Dynamic Surface Reconstruction from 4D-MR Images

Matthias Fenchel[1], Stefan Gumhold[2], Hans-Peter Seidel[3]

[1]Siemens AG Medical Solutions, Magnetic Resonance, Karl-Schall-Str. 4, 91052 Erlangen
[2]TU Dresden, SMT CGV, 01062 Dresden
[3]Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken
Email: [1]matthias.fenchel@siemens.com
[2]sg30@mail.inf.tu-dresden.de
[3]hpseidel@mpi-sb.mpg.de

Abstract

In this work we present a new analysis technique for dynamic 3D images. The method allows the reconstruction of dynamic surfaces describing object boundaries in the 3D images over time. The only input of the user is a grey value threshold used for boundary detection and a bounding box around the surfaces of the object of interest. The output of our method is a dynamic surface mesh that tracks the object boundary over time.

We first extract a set of weighted points from the MR image data using a local boundary voting scheme. The dynamic surface is defined as an extremal surface on the resulting point-sampled boundary probability density. In this way a surface with adjustable smoothness is obtained that can be reconstructed using projection operators similar to the ones used for point set surfaces.

We extend the projection operators to 3-manifolds in 4D and obtain a much better tracking performance of the dynamic surface model. In contrast to computationally expensive active contour algorithms, the new algorithm can be easily parallelized. The proposed approach is applied to 4D-MR images of a human heart in motion.

1 Introduction

Image segmentation is an important task in many medical applications. Almost all computer-based medical methods include a segmentation step to infer relevant information from image data. A variety of segmentation approaches have been proposed including model-based methods, like active contours, and pixel-based ones, e.g. watershed transformations or morphological segmentation.

We follow a probabilistic approach, where the probability density of object boundaries is derived from a local voting scheme. It is sampled on a set of points with the weights measuring the boundary probability. The object boundaries are defined as the surfaces of probability maxima in direction of the gradient of the probability density. The problem of dynamic boundary segmentation therefore boils down to a space-time meshing problem. We solve this by an implicit meshing of the first time frame and a tracking procedure that is guided by the 3-manifold of gradient maxima residing in 4D space-time.

One important advantage is that the number of parameters that have to be adjusted is reduced to one intuitive intensity threshold and the surface component of interest. The proposed approach is applied to 4D space-time MR images of the human heart in motion.

The paper is structured as follows. First we introduce the estimation of the object boundary probabilities in Section 2. The successive section describes the definition of the dynamic surface of maximal probability and introduces projection operators for almost orthogonal projection onto the dynamic surface. Section 4 describes the incremental meshing approach for the dynamic surface. The application to 4D cardiac MRI is presented in section 5. Related work is stated at the beginning of each section.

2 Boundary Probability Density

In this work we aim at a tool to reconstruct surfaces from cardiac MR images. Measurement parameters of the given images make blood voxels appear at a higher intensity than tissue. The inner walls of the cardiac chambers are therefore determined by big discontinuities in image intensity values from high to low.

A variety of algorithms have been proposed for analyzing image intensity variation, including statistical methods [34, 11, 27, 12] and [16], differential methods [30, 18] and [24] and curve fitting methods [14, 17, 13, 28] and [29]. Edge detection in noisy environments can be treated as an optimal linear filter design problem [35, 6, 31, 23] and [32]. Canny [6] formulated edge detection as an optimization problem and defined an optimal filter, which can be efficiently approximated by the first derivative of a Gaussian function in the one-dimensional case. Canny's filter can be implemented recursively [8], which provides a more efficient way for image noise filtering and edge detection. Parametric models, that are often found for edge detection, restrict the type of step edge geometries considered and cannot cope easily with close by features.

There exist a lot of close by features in the human heart and the images can be locally poor in contrast but contain significant noise. Moreover, MR images tend to have biases[1]. For the reasons mentioned, high-end edge detectors are necessary for our application such as Canny's edge detector. However, Canny has many parameters that must be adjusted to obtain an optimal result, which cannot be easily done interactively for 3D or 4D image data. Furthermore, generalization to higher dimensions is complicated. Therefore, a different approach was chosen to infer boundary information from the given images, based on the following demands: only few input parameters should be required and adaption to local contrast should be possible. Furthermore, detected boundaries should be narrow and generalization to higher boundaries straightforward.

2.1 2-Means Cluster Edge Detection in 2D

In this section we describe a boundary detector, which fulfills the demands stated in the introduction to this section, for edge detection in 2D. This allows a comparison to the Canny edge detector as shown in Figure 4.

The boundary detector analyzes the local neighborhood of each image pixel, searching for step edges within the neighborhood. The pixels of the neighborhood are clustered into two sets according to their intensity values. This is done using a standard k-means clustering approach based on the intensity values of the pixels. Let $\mathcal{N}(p)$ denote the local neighborhood around pixel p and $I(p)$ its intensity value. The 2-means clustering procedure partitions the pixels from $\mathcal{N}(p)$ into two clusters C_1 and C_2 by assigning a representative intensity value $I_{1/2}$ to each cluster. The representatives are updated incrementally in a two step procedure. First all pixels $p_j \in \mathcal{N}(p)$ are assigned to the cluster $C_i = \arg_i \min |I_i - I(p)|$ whose intensity value is closer to the intensity value of the pixel. Then each representative value is updated to the mean intensity value $I_i = |C_i|^{-1} \sum_{p \in C_i} I(p)$ of the pixels assigned to its cluster. This Lloyd iteration is known to converge to a global optimum, where the squared distances of the pixel intensity values to their cluster representatives are minimized. In the simple 2-means clustering the Lloyd iteration converged after an average of three to four iterations. The k-means clustering of the pixels results in a stable classification of the local neighborhood into interior (high intensity) and exterior (low intensity) components.

In Figure 4 a typical local clustering result is illustrated with the interior cluster pixels with green frames and the exterior ones with red frames. One can easily see that the boundary shape can be quite complicated even in the local neighborhood without doing harm to the classification of each pixel into inside and outside. The clustering is stable, which means that pixel classifications do not change if the neighborhood mask is moved on to the surrounding pixels. The cluster boundary in the local neighborhood is not located on the pixels but on the edges between pixels. The boundary probability of the edges is accumulated by a local voting scheme that sweeps the neighborhood mask once over the image and counts for each edge between two pixels the number of transitions from one cluster to another. To avoid classifying boundary edges falsely positive, we introduce a significance test (the number of pixels in each of the clusters must be larger than ten percent of the number of pixels in the local neighborhood) and a step-height test (the difference between the cluster representative intensity values must be larger than a user defined threshold τ) that a clustering has to pass before we vote its pixel boundary edges.

The final output of the boundary detector is a set of weighted edges, which we transform to a set of weighted points that can be used as input to the surface definition as described in Section 3. The points simply inherit the weights of the edges and are located on the edge centers, which compose a staggered grid on the image. To compare the quality of the boundary detector to the Canny edge detector, we combine the staggered grid by adding for each pixel the square root of the two weights of its top and right edge. The square root can be motivated by considering the fact that diagonal edges would receive too many counts because of the rasteriza-

[1] regions of equal tissues can have different intensities

tion and thus obtain higher boundary strengths than straight edges. The detected boundary is blurred by this staggering- and combining-procedure a little bit and shifts half a pixel to the lower left as can be seen in Figure 4 b). A comparison to the Canny edge detector is shown in c). Adapting the standard derivative, the Gaussian smoothing and the two threshold values of the Canny filter to obtain a good result was a difficult task. Our algorithm, in contrast, only takes the intuitive step-height threshold τ. Despite the blurring, the boundaries detected by the algorithm are reasonably good, i.e. they are comparable to Canny's edge detection, if not better. This contributes to the fact that our boundary detector is much easier to handle. The generalization of our boundary detection approach to higher dimensions is straightforward. The dimension of the local neighborhood is increased. This results in a higher computational complexity but the 2-means clustering procedure remains the same. The voting is finally done on the (hyper-)faces of the (hyper-) voxels.

3 Defining Dynamic Surfaces from Weighted Points

Recent work on the approximation of point sampled surfaces provides powerful tools for the definition of approximating smooth surfaces. Levin et al. [21, 3, 22] defined a smooth surface that approximates a set of scattered data points as the fixed points of a projection operator. Inspired by Levin's work, Adamson and Alexa [1, 2] defined smooth approximating surfaces from an implicit definition. Shen et al. [33] defined implicits that allow to approximate or interpolate point clouds and polygon soups. Amenta and Kil [4] pointed out the relation to extremal surfaces, which were previously used by Medioni et al. [26] to reconstruct surfaces from very noisy point and normal data.

In this work, we closely follow the implicit surface definition by Adamson and Alexa [2]. We apply it to the output of the boundary detector from Section 2 in order to compute a surface of local probability maxima in the gradient direction, where the boundary detector samples the probability density on a set of weighted points. In Section 3.2 the framework is generalized to dynamic point clouds sampled on several time slices. To facilitate the extraction of a 3D mesh representing a time slice of the dynamic surface, we propose a new version of the almost orthogonal projection operator in Section 3.3, which works for arbitrary subspaces of $\mathbf{R}^4$. This operator allows to project points onto the dynamic surface subject to linear constraints. We extend this projection operator to a more robust version, using damping factors.

3.1 Static Surfaces for Weighted Points

The boundary detector samples the boundary probability density on a set of weighted 3D points $P = \{(\mathbf{p}_i, \omega_i)\}_i, \mathbf{p}_i \in \mathbf{R}^3, \omega_i \in \mathbf{R}$. In order to define the surface of probability extrema, a gradient direction and a test for an extremum is necessary. Both ingredients can be derived from a weighted least squares fitting plane similar to Adamson and Alexa [1] and Amenta and Kil [4]: For each point $\mathbf{x}$ in space we define a weighted least-squares fitting plane $H(\mathbf{x})$ represented by a normal vector $\mathbf{n}(\mathbf{x})$ and its signed distance $d(\mathbf{x})$ from the origin. We call $\mathbf{x}$ the reference point because the input points $\mathbf{p}_i$ are weighted by their distances $r_i = \|\mathbf{p}_i - \mathbf{x}\|$ to $\mathbf{x}$. The weighted least-squares plane is abbreviated by *WLS plane*. A positive, monotonically decreasing weighting function $\theta(r)$ is used to map the distances from the reference point to weights. A secondary weighting is done by the point weights ω_i. The WLS plane $H(\mathbf{x}) = (\mathbf{n}(\mathbf{x}), d(\mathbf{x}))$ is chosen to minimize the weighted least-squares energy function $e(\mathbf{x}, \mathbf{n}, d)$

$$e(\mathbf{x}, \mathbf{n}, d) \overset{\text{def}}{=} \frac{\sum_i \left(\mathbf{n}^t \mathbf{p}_i - d\right)^2 \omega_i \theta(r_i)}{\sum_i \omega_i \theta(r_i)} \tag{1}$$

in the arguments $\mathbf{n}$ and d with $r_i \overset{\text{def}}{=} \|\mathbf{p}_i - \mathbf{x}\|$. Formally, the WLS plane $H(\mathbf{x})$ can be defined as

$$H(\mathbf{x}) = (\mathbf{n}(\mathbf{x}), d(\mathbf{x})) \overset{\text{def}}{=} \min \arg_{\mathbf{n},d} e(\mathbf{x}, \mathbf{n}, d). \tag{2}$$

Algorithmically, the WLS plane $H(\mathbf{x})$ of each reference point $\mathbf{x}$ can be computed by the standard least-squares fitting procedure from the normal equations. Its normal can be considered as a robust estimation of the gradient of the probability density. Note that, in general, the reference point is not located on its WLS plane itself, if the reference point is not a maximum of the probability density in direction of the WLS normal. The surface $\mathcal{S}$ of maxima in gradient direction is then defined as the reference points in space that are contained in their WLS planes:

$$\mathcal{S} \overset{\text{def}}{=} \{\mathbf{x} | \mathbf{x} \in H(\mathbf{x})\}. \tag{3}$$

This definition is equivalent to the definition by Adamson and Alexa [1] but avoids the direct use of the weighted average point $\mathbf{a}(\mathbf{x})$. It is replaced by the distance $d(\mathbf{x}) = \mathbf{n}^t\mathbf{a}(\mathbf{x})$ of the WLS plane to the origin. The surface can therefore be interpreted as an implicit surface:

$$\mathcal{S} = \left\{\mathbf{x}|\mathbf{n}(\mathbf{x})^t\mathbf{x} - d(\mathbf{x}) = 0\right\} \tag{4}$$

It is a smooth manifold in the region of space, in which the WLS plane is uniquely defined. This was shown by Adamson and Alexa in [1] for the case without point weights ω_i. The proof can be easily generalized to the weighted case by interpreting a point with weight $\omega_i \approx n_i/N$ as n_i points at the same location, where N is a natural number that tends to infinity.

3.2 Dynamic Surfaces from Weighted Points

In the dynamic case, the input is a set $\bar{P}$ of weighted points $(\bar{\mathbf{p}}_i = (\mathbf{p}_i, t_i), \omega_i)$ in 4D space-time. Vectors and operators in space-time are marked by the bar over the symbol. All the definitions of the previous section can be generalized easily. For each reference point $\bar{\mathbf{x}}$ in space-time we define a WLS hyperplane $\bar{H}(\bar{\mathbf{x}})$ from the distance weighted WLS energy. The dynamic surface $\bar{\mathcal{S}}$ is then the 3-manifold in space-time consisting of all reference points contained in their WLS hyperplanes:

$$\bar{\mathcal{S}} \overset{\text{def}}{=} \left\{\bar{\mathbf{x}}|\bar{\mathbf{x}} \in \bar{H}(\bar{\mathbf{x}})\right\}. \tag{5}$$

3.3 Projection onto lower Dimensional Sub-Spaces

We did not extend the weighted least-squares approach to the fitting of higher order polynomials as proposed by Levin [22] because the given surface definitions 3 and 5 allow further projection operators. A projection operator Π projects a point $\mathbf{p}$, located close to the [dynamic] surface, onto the surface. In Section 4, we need an operator that projects onto the dynamic surface at a given time $t = t_i$ for the extraction of dynamic meshes, which change in time. One cannot simply use the space-time version $\bar{\Pi}_{\bar{\perp}}$ of the projection operator, because it does not necessarily project onto the required hyperplane, defined by $t = t_0$.

Instead we define a new projection operator $\bar{\Pi}_{\bar{\perp}}|_{\mathcal{R}}$, which operates on a linear sub-space $\mathcal{R}$ of $\mathbf{R}^4$. This sub-space can be defined by any set of linear constraints. One can therefore restrict the operator to a 3D-plane in space-time, a 2D-plane in space or space-time and a space-time line. The restricted projection operator $\bar{\Pi}_{\bar{\perp}}|_{\mathcal{R}}$ takes a point $\bar{\mathbf{p}}$ from the sub-space $\mathcal{R}$ and projects it to that point $\bar{\mathbf{x}}$ on $\bar{\mathcal{S}}$ within $\mathcal{R}$. The projection is almost orthogonal within $\mathcal{R}$. In Section 4 we will use the time slices $t = t_i$ as a linear constraint.

The only change in the restricted projection procedure is that we project the reference point $\bar{\mathbf{x}}$ not to the WLS hyperplane $\bar{H}(\bar{\mathbf{x}})$ but to the subset of the hyperplane $\bar{H}(\bar{\mathbf{x}})$ defined by the lower dimensional space $\mathcal{R}$, i.e. we project $\bar{\mathbf{p}}$ onto $\bar{H}(\bar{\mathbf{x}}) \cap \mathcal{R}$. Together with a damping factor β, that gradually slows down the projection, resulting in more stability, the restricted projection is computed using the following iteration:

```
**procedure** $\bar{\Pi}_{\bar{\perp}}|_{\mathcal{R}}(\bar{\mathbf{p}})$
    $\bar{\mathbf{x}}_0 \overset{\text{def}}{=} \bar{\mathbf{p}} \in \mathcal{R}$
    **repeat**
        $\bar{\mathbf{x}}_{i+1} \overset{\text{def}}{=} \beta\Pi_{\bar{H}(\bar{\mathbf{x}}_i)\cap\mathcal{R}}(\bar{\mathbf{p}}) + (1-\beta)\bar{\mathbf{x}}_i$
    **until convergence**
```

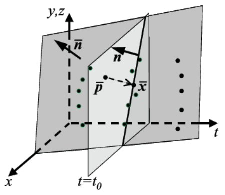

Figure 1: Illustration of the projection operator $\bar{\Pi}_{\bar{\perp}}|_{t=t_0}$ restricted to the time slice $t = t_0$.

Figure 1 illustrates the restricted projection operator for the case that $\mathcal{R}$ is the time slice $t = t_0$ shown in light grey. The WLS hyperplane is fitted in 4D space-time to the weighted points. It has the space-time normal $\bar{\mathbf{n}}$. The restriction of the WLS hyperplane to the time slice is the bold line, which represents a 2D-plane in the time slice $t = t_0$ with normal $\mathbf{n}$. The point $\bar{\mathbf{p}}$ is projected along $\mathbf{n}$ onto this 2D-plane resulting in the next reference point $\bar{\mathbf{x}}$. As $\bar{\mathbf{p}}$ has to be in the restriction space $\mathcal{R}$, this projection makes sense. In the next iteration the hyperplane is fitted with distance weights relative to the new reference point $\bar{\mathbf{x}}$. Again, it is restricted to the time slice and $\bar{\mathbf{p}}$ is projected to the resulting 2D-plane.

If the iterative projection procedure converges, it is clear that the resulting point $\bar{\mathbf{x}}$ is within $\mathcal{R}$. What remains to be shown is that $\bar{\mathbf{x}}$ is also on the dynamic surface $\bar{\mathcal{S}}$. Let us assume that the procedure converged after i iterations. For any $\beta > 0$ the convergence criterion tells us that $\bar{\mathbf{x}}_i$ must have projected onto itself and therefore be located on the restriction of its own WLS plane $\bar{H}(\bar{\mathbf{x}}_i) \cap \mathcal{R}$. This implies that $\bar{\mathbf{x}}_i$ is also on the hyperplane $\bar{H}(\bar{\mathbf{x}}_i)$ itself, which is exactly definition 5 of a point on $\bar{\mathcal{S}}$, q.e.d.

4 Dynamic Meshing

Hoppe et al. [15] introduced the atomic connectivity operations used by most dynamic meshing approaches: edge-collapse, edge-split and edge-flip. They used it in the context of mesh optimization. Lachaud and Montanvert [20] extended Hoppe's basic operations by operations that allow to merge and split surface components and therefore allow to change the topology. They also propose the use of $\sqrt{3}$-subdivision with successive edge-flips to increase mesh resolution globally. McInerney and Terzopoulos [25] propose a dynamic surface mesh called T-Snakes, whose connectivity is adapted based on a surrounding regular grid. T-Snakes also allow changes in topology.

Kobbelt et al. [19] make use of three operations proposed by Hoppe et al. and come up with a multi-resolution framework for meshes with changing connectivity but identical topology. First, all edges are adjusted to lengths within an interval $[l_{\min}, l_{\max}]$ by application of edge-collapse and edge-split operations. In a second update stage, edges are flipped in order to bring the vertex valences as close to six as possible, i.e. they perform an edge-flip whenever the sum of the squared difference of the vertex valences to ordinary valence six can be reduced.

Zhukov et al. [36] use the same update criteria for their deformable model. Each of these approaches does not allow changes of the topology. Cheng et al. [7] base dynamic meshing on the skin surface by Edelsbrunner [9]. The skin is defined by a finite set of control spheres as the envelope of an infinite number of convex combinations of the spheres. Edge-collapse and edge-split operations are used to adjust the vertex density, which is given by a sampling of the skin according to the local curvature with a maximal meshing error ϵ.

4.1 Meshing the First Frame

There are different possibilities to create a mesh for the first frame. We chose to use a marching cubes approach, making use of the implicit surface definition given in equation 4. The advantage of the marching cubes algorithm is its ability to handle surfaces of arbitrary topology. To improve the mesh quality we extracted the first frame in a high resolution and simplified it using the edge-collapse based mesh simplification approach proposed by Garland and Heckbert [10].

As input to the marching cubes algorithm we used a signed distance function derived from equation 4. The distance $g(\mathbf{x})$ can directly be defined up to its sign as

$$g(\mathbf{x}) \overset{\text{def}}{=} \pm \left| \mathbf{n}(\mathbf{x})^t \mathbf{x} - d(\mathbf{x}) \right| . \qquad (6)$$

The sign cannot be found by means of the WLS-fitting procedure as both $\mathbf{n}$ and $-\mathbf{n}$ are solutions to the normal equations.

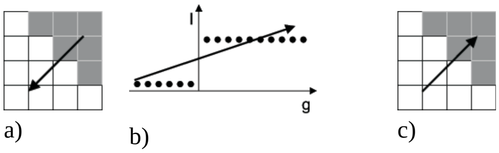

Figure 2: Calculation of the consistent normal orientation. a) shows the neighborhood and the preliminary normal. b) shows the transformation on an axis along the normal direction. The least squares fit is shown as an arrow. c) Positive slope of the least squares fit results in a switch of the normal direction.

Consistent orientation of the normals can be achieved in our case by taking into account that in our MR images the inside of the heart chambers are regions of higher image intensities. Therefore, the correct normal must be oriented from the region of high intensity to the region of low intensity as shown in Figure 2 c). Again we use a voting scheme for a robust determination of the correct sign of the distance function g evaluated at the reference point $\mathbf{x}$. First (Figure 2 a) we assume the normal $\mathbf{n}$ from the plane fitting procedure to be correct. Then we compute the signed distances for all voxel locations in the neighborhood of $\mathbf{x}$ and construct a 2D diagram of the intensity values of the voxels over their signed distance as shown in Figure 2 b). If the normal was oriented correctly, the diagram should show a decreasing function, which can be checked with the slope of a least-squares fitted linear function as illustrated by the arrow in Figure 2 b). Here the slope is positive and therefore the normal must be oriented in opposite direction as shown in c).

The marching cubes algorithm is performed on a region of interest, which is selected by a bounding box provided by the user. The extracted mesh is first broken down into its connected components. Usually, the component of interest is the one, consisting of the largest number of triangles.

4.2 Time Evolution of Vertex Locations

The positions of the mesh vertices must be located on the 3-manifold in space-time. Given a mesh m_{t_i} at time $t = t_i$, to which positions $\bar{\mathbf{p}}_v$ should the vertices be moved at time $t = t_{i+1}$? A simple approach would be to just transform the 3D position $\bar{\mathbf{p}}_v = \{\mathbf{p}_v, t_i\}$ of a vertex v to $\bar{\mathbf{p}}_v = \{\mathbf{p}_v, t_{i+1}\}$ and to perform a projection as defined in 3.3 starting at the space time point $\bar{\mathbf{p}}_v = \{\mathbf{p}_v, t_{i+1}\}$ and converging in a vertex position $\bar{\mathbf{p}}'_v = \{\mathbf{p}'_v, t_{i+1}\}$

However, this approach does not make use of the dynamic surface information that is available. A better idea is to use the space-time information at $\bar{\mathbf{p}}_v = \{\mathbf{p}_v, t_i\}$ to make a prediction for the probable position of a vertex at time $t = t_{i+1}$ and to perform a projection afterwards that is likely to converge the faster the better the prediction.

Our approach for making a prediction is visualized in Figure 3 for a vertex point $\bar{\mathbf{p}}_v = \{\mathbf{p}_v, t_i\}$ from the dynamic surface at $t = t_i$. The hypersurface $S(t = t_i)$ at $t = t_i$ is shown as a dark curved line. We assume that motion of a vertex is linear with respect to the time domain. The idea is to calculate the WLS hyperplane $\bar{H}$ framed in grey at $\bar{\mathbf{p}}_v = \{\mathbf{p}_v, t_i\}$ and make use of the fact that it is close to the tangent hyperplane of the dynamic surface as $\bar{\mathbf{p}}_v = \{\mathbf{p}_v, t_i\}$ is a point on the surface. If we project the unit vector in time direction onto this tangent hyperplane, we obtain a vector $\bar{\mathbf{d}}$, as visualized in the figure. Following the vector $\bar{\mathbf{d}}$ to the intersection with the hyperplane at $t = t_{i+1}$ yields a prediction for the next position of the vertex. The following projection then yields the actual position $\bar{\mathbf{p}}'_v = \{\mathbf{p}'_v, t_{i+1}\}$ on the 3-manifold. Figure 3 visualizes the steps of the prediction for one vertex.

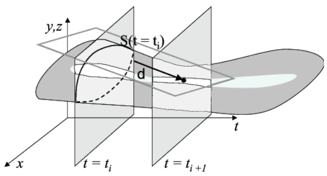

Figure 3: Space-time view of the prediction operation. The three manifold is shown in dark grey and the hypersurface $S(t = t_i)$ at $t = t_0$ as the dark curved line. The tangent plane is framed in light grey, while the prediction vector $\bar{\mathbf{d}}$ is painted bold.

4.3 Time Evolution of Mesh Connectivity

In this work we follow a similar approach to Zhukov et al. [36]. After providing an initial mesh, we maintain mesh quality in the subsequent frames by applying the following operations if necessary:

1. If the length of an edge drops below some threshold value l_{min}, it is collapsed, if the mesh connectivity allows a collapse.
2. If the length of an edge exceeds some threshold value l_{max}, it is split by midpoint insertion.
3. while there are vertices, whose valence differs from an optimum of 6, edge flips are performed, until the sum $\sum_i (valence(v_i) - 6)^2$ has reached a minimum.
4. After applying all dynamic meshing operations, all vertices are projected back to the surface.

This approach allows maintaining mesh quality within reasonable limits. The constants l_{min} and l_{max} are initialized by using statistical values of the average edge lengths of the initial mesh, i.e. $l_{min} = c_{min} \cdot l_{average}$ and $l_{max} = c_{max} \cdot l_{average}$ with the constants $c_{min} = 0.25$ and $c_{max} = 1.5$.

During the time propagation process, as vertices are moved individually, triangles can flip changing their normal orientations, which results in invalid twisted mesh connectivities. These triangle flips can be avoided by either using smaller temporal sampling steps, thus preventing triangle flips from happening at all, or by detecting flipped triangles and adapting connectivity consistently. We have chosen the first alternative. The second alternative is left out for future work.

5 Application to 4D-MRI

5.1 Data Acquisition and Pre-Processing

25 frames of 25 slices were taken of the cardiac cycle of a 26-year old male proband.[2] Each slice had an in-plane resolution of 156x192 pixels with pixel spacings of 1.67 mm and a slice thickness of 5 mm. The slices were acquired without gaps perpendicular to the central long axis of the heart. The data were prepared by assembling the slices according to their spatial and temporal locations to a regular 4D image.

[2]The cardiac data were provided by courtesy of the Department of Radiology of the University Hospital of Tübingen.

5.2 Slice Enrichment

Since resolution in direction of the slice normal is less than half of the resolution in both other directions, an interpolation step, that introduces additional slices, was necessary. While usual techniques of linear, sinc or spline interpolation did not yield satisfying results, an optical flow based interpolation scheme was chosen. In this approach, the optical flow between two neighbor slices is calculated according to Bergen et al. [5], resulting in two dense disparity maps with a vector for each pixel, pointing to the pixel with the most probable correspondence in the neighbor slice. The interpolation method is visualized in Figure 5. Interpolation using optical flow is done by introducing a new image of identical size between the two slices. We then warp the image according to the disparity vectors given by the optical flow field and rescale the image. This method produces good results without over- or undersampling artifacts.

The slice enrichment process yields approximately isotropic voxels. Subpixel interpolation on the isotropic grid, which is necessary for the surface reconstruction process, is done, for performance reasons, by using quadrilinear interpolation.

6 Conclusions and Future Work

We presented an algorithm capable of reconstructing smooth dynamic surfaces from 4D image input data. We reconstructed the surface of the left ventricle and atrium of a human heart requiring only little human interaction. The results are shown in Figure 6. Although the 4D-MR image data used as input are not of optimal quality, the method has proven stable. In clinical routine, however, the MR image input is expected to be of even worse quality. Future work should therefore be directed towards further improving the boundary detection. Moreover, extraction of the first frame could be done by a method that allows more flexibility like an adaptive step size, which reduces or increases the sampling rate according to the local curvature. Thus, meshes of better quality would be obtained for the first frame, making the preprocessing step of the connectivity unnecessary. The remeshing step should be improved, introducing predefined quantitative surface error criteria for the collapse and splitting process. Future work should also consider connectivity adapting schemes that allow efficient resolution of inverse triangles that have been flipped during the reconstruction process, as mentioned in Section 4.3. Furthermore, topology changes could be supported during the reconstruction process by searching for critical points. Last but not least, edge flips could be replaced by a sequence of insertions, vertex translations and collapses, avoiding popping effects and making the animation really smooth.

References

[1] A. Adamson and M. Alexa. Approximating and intersecting surfaces from points. In *Proceedings of Eurographics Symposium on Geometry Processing*, pages 230–239, 2003.

[2] A. Adamson and M. Alexa. On normals and projection operators for surfaces defined by point sets. In *Proceedings of Eurographics Symposium on Point-based Graphics*, pages 149–156, 2004.

[3] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. *IEEE Visualization 2001*, pages 21–28, October 2001. ISBN 0-7803-7200-x.

[4] N. Amenta and Y. J. Kil. Defining point set surfaces. *ACM Transactions on Graphics (TOG)*, 23(3):264–270, 8 2004.

[5] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *ECCV92*, pages 237–252, 1992.

[6] J. Canny, 1986. J. Canny, A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. PAMI-8 (1986), pp. 679–698.

[7] H.-L. Cheng, T. K. Dey, H. Edelsbrunner Edelsbrunner, and John Sullivan. Dynamic skin triangulation. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-01)*, pages 47–56, New York, January 7–9 2001. ACM Press.

[8] R. Deriche, 1987. R. Deriche, Optimal edge detection using recursive filtering, Proceedings of the First International Conference on Computer Vision, London, 1987.

[9] H. Edelsbrunner. Deformable smooth surface design. *Discrete and Computational Geometry*, 21(1):87–115, January 1999.

[10] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. *Computer Graphics*, 31(Annual Conference Series):209–216, 1997.

[11] J. Haberstroh, 1993. J. Haberstroh and L. Kurz, Line detection in noisy and structured background using Graco-Latin squares. CVGIP: Graphical Models Image Process. 55 (1993), pp. 161–179.

[12] F.R. Hansen, 1982. F.R. Hansen and H. Elliot, Image segmentation using simple Markov field models. Comput. Graphics Image Process. 20 (1982), pp. 101–132.

[13] R.M. Haralick, 1981. R.M. Haralick and L. Watson, A facet model for image data. Comput. Graphics Image Process. 15 (1981), pp. 113–129.

[14] R.M. Haralick, 1984. R.M. Haralick, Digital step edges from zero crossing second directional derivatives. IEEE Trans. Pattern Anal. Mach. Intell. PAMI-6 (1984), pp. 58–68.

[15] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. *Computer Graphics*, 27(Annual Conference Series):19–26, 1993.

[16] J.S. Huang, 1988. J.S. Huang and D.H. Tseng, Statistical theory of edge detection. Comput. Vision Graphics Image Process. 43 (1988), pp. 337–346.

[17] M.H. Huechel, 1971. M.H. Huechel, An operator which locates edges in digitized pictures. J. Assoc. Comput. Mach. 18 (1971), pp. 113–125.

[18] R. Kirsh, 1971. R. Kirsh, Computer determination of the constituent structure of biological images. Comput. Biomed. Res. 4 (1971), pp. 314–328.

[19] Leif P. Kobbelt, Thilo Bareuther, and Seidel Seidel. Multiresolution shape deformations for meshes with dynamic vertex connectivity. *Computer Graphics Forum*, 19(3):249–260, August 2000. ISSN 1067-7055.

[20] J.-O. Lachaud and A. Montanvert. Deformable meshes with automated topology changes for coarse-to-fine 3D surface extraction. *Medical Image Analysis*, 3(2):187–207, 1999.

[21] D. Levin. The approximation power of moving least-squares. *Mathematics of Computation*, 67(224):1517–1531, 1998.

[22] D. Levin. *Geometric Modeling for Scientific Visualization*, chapter Mesh-Independent Surface Interpolation, pages 37–49. Springer-Verlag, 2003.

[23] B.S. Manjunath, 1993. B.S. Manjunath and R. Chellappa, A unified approach to boundary perception: edges, textures and illusory contours. IEEE Trans. Neural Networks 4 (1993), pp. 96–108.

[24] D. Marr, 1984. D. Marr and E. Hidreth, Theory of edge detection. Proc. Roy. Soc. London PAMI-6 (1984), p. 58.

[25] T. McInerney and D. Terzopoulos. T-snakes: Topology adaptive snakes. *Medical Image Analysis*, 4(2):73–91, 2000.

[26] G. Medioni, M.-S. Lee, and C.-K. Tang. *A Computational Framework for Segmentation and Grouping*. Elsevier, 2000.

[27] N.E. Nahi, 1972. N.E. Nahi and T. Assefi, Bayesian recursive image estimation. IEEE Trans. Comput. 7 (1972), pp. 734–738.

[28] V.S. Nalwa, 1984. V.S. Nalwa and T.O. Binford, On detecting edges. IEEE Trans. Pattern Anal. Mach. Intell. PAMI-6 (1984), pp. 58–68.

[29] V.S. Nalwa, 1986. V.S. Nalwa and T.O. Binford, On detecting edges. IEEE Trans. Pattern Anal. Mach. Intell. PAMI-8 (1986), pp. 699–714.

[30] J.M.S. Prewitt, 1970. J.M.S. Prewitt, Object enhancement and extraction. In: B.S. Lipkin and A. Rosenfeld, Editors, Picture Processing and Psychopictorics, Academic Press, New York (1970).

[31] T.D. Sanger, 1989. T.D. Sanger, Optimal unsupervised learning in a single layer feedforward neural network. Neural Networks 2 (1989), pp. 459–473.

[32] S. Sarkar, 1991. S. Sarkar and K.L. Boyer, On optimal infinite impulse response edge detection filters. IEEE Trans. Pattern Anal. Mach. Intell. PAMI-13 (1991), pp. 1154–1171.

[33] C. Shen, J. F. O'Brien, and J. R. Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. In *Proceedings of ACM SIGGRAPH 2004*. ACM Press, August 2004.

[34] D. Stern, 1988. D. Stern and L. Kurz, Edge detection in correlated noise using Latin squares models. Pattern Recognition 21 (1988), pp. 119–129.

[35] V. Torre, 1986. V. Torre and T. Poggio, On edge detection. IEEE Trans. Pattern Anal. Mach. Intell. PAMI-2 (1986), pp. 147–163.

[36] L. Zhukov, Z. Bao, I. Guskov, J. Wood, and D. Breen. Dynamic deformable models for 3d mri heart segmentation. In *Proceedings of SPIE Medical Imaging*, 2002.

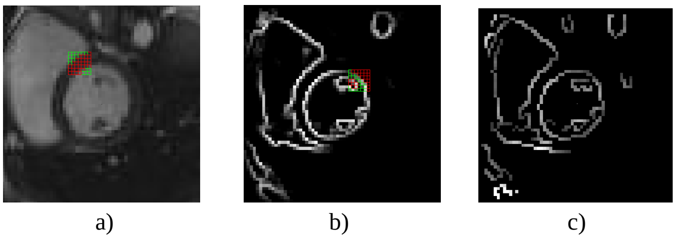

Figure 4: a) one 2D slice of the MRI data set from Section 5 with a local 2-means clustering shown with voxels framed red (low intensity) and voxels framed green (high intensity). b) proposed edge detector with a threshold of 15; the staggered grids are merged for comparison. c) Canny edge detector with standard derivative of 0.5, lower threshold of 10, higher threshold of 20 and scaling factor of 10.

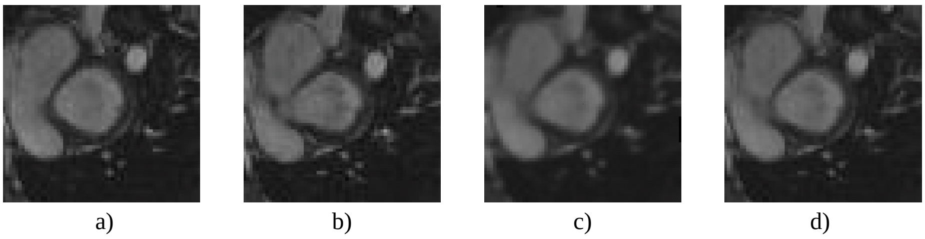

Figure 5: Interpolation between two slices a) and b) using optical flow based interpolation (c)) respectively linear interpolation (d)). c) shows higher contrast at the step edges between blood and tissue. d) shows a linear interpolation with blurred borders. Contrast at the step borders is smaller.

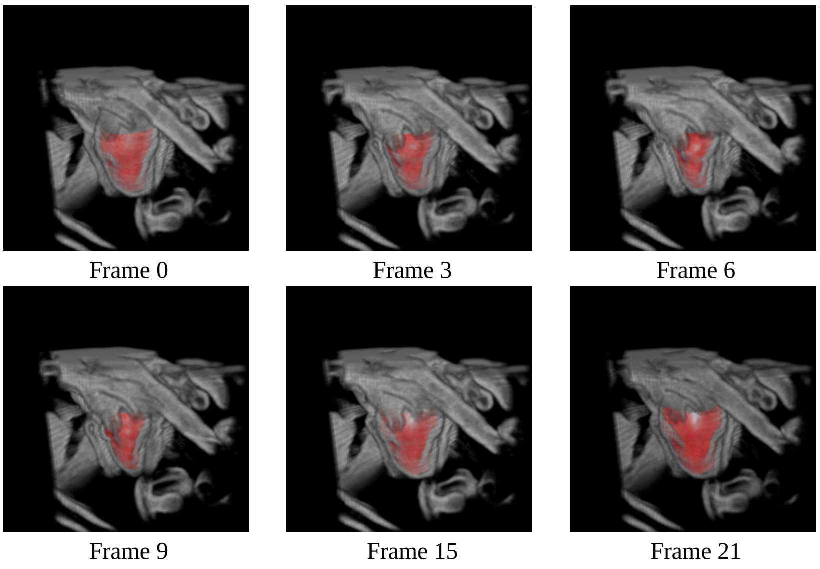

Figure 6: The figure shows the extracted mesh (red) blended with a volume rendering of the result of the boundary detection. Frame 6 shows the surface during the systole of the heart cycle, while frame 15 shows it during the diastole.