Adaboost gives the errors of the testing data e_i. We define the confidence rating as:

$$c_i = \ln \frac{1}{e_i}; \tag{14.3}$$

Intuitively, when c_i is big, we trust its classification and increase the image force $\mathbf{f}$, and conversely. Thus, we define the image force at landmark point i as:

$$\mathbf{f} = \mu \cdot \frac{[\mathbf{x}(j) - \mathbf{x}(i)] \cdot c(i)}{||\mathbf{x}(j) - \mathbf{x}(i)||_2} \tag{14.4}$$

where μ is a scale as a small step size.

The detailed algorithm to update the parameters of the ASM model with the image force $\mathbf{f}$ can be found in [2].

14.1.3 Heart Detection Based on Adaboost Learning

The heart detection algorithm used is influenced by the Adaboost face detection algorithm developed in [30]. The reason we adapt a face detection method is that these two problems are closely related. Often, there are marked variations between different face images, which come from different facial appearance, lighting, expression, etc. In heart detection, we have the similar challenges: the heart images have different tag patterns, shape, position, phase, etc.

We use the same Haar wavelet features as in [30]. The training data contained 297 manually cropped heart images and 459 randomly selected non-heart images. The testing data consisted of 41 heart images and 321 non-heart images. These data were resized to 24×24 pixels and contrast equalized. Adaboost training gave a strong classifier by combining 50 weak features. For an input task image, the detection method searched every square window over the image and found a window with the highest probability as the final detection. If we rotate the task image by a set of discrete angles before the detection procedure, and compare the probabilities across the discrete angles, we are also able to detect hearts in rotated images (see Fig. 14.7).

14.1.4 Representative Experimental Results and Validation

We applied our segmentation method to three data sets, one from the same subject and with the same imaging settings as the training data (but excluding the training data), and the other two novel data sets from two different subjects and with slightly different imaging settings. The three data sets each contained tagged MR images with different phases, positions and tagging orientations. Each task image was rotated and scaled to contain a 80×80-pixel-sized chest-on-top heart, using the detection method before the segmentation. Each segmentation took 30 iterations to converge. Our experiment was coded in Matlab 6.5 and run on a PC with dual Xeon 3.0G CPUs and 2G memory. The whole learning process took about 20 hours. The segmentation process of one heart took 120 s on average. See Fig. 14.8 for representative results.

For validation, we used the semiautomatically segmented contours as the ground truth for the data sets as shown in the first and second rows. For the other data set, because we don't have segmented ground truth, we used cross validation, since we know that at the same position and phase, the heart shapes in the vertical-tagged and horizontal-tagged images should be similar. We denote the ground truth contours as T and our automatic segmentation contours as S. We defined the average error distance as $\bar{D}_{\text{error}} = \text{mean}_{s_i \in S}(\min ||T - s_i||_2)$. Similarly the cross distance is defined as $\bar{D}_{\text{cross}} = \text{mean}_{s_i^{\text{vertical}} \in S^{\text{vertical}}}(\min ||S^{\text{horizontal}} - s_i^{\text{vertical}}||_2)$. In a 80×80 pixel-sized heart, the average error distances between the automatically segmented contours and the ground truth for the first data set were: $\bar{D}_{\text{error}}(\text{LV}) = 1.12$ pixels, $\bar{D}_{\text{error}}(\text{RV}) = 1.11$ pixels, $\bar{D}_{\text{error}}(\text{Epi}) = 0.98$ pixels. For the second data set, $\bar{D}_{\text{error}}(\text{LV}) = 1.74$ pixels, $\bar{D}_{\text{error}}(\text{RV}) = 2.05$ pixels, $\bar{D}_{\text{error}}(\text{Epi}) = 1.33$ pixels. In the third data set, the cross distances are: $\bar{D}_{\text{cross}}(\text{LV}) = 2.39$ pixels, $\bar{D}_{\text{cross}}(\text{RV}) = 1.40$ pixels, $\bar{D}_{\text{cross}}(\text{Epi}) = 1.94$ pixels. The larger distance in the cross validation arises in part from underlying mis-registration between the (separately acquired) horizontal and vertical images. Thus, the true discrepancy due to the segmentation should be smaller. From the above quantitative results, we find that for a normal-sized adult human heart, the accuracy of our segmentation method achieves an average error distance of less than 2 mm. The cross validation results of the third data set suggest that our method is very robust as well.

14.2 3D Segmentation and Blood Flow Simulation

Following a heart attack or the development of some cardiovascular diseases, the movement of the heart walls during the cardiac cycle may change. This affects the motion of blood through the heart, potentially leading to an increased risk of thrombus. While Doppler ultrasound and MRI can be used to monitor valvular blood flow, the image resolutions are low and they cannot capture the interactions between the highly complex heart wall and the blood flow. For this reason, with the rapid development of high-resolution cardiac CT, patient-specific blood flow simulation can provide a useful tool for the study of cardiac blood flow.

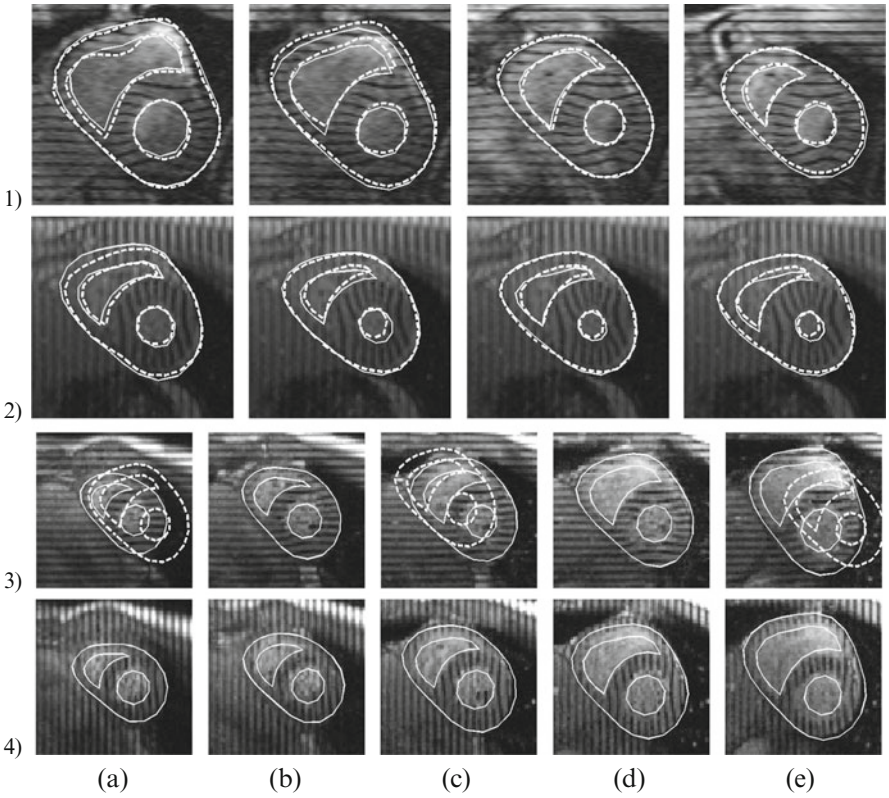

Fig. 14.8 Representative segmentation results. For better representation, the images in the *first row* vary in position and remain at the same phase, while the images in the *second row* vary in phase but remain at the same position. The *solid contours* are from our automatic segmentation method; the *dashed contours* are semiautomatic. Notice that the papillary muscles in LV are excluded from the endocardium. Semiautomatic results are not available for the *third* and *fourth rows*, so we compare our segmentation results between the horizontal and vertical tagged images that are at same position and phase. Qualitatively, the contours are quite consistent, allowing for possible misregistration between the nominally corresponding image sets. In (3a), (3c), and (3e) the *dashed contours* are testing examples of poor initializations, while the final contours are *solid*. Although the initialization is far away from the target, the shape model moves and converges well to the target

Recently, Mihalef et al. [20] used smoothed 4D CT data to simulate left ventricular blood flow and compared the flow through the aortic valve in a healthy heart and two diseased hearts. However, the models derived from CT data in [20] were too highly smoothed to capture the local structural details and were not useful for understanding the true interactions between the blood flow and the walls (Fig. 14.9).

Later, in [15], more accurate heart models were achieved by generating a triangular mesh using initial median filtering and isosurfacing of the CT data at

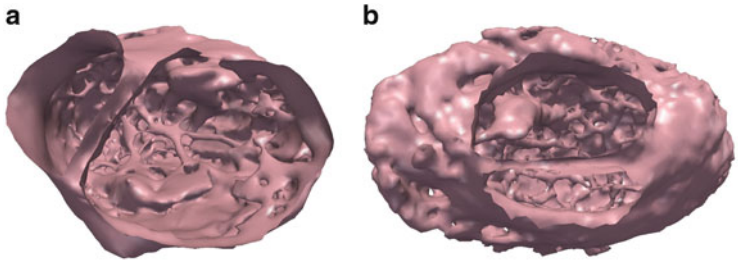

Fig. 14.9 Meshes reconstructed from CT data (valves removed). (**a**) Healthy heart (**b**) diseased heart

mid-diastole. Then, motion was transferred to this model from the smooth mesh motion obtained from the same CT data to create the animation. This allowed for more realistic features to be present on the heart walls in the simulation, including the papillary muscles and some parts of the trabeculae. However, while this approach was an improvement from the smooth-wall assumption, the trabeculae were missing details and did not move accurately.

Earlier work in blood flow simulation used less refined models. For example, [14] was the first to extract boundaries from MRI data to perform patient-specific blood flow simulations. Later, Long et al. [17] and Saber et al. [23] used simple models of the left side of the heart, with smooth ventricular walls, and imposed boundary conditions in the valve regions.

In this paper, we use an even further improved method of generating and moving the mesh to capture these smaller details (Fig. 14.9) and generate a more accurate simulation (Fig 14.12). Our approach estimates the predefined motion for the valves, whose asynchronous opening and closing provides a simple geometric mechanism for taking care of those boundary conditions. To the best of our knowledge, contrary to all previous methods, we are able to visualize blood flow in unprecedented detail.

14.2.1 Heart Model Reconstruction

The heart models are reconstructed using a deformable model method. A semiautomatic segmentation is used to get the initial segmentation from high resolution CT data for an initial (3D) frame of data. This semiautomatic segmentation is time consuming and tedious, so it is not efficient to use it for segmentation of all the frames. The initial high resolution mesh model is generated as an isosurface of the segmentation. Geometric processing is then applied to the initial model to get a smooth and regular mesh with an appropriate number of vertices. Based on the initial model from one time frame, our method deforms it towards the boundaries on the other frames. During the formation, the topology of the model is kept unchanged. We can also get the one-to-one correspondence between frames, as a requirement

for the fluid simulator in later processes. These novel and powerful methods can extract the full 3D surfaces of these complex anatomical structures. The results have been validated based on the ground truth segmented by multiple clinical experts. Valves are hard to be captured in CT images such that valve models are added to the heart meshes in the sequence. In the following subsections we write the details of our work.

14.2.1.1 CT Data Acquisition

The CT images were acquired on a 320-MSCT scanner (Toshiba Aquilion ONE, Toshiba Medical Systems Corporation) using contrast agent. This advanced diagnostic imaging system is a dynamic volume CT scanner that captures a whole-heart scan in a single rotation, and achieves an isotropic 0.3 mm volumetric resolution with less motion artifact than conventional 64-MSCT scanners. A conventional contrast-enhanced CT angiography protocol was adapted to acquire the CT data in this work. After the intravenous injection of the contrast agent, the 3D+time CT data were acquired in a single heart beat cycle when the contrast agent was circulated to the left ventricle and aorta. After acquisition, 3D images were reconstructed at ten time phases in between the R-to-R waves using ECG gating. The acquired isotropic data had an in-plane dimension of 512 by 512 pixels, with an effective atrio-ventricular region measuring about 300^3 pixels.

14.2.1.2 Reconstruction Framework

We propose a framework to reconstruct the cardiac model. This framework includes: initial model construction, deformable model-based segmentation, and interpolation between time frames. The initial model is generated using snake segmentation on one time frame of the CT image. The initial model needs geometry processing, such as decimating, detail-preserving smoothing, and isotropic remeshing to get high-quality meshes. Based on the initial model, segmentation of the rest of the CT images is automatically performed using the deformable model. The segmentation of a sequence of CT images is interpolated in time to get a higher effective temporal resolution.

14.2.1.3 Initial Model Construction

The model initialization framework is illustrated in Fig. 14.10. While generating the initial model, a flexible method is preferred to provide more freedom for users. Different thresholds could be set for different part of the heart. We use a semiautomatic segmentation method to get the initial model [32]. This segmentation process is very time consuming and could not be used to segment all frames. It needs a lot of tedious work during the model initialization. However, once this model has been generated, it is used to segment the rest of other frames automatically.

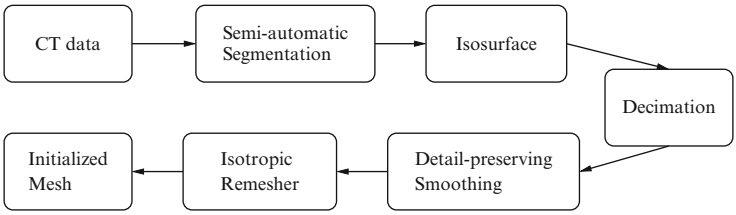

Fig. 14.10 Initial model construction

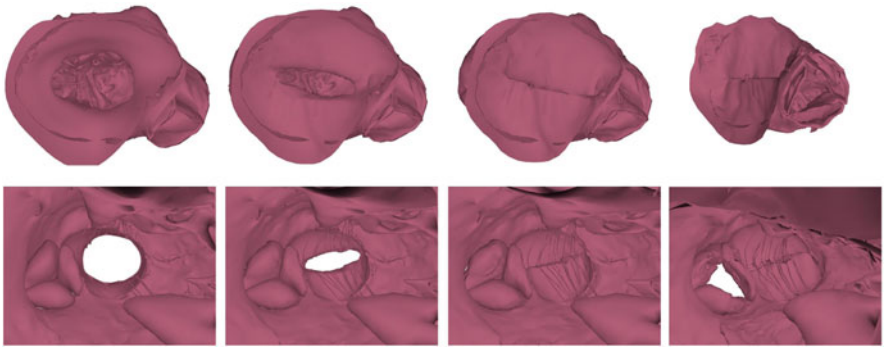

Fig. 14.11 Outside and inside views of the valves at various stages of the cardiac cycle. The mitral valve is open at first. Gradually the mitral valves closes and the aortic valves opens

Isosurface detection is applied to generate the model mesh from the first segmented result. However, the resulting mesh is usually bulky, noisy, and irregular. To get a better initialization model, some geometric processing should be done on that mesh, such as decimating, detail-preserving smoothing, and isotropic remeshing. Such geometric processing, which leads to high-quality meshes, is essential to later model deformation.

The initial model is too large to readily modify, so we need to decimate the mesh to an appropriate size. The desirable number of vertices is given as a constraint. Edge collapses, which simply collapse one vertex into one of its neighbors, are performed during decimation. Some error metrics are used to decide the priority of the edge collapses. Finally, we get a mesh with much fewer vertices, but that still retains most of the shape details. The meshes have been decimated to about 20,000 vertices. Those are complex enough to capture the fine details of the heart.

Detail-preserving smoothing is performed after decimation. The smoothing is restricted to the tangential direction. Instead of moving each vertex towards the centroid of its neighbors, which would smooth out the shape details and sharp features, detail-preserving smoothing ensures higher quality meshes without losing details.

Isotropic remeshing is important for the mesh quality. In irregular meshes, the vertices with high valences exert strong internal forces to drag other vertices, which can cause unrealistic results in deformable models [28]. An incremental isotropic

remeshing technique is used to remesh the given triangular mesh so that all edges have approximately the same target edge length and the triangles are as regular as possible. The target edge length is set to be the mean of the current edge lengths. Edge length thresholds are set around the target edge length. During the incremental isotropic remeshing process, edges longer than the higher edge bound are split until all edges are shorter than the threshold; shorter edges are collapsed if collapsing does not result in new edges longer than the higher threshold; edges are flipped to equal valences; vertices are moved to new positions to get regular triangle meshes; and finally vertices are projected back to the original surfaces to keep the shape unchanged. This process would generally be iterated several times to get the final results.

After all these geometric processing steps, we finally get a high-quality triangular mesh with an appropriate number of vertices. This mesh is used as an initialization for other frames.

14.2.1.4 Deformable Model-Based Segmentation

To get the segmentation of the rest frames as well as one-to-one correspondence between frames, we deform our initial model to the boundaries during tracking. To do so, we define an energy function, including an *external energy*, derived from the image so that it is smaller at the boundaries, and a *model energy*, which reflects the differences between the original model and the deformed model. By minimizing the energy function, it drags the model towards the boundaries and keeps the shape of the model unchanged during deformation.

Given a gray level image $I(x,y)$, viewed as a function of continuous position variables (x,y), the model M_{t-1} derived from the previous frame is used to fit the current frame M_t. The energy function we want to minimize is defined as follows:

$$E(M_t, I_t, M_{t-1}) = E_{\text{ext}}(M_t, I_t) + E_{\text{model}}(M_t, M_{t-1}). \tag{14.5}$$

The external energy E_{ext} is designed to move the deformable model towards object boundaries.

$$E_{\text{ext}}(M_t, I_t) = -|\nabla I|^2, \tag{14.6}$$

where ∇ is the gradient operator.

The model energy is defined as the differences of *vertex normals* and *attribute vectors*. An attribute vector is attached to each vertex of the model [27], which reflects the geometric structure of the model from a local to global level. In 3D, for a particular vertex V_i, each attribute is defined as the volume of a tetrahedron on that vertex. The other three vertices form the tetrahedron are randomly chosen from the lth level neighborhood of V_i. Smaller tetrahedrons reflect the local structure near a vertex while larger tetrahedrons reflect a more global information around a vertex. The attribute vector, if sufficient enough, uniquely characterizes different parts of a surface of a boundary.

The volume of a tetrahedron is defined as $f_l(V_i)$. The attribute vector on a vertex is defined as:

$$F(V_i) = [f_1(V_i), f_2(V_i), \ldots, f_{R(V_i)}(V_i)], \tag{14.7}$$

where $R(V_i)$ is the neighborhood layers we want to use around V_i.

As we elaborated earlier in this subsection, the model energy term reflects the differences of vertex normals and attribute vectors between the original model and the deformed model.

$$E_{\text{model}}(M_t, M_{t-1}) = \sum_{i=1}^{N} (\alpha(n_{t,i} - n_{t-1,i})^2 + \sum_{l=1}^{R(V_i)} \delta_l (f_{t,l}(V_i) - f_{t-1,l}(V_i))^2), \tag{14.8}$$

where $f_{t,l}(V_i)$ and $f_{t-1,l}(V_i)$ are components of attribute vectors of the model and deformed model at vertex V_i, respectively. α Determines the importance of the vertex normals compared to the attribute vectors. δ_l Here denotes the importance of the lth neighborhood layers. $R(V_i)$ is the number of neighborhood layers around vertex V_i.

A greedy algorithm is used here to minimize the energy function. The proposed algorithm is iterative. During each iteration, the first step is to minimize the external energy, moving vertices towards the minimum gradient of the image; the second step is to minimize the model energy; a neighborhood of a vertex has been examined and the point in the neighborhood with the minimum model energy would be chosen as the new location of the vertex. The iterations continue until the energy converges. While this greedy algorithm might fall into a local minimum, the experiments show satisfactory results.

During the deformation, we suggest moving a surface segment as a whole, rather than a single vertex. This would avoid this risk of getting trapped in a local minimum and also speed up the convergence. Let V_i be the vertex to be deformed during a particular iteration. The first to $R(V_i)$th neighborhood layers are about to move together as a surface segment. Suppose V_i is to move to $V_i + \Delta$ as a tentative position. Then the new position of each vertex $nbr_{l,m}(V_i)$, the mth vertex on lth neighborhood layer, is set to move to

$$nbr_{l,m}(V_i) + \Delta \cdot \exp\left(-\frac{l^2}{2\delta^2}\right), \tag{14.9}$$

where δ is a parameter determining the locality of the transformation. We make the deformation unchanged on the boundary of the surface segment, such that the continuity has been maintained.

The parameter $R(V_i)$ that determines the locality if the deformation is chosen to be large in the initial iteration and is then gradually reduced to 1. Therefore, initially there are more vertices involved in the deformation. More global features are used in deformation. In later states, more local deformations are performed.

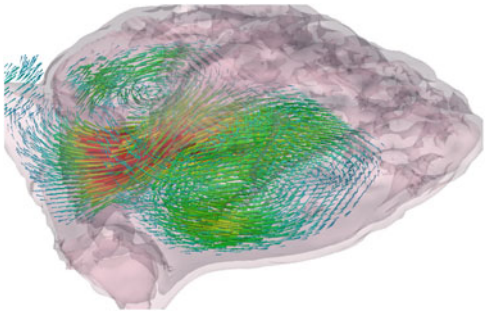

Fig. 14.12 Visualization of blood flow from outside heart during diastole

14.2.1.5 Valves Deformation and Interpolation

The aortic and mitral valves are thin and move fast, the CT data are not currently able to capture these details at all frames. So, we need a way to add previously created 3D models of the valves to each mesh in the sequence, and have them open and close at the correct times. We start by fitting both valve models to the first mesh, in both their open and closed states that can be seen in CT data. Upon completion, we have four new meshes (open mitral, closed mitral, open aortic, closed aortic), which each perfectly line up to their correct position in the first mesh in the sequence.

We seek to have similar collections of four properly fitted valve meshes in their opened and closed states for each frame in the sequence. Since the heart moves considerably in the course of the cardiac cycle, we now need a way to automatically and realistically move the valves along with the rest of the heart, so that there are no improper holes or overlap. The valves are deformed according to the following strategy: First, the part of the valves connected to the heart are deformed together with the heart movements. Then the already deformed valves would drag the rest to the appropriate positions.

Now, for each frame in our sequence, we have both opened and closed mitral and aortic valves that are correctly fitted. We next must determine which open/close state each valve must be set at for each frame. We know that in the normal cardiac cycle, the mitral valve is open during diastole, aortic valve is open during systole, and both valves are closed for a very short time between these stages. Therefore, it is simple to decide on each frame whether or not the valves are open or closed (Fig. 14.11).

We now have ten meshes with share one-to-one correspondence, and that have fitted valves that open and close at the correct frames. To perform an accurate simulation, we desire more intermediate frames. While we could simply use linear interpolation to determine how each vertex moves from one frame to the next, we found that the movement appears unnatural and undesirable. So, we instead use periodic cubic spline interpolation, achieving far better results. We chose to generate a total of 50 meshes for the full animation. With this, we are ready to perform the fluid simulation.

14.2.2 Fluid Simulation

The motion of an incompressible fluid is governed by the laws of conservation of momentum and mass. These two laws are modeled by the Navier–Stokes equations

$$\rho\left(\frac{\partial u}{\partial t}+u\cdot\nabla u\right)=-\nabla P+\mu\nabla^2 u,$$

$$\nabla\cdot u=0.$$

Here, ρ is the fluid density, u is the 3D velocity vector field, P is the scalar pressure field, and μ is the coefficient of viscosity. The first equation enforces conservation of momentum. The second equation states that the divergence of velocity is zero everywhere to model that there are exist sources or sinks anywhere in the flow, conserving mass.

Foster and Metaxas [6] were the first to develop a very fast method of solving the Navier–Stokes equations for graphics applications. They did so by applying a staggered grid across the domain and explicitly solving for the three components of velocity at the cell faces. They then used successive over-relaxation to solve for pressure and correct the velocities to maintain incompressibility.

Our fluid–solid interaction system uses a "boundary immersed in a Cartesian grid formulation," allowing for an easy treatment of complex geometries embedded in the computational domain, which can be especially advantageous when dealing with moving boundaries. Recent work that employs such a formulation is [31]. It applies the formulation of Sussman [29] to both graphics and medical simulations. Very recently de Zélicourt et al. [4] implemented the approach of Gilmanov and Sotiropoulos [8] to obtain a system that can efficiently deal with rather complex geometric data like a system of blood vessels.

The 3D mesh we generate from CT data is represented by a marker level set (MLS), introduced and validated in [19]. Here, markers are placed on the boundary and are used to correct the level set at every time step. Since markers are only placed on the surface, MLS has been proven to be more efficient and significantly more accurate for complex boundaries. Additionally, our specific solver achieves efficiency by implementing an adaptive mesh refinement approach.

The heart models used here are embedded in a computational mesh of 100^3 cells on which the full Navier–Stokes equations with a viscous component are solved using finite difference method. The blood is modeled as a Newtonian fluid, with viscosity set at 4 mPa s and density set at 1,060 kg/m^3, which are physiologically accepted values for a normal human heart. The heart geometric model is fed to the solver as a discrete set of meshes with point correspondences, which allows for easy temporal interpolation and also obtaining the velocity of the heart mesh at every point in time. The heart mesh and its velocity are rasterized onto the Eulerian grid as a MLS and an Eulerian velocity, respectively. The MLS and the velocity are used to impose the appropriate boundary conditions in the fluid solver. A simulation of two complete cardiac cycles takes about four days on a machine with a Core 2 Quad processor and 8 GB of RAM.

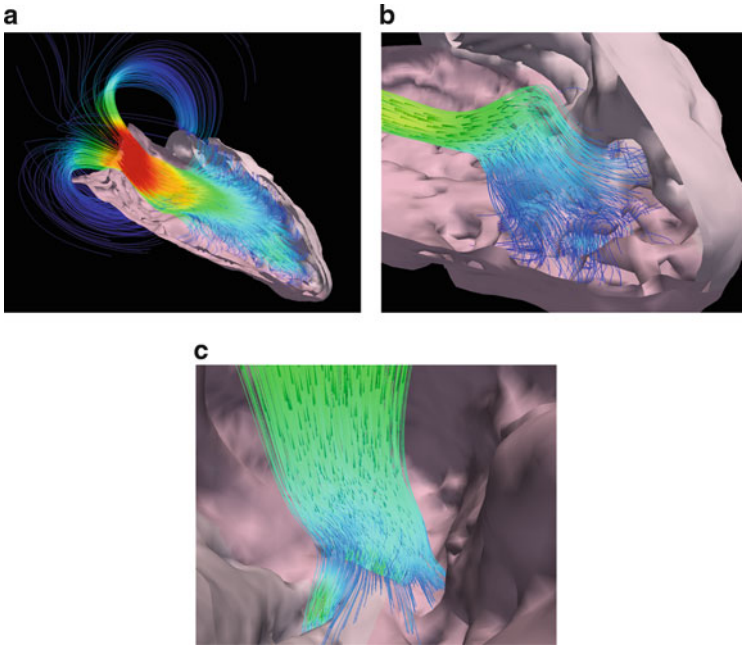

Fig. 14.13 Visualization of streamlines within the healthy heart. (**a**) Streamlines of cardiac blood flow during diastole. (**b**) Blood flow streamlines near apex during diastole. (**c**) Blood flow streamlines during systole at the apex, against the trabeculae

14.2.3 Visualizations

With the fluid velocity fields and level sets generated for each time step, we use Paraview [22] to visualize the simulations. We analyzed a healthy heart and two diseased hearts, and we describe below our visualization methods and our results.

14.2.3.1 Blood Flow Velocity

We performed a visualization of the velocity field within the entire heart, as seen in Fig. 14.14, left and middle columns. The velocity of the blood at a given point is represented by a cone pointed in the direction of the flow. The size of cone increases linearly as the magnitude of the velocity increases. Additionally, we adjust the color of a cone by first setting its hue to 160 (blue), and then linearly lowering this value to a minimum of 0 (red) as velocity increases. We also visualized cross-subsections of the heart to give a clearer picture of how each of the structures and trabeculae affects the blood flow. Screenshots of this visualization can be seen in Fig. 14.14, right column.

Streamline visualizations are shown in Fig. 14.13. The color at a point within a streamline is chosen in the same way as the cones described above. Red streamlines

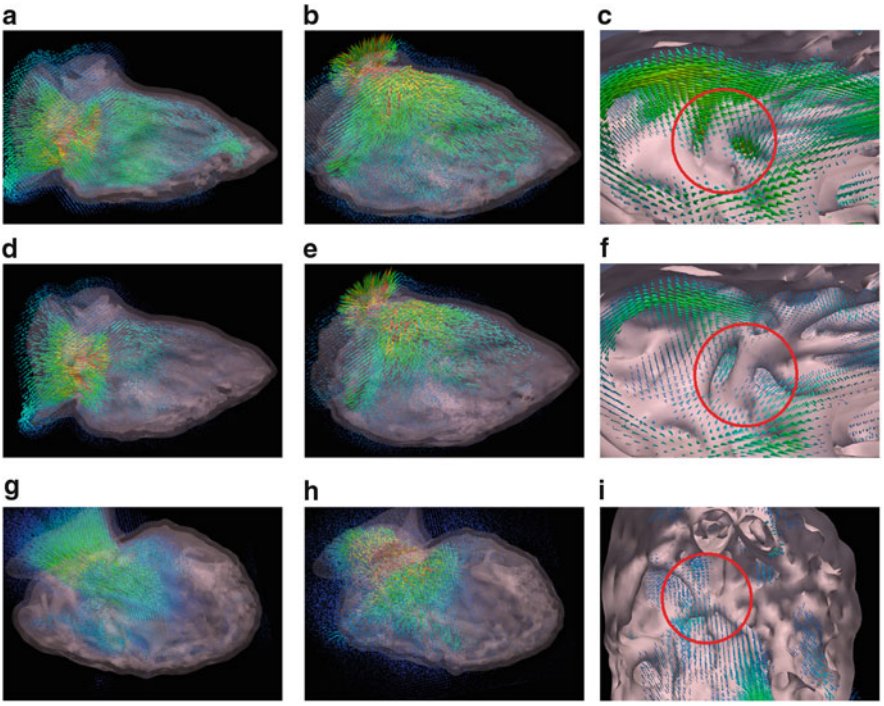

Fig. 14.14 Velocity fields at various time steps for three different hearts. *Top row*: healthy heart, *middle row*: hypokinetic heart, *bottom row*: dyssynchronous heart. *Left column*: diastole, *middle column*: systole, *right column*: velocity field at trabeculae during systole. In the top row, we can clearly see high flow velocities in the trabeculae of the healthy heart. In rows 2 and 3, flow is greatly reduced in these regions

signify fast moving blood, while blue streamlines represent lower speeds. In order to disambiguate direction, we add a small number of arrows within the streamline to point the way it is flowing.

14.2.3.2 Blood Residence Time

In addition to the blood flow velocities, we wish to visualize the residence time of blood within the heart. By doing so, we can quantitatively determine regions of the heart that are at greater risk of thrombus, as slower flows are known to be a significant factor predisposing to thrombus formation.

In order to compute the residence time of blood, we must first determine which regions in the computational domain are interior to the heart. This region changes at every time step, due to the deformation of the heart. We find this interior area by determining which cells are within concave regions of the heart mesh. For each empty (non-solid) cell in the domain at index (i,j,k), we check whether there exists a pair (l_1,l_2) such that both $l_1,l_2 > 0$, and either both cells $(i+l_1,j,k)$ and

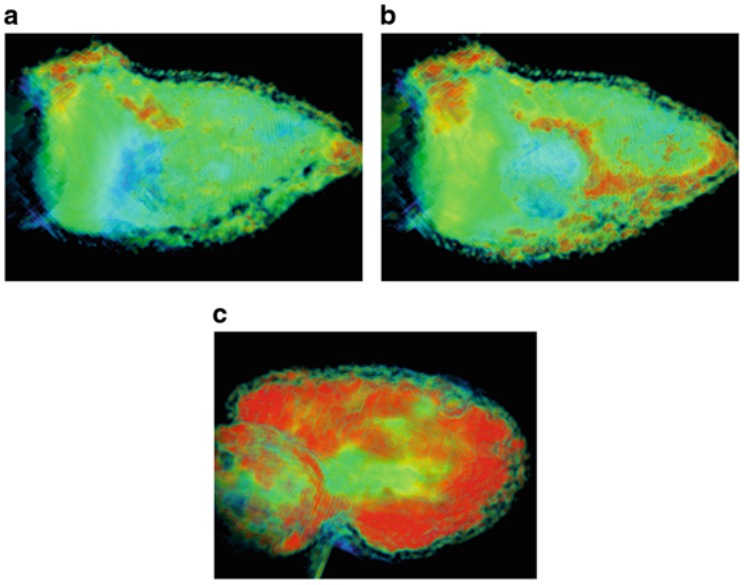

Fig. 14.15 Visualization of average particle residence time. *Dark areas* represent longer average residence time. (**a**) Healthy Heart (**b**) heart with hypokinesis (**c**) heart with dyssynchronous wall movement

$(i-l_2, j, k)$ are solid, cells $(i, j+l_1, k)$ and $(i, j-l_2, k)$ are solid, or cells $(i, j, k+l_1)$ and $(i, j, k-l_2)$ are solid. While this method does not guarantee that all cells within concave regions are determined, our results show that it accurately determines each cell interior to the heart.

At the initial time step, 10,000 particles are generated randomly within the heart. At the beginning of each consecutive time step, new particles are generated within interior cells that are adjacent to exterior cells. Since nearly all such cells are just outside the valves, this allows fresh blood particles to enter the heart during diastole. While some particles are also generated outside the aortic valve, these never enter the heart and are completely removed during systole, and so they do not meaningfully affect the results. Each new particle has an initial age of zero, and this age is incremented at every time step.

At each consecutive time step, we determine a particle's velocity by trilinear interpolation, given the computed fluid velocities at the center of each cell. Each particle's new position is calculated using simple Euler time integration. Then, each particle that now occupies an exterior cell is removed from the system, and the average particle residence time within each cell can then be easily determined. We run this for four cardiac cycles and create volumetric visualization using Paraview, as demonstrated in Fig. 14.15. Here, lighter areas represent blood that has been in the heart for 0–3 cycles, and dark areas represent 3–4 cycles.

14.2.4 Discussion

14.2.4.1 Comparison with Diseased Hearts

The simulation and visualization methods are performed described above on three different hearts. The first is a healthy heart with no visible medical problems. The second is a heart that has simulated hypokinesis, where the motion of the heart walls is decreased at the apex by a maximum of 50 %. The third comes from a patient who has post tetralogy of Fallot repair. This heart is known to suffer from right ventricle hypertrophy, significant dyssynchrony in the basal-midseptum of the left ventricle, and a decreased left ventricle ejection fraction of about 30 %.

The streamline visualizations provide detailed information on the trabeculae–blood interaction. Figure 14.13b, taken during diastole, demonstrates how the complex surface causes the flow to fill the empty spaces between the trabeculae. We can clearly see the development of many small vortices around the trabeculae, which previous methods of cardiac blood flow simulation have not even attempted to capture. Then, in Fig. 14.13c, during systole, we see another example of how the blood is forcefully expelled out of the spaces between the trabeculae, rather than simply flowing directly towards the aortic valve as older methods with simpler meshes have suggested.

Estimated ejection fraction can be calculated using particles to validate our simulation. During systole, we know exactly how many particles there originally existed in the system, and how many are being expelled and deleted at each time step. To estimate the ejection fraction, we simply divide the total number of deleted particles by the original number of particles.

We performed a partial validation by comparing the estimated ejection fraction to the true ejection fraction. The computed ejection fraction is approximately 45 % for the healthy heart, 40 % for the hypokinesis heart, and 30 % for the dyssynchronous heart. These values for the healthy and dyssynchronous heart are in agreement with the true values, so we have confidence in the rest of our results.

Velocity field visualizations are illustrated in Fig. 14.14. We can see that in the healthy heart, the inflow during diastole is significant and fairly uniformly distributed, circulating blood throughout the heart. During systole, the velocity field throughout the heart remains high, and fluid in the apex moves toward the valves. In Fig. 14.14c, we see more detail of the interactions between blood flow and the trabeculae, as the blood is visibly expelled from these regions. However, in the heart suffering from hypokinesis, we find that the velocity field is much weaker toward the apex during both diastole and systole. In Fig. 14.14f, we also see that the trabeculae are no longer adequately expelling blood as they do in the healthy heart case. We also see in Fig. 14.14g–i that the flow patterns in the heart with dyssynchronous heart wall movement appears non-normal, with overall lower velocities and even less fluid being pushed out from the trabeculae.

We then compare the visualizations of the average particle residence times for each of the three simulations, as seen in Fig. 14.15. Each of these images was made at the same time step, at the start of systole, after four cardiac cycles. We find that in Fig. 14.15a, in the healthy heart, nearly the entire domain contains blood with average residence time of less than three cycles, suggesting that the blood is not remaining stagnant, and turning over well between cardiac cycles. In contrast, Fig. 14.15b shows that in the heart suffering from hypokinesis, the average residence time is significantly higher near the walls, particularly near the hypokinetic apex. Finally, in Fig. 14.15c, we find that a very significant region of the blood has a long residence time, suggesting that due to the low ejection fraction and relatively low fluid velocities, blood is not being adequately circulated and thus is remaining stagnant near the walls, again, particularly toward the apex of the heart.

All our results have been validated based on ejection fraction, and based on visual observation by experts. Note that currently there is no method based on MRI to validate our detailed results in such resolution.

14.3 Conclusions

In this chapter, we have proposed a learning scheme for fully automatic and accurate segmentation of cardiac tagged MRI data. First we developed a semiautomatic system to achieve efficient segmentation with minimal user interaction. Then the learning-based framework has three steps. In the first step we learn an ASM shape model as the prior shape constraint. Second, we learn a confidence-rated complex boundary criterion from the local appearance features to use to direct the detected contour to move under the influence of image forces. Third, we also learn a classifier to detect the heart. This learning approach achieves higher accuracy and robustness than other previously available methods. Since our method is entirely based on learning, the way of choosing the training data is critical. We find that if the segmentation method is applied to images at phases or positions that are not represented in the training data, the segmentation process tends to get stuck in local minima. Thus the training data need to be of sufficient size and range to cover all possible variations that may be encountered in practice.

We then described our new framework to generate detailed mesh sequences from CT data and used them to run patient-specific blood flow simulations. We then created several visualizations to reveal the interactions between the complex trabeculae of the heart wall and the blood, which has never been possible before, and used them to compare the flow fields between a healthy heart and two diseased hearts, which would potentially be extremely useful to doctors to help in diagnosis and treatment plans. This is the first time to compare blood flow fields at thus level of resolution.

References

1. Amini AA, Chen Y, Elayyadi M, Radeva P (2001) Tag surface reconstruction and tracking of myocardial beads from SPAMM-MRI with parametric b-spline surfaces. IEEE Trans Med Imaging 20(2):94–103
2. Cootes T, Taylor C, Cooper D, Graham J (1995) Active shape models - their training and application. Comput Vis Image Underst 61(1):38–59
3. Daugman J (1985) Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. J Opt Soc Am A 2(7): 1160–1169
4. de Zélicourt D, Ge L, Wang C, Sotiropoulos F, Gilmanov A, Yoganathan A (2009) Flow simulations in arbitrarily complex cardiovascular anatomies - an unstructured cartesian grid approach. Comput Fluids 38(9):1749–1762
5. Dunn D, Higgins WE, Wakeley J (1994) Texture segmentation using 2-d Gabor elementary functions. IEEE Trans Pattern Anal Mach Intell 16:130–149
6. Foster N, Metaxas D (1996) Realistic animation of liquids. Graph Models Image Process 58:471–483
7. Freund Y, Schapire RE (1995) A decision-theoretic generalization of on-line learning and an application to boosting. In: EuroCOLT '95: proceedings of the second European conference on computational learning theory, pp 23–37
8. Gilmanov A, Sotiropoulos F (2005) A hybrid cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies. J Comput Phys 207(2): 457–492
9. Ginneken BV, Frangi AF, Staal JJ et al (2002) Active shape model segmentation with optimal features. IEEE Trans Med Imaging 21(8):924–933
10. Huang X, Paragios N, Metaxas D (2003) Establishing local correspondences towards compact representations of anatomical structures. In: Proceedings of international conference on medical imaging computing and computer-assisted intervention. Lecture notes in computer science, vol 2879, pp 926–934
11. Huang X, Li Z, Metaxas DN (2004) Learning coupled prior shape and appearance models for segmentation. In: Medical image computing and computer-assisted intervention - MICCAI (1), pp 60–69
12. Huang X, Metaxas D, Chen T (2004) Metamorphs: deformable shape and texture models. In: IEEE conference on computer vision and pattern recognition, vol 1, pp 496–503
13. Jiao F, Li S, Shum H, Schuurmans D (2003) Face alignment using statistical models and wavelet features. In: IEEE conference on CVPR, vol 1, pp 321–327
14. Jones T, Jones TN, Metaxas DN (1998) Patient-specific analysis of left ventricular blood flow. In: Medical image computing and computer-assisted intervention (MICCAI), pp 156–166
15. Kulp S, Metaxes D, Qian Z, Voros S, Axel L, Mihalef V (2011) Patient-specific modeling and visualization of blood flow through the heart. In: IEEE international symposium on biomedical imaging
16. Li S, Zhu L, Jiang T (2004) Active shape model segmentation using local edge structures and Adaboost. In: Medical imaging augmented reality, pp 121–128
17. Long Q, Merrifield R, Yang GZ, Xu XY, Kilner PJ, Firmin DN (2003) The influence of inflow boundary conditions on intra left ventricle flow predictions. J Biomech Eng 125(6):922–927
18. Manglik T, Axel L, Pai W, Kim D, Dugal P, Montillo A, Qian Z (2004) Use of bandpass Gabor filters for enhancing blood-myocardium contrast and filling-in tags in tagged MR images. In: Proceedings of international society for magnetic resonance in medicine, p 1793
19. Mihalef V, Metaxas D, Sussman M (2007) Textured liquids based on the marker level set. Comput Graph Forum 26(3):457–466
20. Mihalef V, Ionasec R, Wang Y, Zheng Y, Georgescu B, Comaniciu D (2010) Patient-specific modeling of left heart anatomy, dynamics and hemodynamics from high resolution 4D CT. In: IEEE international symposium on biomedical imaging, pp 504–507

21. Montillo A, Metaxas D, Axel L (2002) Automated segmentation of the left and right ventricles in 4d cardiac SPAMM images. In: Medical imaging computing and computer-assisted intervention, pp 620–633
22. Paraview - Open Source Scientific Visualization. http://www.paraview.org
23. Saber NR, Wood NB, Gosman AD, Merrifield RD, Yang G-Z, Charrier CL, Gatehouse PD, Firmin DN (2003) Progress towards patient-specific computational flow modeling of the left heart via combination of magnetic resonance imaging with computational fluid dynamics. Ann Biomed Eng 31:42–52
24. Schapire RE (2002) The boosting approach to machine learning: an overview. In: MSRI workshop on nonlinear estimation and classification
25. Schapire RE, Freund Y, Bartlett P, Lee WS (1998) Boosting the margin: a new explanation for the effectiveness of voting methods. Ann Stat 26(5):1651–1686
26. Sederberg TW, Parry SR (1986) Free-form deformation of solid geometric models. In: Proceedings of the 13th annual conference on computer graphics, pp 151–160
27. Shen D, Davatzikos C (2000) Adaptive-focus statistical shape model for segmentation of 3D MR structures. In: Medical image computing and computer assisted intervention, pp 206–215
28. Shen T, Li H, Qian Z, Huang X (2009) Active volume models for 3d medical image segmentation. In: IEEE conference on computer vision and pattern recognition, pp 707–714
29. Sussman M (2005) A parallelized, adaptive algorithm for multiphase flows in general geometries. Comput Struct 83(6–7):435–444
30. Viola P, Jones M (2001) Robust real-time object detection. In: Second international workshop on statistical and computational theories of vision - modeling, learning, and sampling, Vancouver, 13 July 2001
31. Yokoi K, Xiao F, Liu H, Fukasaku K (2005) Three-dimensional numerical simulation of flows with complex geometries in a regular cartesian grid and its application to blood flow in cerebral artery with multiple aneurysms. J Comput Phys 202(1):1–19
32. Zhu S, Lee T, Yuille A (1995) Region competition: unifying snakes, region growing, energy/Bayes/MDL for multi-band image segmentation. In: International conference on computer vision, June 1995, pp 416–423