Real-time Tracking of the Left Ventricle in 3D Ultrasound Using Kalman Filter and Mean Value Coordinates

Erik Smistad[1,2] and Frank Lindseth[1,2]

[1] Norwegian University of Science and Technology, Trondheim, Norway.
[2] SINTEF Medical Technology, Trondheim, Norway.

Abstract. A method for real-time automatic tracking of the left ventricle (LV) in 3D ultrasound is presented. A mesh model of the LV is deformed using mean value coordinates enabling large variations. Kalman filtering and edge detection is used to track the mesh in each frame. The method is evaluated using the framework of the Challenge on Endocardial Three-dimensional Ultrasound Segmentation (CETUS). The results show that the method is able to robustly track the LV in all sequences with a mean mesh difference of about 2.5 mm.

1 Introduction

Ultrasound is a real-time, flexible and affordable medical image modality which makes it ideal for intraoperative imaging. However, 3D ultrasound images can be hard to interpret and visualize due to noise and other imaging artifacts. Tracking of structures in ultrasound images can be a way to make this easier and provide quantitative measures such as volume size.

The Kalman filter [1] is a method for estimating a state using a series of noisy measurements over time. This method can be used to track meshes in ultrasound images by using a state consisting of translation, rotation, scaling and deformation parameters of a mesh model in addition to edge detection measurements. Jacob et al. [2] developed a tracking method for myocardial borders using a Kalman filter and active contours in 2D ultrasound. Orderud [3] presented a method for tracking the left ventricle (LV) in 3D ultrasound using a deformable contour model. This was later extended to incorporate local deformation using a B-spline surface in [4] and using a subdivision surface in [5]. B-spline and subdivision surfaces can model mesh deformations using a set of control points.

If the surface is simple, only a few control points are needed which results in faster computation of the Kalman filter. However, creating such a model can involve a lot of manual work. In this paper, we present a fully automatic method for tracking a closed surface with free-form surface deformation based on the Kalman filter approach by Orderud et al. [4]. The presented method uses a mesh model with mean value coordinates which is able to deform a complex shape with few control points. It also makes the shape modelling easier as only a surface consisting of a set of points is required, and some calculations such as generating surface points needed for edge detection are avoided. The new method is tested on 3D ultrasound images of the LV of the heart, and evaluated as part of the Challenge on Endocardial Three-dimensional Ultrasound Segmentation (CETUS).

2 Methods

In this section, the different methods used are described. First, the mesh model for the LV is presented together with the method for deforming it. Next, the Kalman filter used to track the mesh in the images is described and finally, a pseudo-code of the complete implementation is provided.

2.1 Mesh model

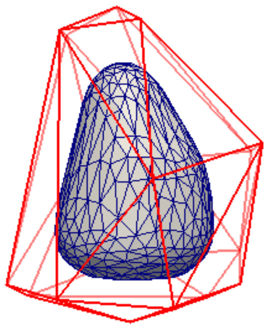


Fig. 1. Mesh model of the LV and the control mesh around (red).

The LV is modelled as a set of points $\mathbf{p}$. The initial mesh is defined by the points $\mathbf{p}_0$ and is first transformed using a local transformation $\mathbf{p}_l = T_l(\mathbf{p}_0, \mathbf{x}_l)$ and then a global transformation $\mathbf{p} = T_g(\mathbf{p}_l, \mathbf{x}_g)$. $\mathbf{x}_l$ and $\mathbf{x}_g$ are the local and global transformation parameters. $\mathbf{p}_0$ was created from the end diastolic reference mesh of the first patient and resampled down to $M = 386$ vertices using the surface simplification method of Garland and Heckbert [6]. The reason for reducing the number of vertices is to increase the speed of the implementation.

Mean value coordinates are used to perform the local deformation of the mesh. This is done using a control mesh $\mathbf{c}$ which has a lot less vertices than the mesh. The control mesh was created from $\mathbf{p}_0$ by resampling it to $N = 18$ vertices using [6] and finally scaling it by 1.5. The mesh is deformed by moving the vertices in the control mesh. Initially, a weight $w_{i,j}$ is calculated between each vertex j in each triangle of the control mesh and vertex i in the mesh $\mathbf{p}_0$. The mean value coordinate weights are calculated as described by Ju et al. [7] using equation (1) where Ψ and θ are the dihedral angles and arc lengths as depicted in Fig. 2.

$$w_{i,a} = \frac{\theta_a - \cos(\Psi_b)\theta_c - \cos(\Psi_c)\theta_b}{2\sin(\Psi_b)\sin(\Psi_c)|\boldsymbol{c}_a - \boldsymbol{p}_i|} \tag{1}$$

The weights for the other vertices in the triangle ($\boldsymbol{c}_b$ and $\boldsymbol{c}_c$) are calculated using the same formula by swapping a with b or c. After all the weights have been calculated, they are normalized as $w'_{i,j} = w_{i,j}/\sum_{k,l} w_{k,l}$. The calculation of the normalized weights are only performed once for the mesh model and is not repeated for every dataset. The local state vector $\boldsymbol{x}_l$ consist of a displacement vector $\boldsymbol{d}_j$ for each of the control mesh' vertices. The local deformation of the mesh is calculated by using the normalized weights and the displacement vectors:

$$\boldsymbol{p}_{l,i} = T_l(\boldsymbol{p}_0, \boldsymbol{x}_l)_i = \sum_{j=0}^{N} w'_{i,j}(\boldsymbol{c}_j + \boldsymbol{d}_j) \tag{2}$$

The global transformation is performed using equation (3) below. The mesh is first moved so that its centroid $\boldsymbol{C}$ is placed in the origin. Next, rotation around each axis is performed as well as scaling. Finally, translation is performed using $\boldsymbol{T}$. As translation, rotation and scaling is used in all three directions, the global state vector $\boldsymbol{x}_g$ consist of 9 values.

$$\boldsymbol{p}_i = T_g(\boldsymbol{p}_l, \boldsymbol{x}_g)_i = \mathbf{R}_z\mathbf{R}_y\mathbf{R}_x\mathbf{S}(\boldsymbol{p}_{l,i} - \boldsymbol{C}(\boldsymbol{p}_l)) + \boldsymbol{C}(\boldsymbol{p}_l) + \boldsymbol{T} \tag{3}$$

Initially, the mesh $\mathbf{p}_0$ is placed automatically in the center of the image and scaled to 0.8 of its original size, effectively placing the mesh model inside the LV. Although it is slightly counterintuitive to do the local transformation first, this greatly simplifies the calculations. For instance, performing the global transformation first would require calculating the weights $w_{i,j}$ in every iteration.

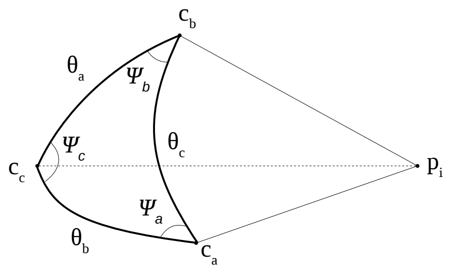
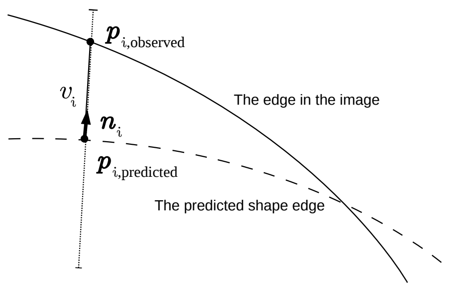


Fig. 2. Notation used for the calculation of mean value coordinates. $\boldsymbol{p}_i$ is a point on the model mesh. $\boldsymbol{c}_a, \boldsymbol{c}_b, \boldsymbol{c}_c$ are control points for one triangle in the control mesh. Ψ and θ are the angles for the spherical triangle formed by these control points.



$\boldsymbol{p}_{i,\text{observed}}$
v_i
The edge in the image
$\boldsymbol{n}_i$
$\boldsymbol{p}_{i,\text{predicted}}$
The predicted shape edge

Fig. 3. Edge detection for a vertex i on the predicted mesh $\mathbf{p}$. A line is created with the center at the vertex position and in the direction of its normal $\boldsymbol{n}_i$. v_i is then the normal displacement between the vertex and the detected edge on this line.

2.2 Mesh tracking using a Kalman filter

As proposed by Orderud et al. [4], the state of a mesh $\mathbf{p}$ is described using both the local and global transformation parameters $\mathbf{x} = [\boldsymbol{x}_l, \boldsymbol{x}_g]$. A Kalman filter with a motion model (4) is used to predict the mesh's state $\bar{\mathbf{x}}_{k+1}$, and the corresponding covariance error matrix $\bar{\mathbf{P}}_{k+1}$ in (5). For the state transition model, diagonal matrices were used with values 1.5 for $\mathbf{A}_1$ and -0.5 for $\mathbf{A}_2$. A diagonal matrix was also used for the process error matrix $\mathbf{Q}$ with values 1.5 for the global and 0.001 for the local transformation parameters.

$$\bar{\mathbf{x}}_{k+1} = \mathbf{A}_1\hat{\mathbf{x}}_k + \mathbf{A}_2\hat{\mathbf{x}}_{k-1} \tag{4}$$

$$\bar{\mathbf{P}}_{k+1} = \mathbf{A}_1\hat{\mathbf{P}}_k\mathbf{A}_1^T + \mathbf{A}_2\hat{\mathbf{P}}_{k-1}\mathbf{A}_2^T + \mathbf{A}_1\hat{\mathbf{P}}_k\mathbf{A}_2^T + \mathbf{A}_2\hat{\mathbf{P}}_{k-1}\mathbf{A}_1^T + \mathbf{Q} \tag{5}$$

The edge detection finds the normal displacement ($v_i = \boldsymbol{n}_i^T(\boldsymbol{p}_{i,\text{observed}} - \boldsymbol{p}_{i,\text{predicted}})$) for each vertex i in the predicted mesh in a 25 mm long line centered at the vertex and in the direction of the normal $\boldsymbol{n}$ as shown in Fig. 3. The edge is detected using the STEP model [8] which entails finding a k that maximizes the following measure where $l(t)$ is the image intensity at step t along the line with step size 0.6 mm (see Fig. 4).

$$\sum_{t=0}^{k} \left| \frac{1}{k+1} \sum_{j=0}^{k} (l(j)) - l(t) \right| + \sum_{t=k+1}^{L-1} \left| \frac{1}{L-k} \sum_{j=k+1}^{L-1} (l(j)) - l(t) \right| \tag{6}$$

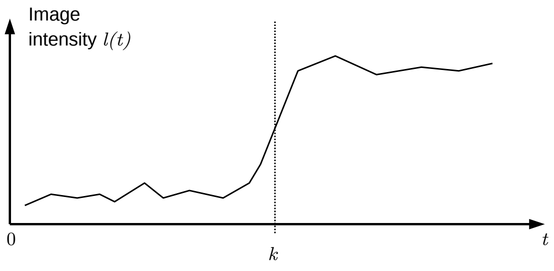
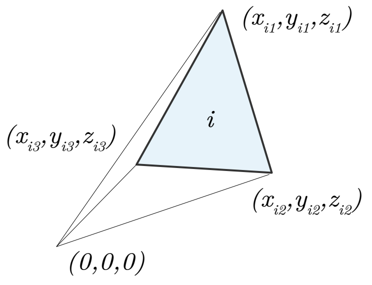


Fig. 4. Edge detection using the STEP model.



(x_{i1}, y_{i1}, z_{i1})
i
(x_{i3}, y_{i3}, z_{i3})
(x_{i2}, y_{i2}, z_{i2})
(0,0,0)

Fig. 5. Tetrahedron formed by a triangle i and the origin.

A measurement noise value r_i is also recorded for each vertex and is calculated based on the edge strength:

$$r_i = \frac{1}{\frac{1}{k+1}\sum_{j=0}^{k} l(j) - \frac{1}{L-k}\sum_{j=k+1}^{L-1} l(j)} \tag{7}$$

However, these edge measurements are nonlinear and thus an extended Kalman filter has to be used in which the observation model is linearized. This is done by calculating Jacobi matrices that relate changes in each vertex i to changes in the mesh state $\mathbf{x}$. The final measurement vector $\boldsymbol{h}_i^T$ is the normal projection of these Jacobi matrices:

$$\boldsymbol{h}_i^T = \boldsymbol{n}_i^T \frac{\partial T_g(\mathbf{p}_l, \boldsymbol{x})_i}{\partial \boldsymbol{x}} = \boldsymbol{n}_i^T \left[\frac{\partial T_g(\mathbf{p}_l, \boldsymbol{x}_g)_i}{\partial \boldsymbol{x}_g}, \frac{\partial T_g(\mathbf{p}_l, \boldsymbol{x}_g)_i}{\partial \mathbf{p}_l} \frac{\partial T_l(\mathbf{p}_0, \boldsymbol{x}_l)_i}{\partial \boldsymbol{x}_l} \right] \tag{8}$$

By assuming that the measurements are independent, the measurement noise covariance matrix $\mathbf{R}$ becomes a diagonal matrix of the measurement noise values r_i. The multiplications of $\mathbf{R}$, the measurement-to-state transition matrix $\mathbf{H}$ and the measurements $\boldsymbol{v}$ becomes a simple summation as shown in equation (9) [3]. This makes the Kalman update equations (10) invariant to the number of measurements which improves speed significantly, as matrix inversion of large matrices is avoided.

$$\mathbf{H}^T\mathbf{R}^{-1}\mathbf{v} = \sum_{i=0}^{M} \boldsymbol{h}_i^T r_i^{-1} v_i \qquad \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H} = \sum_{i=0}^{M} \boldsymbol{h}_i^T r_i^{-1} \boldsymbol{h}_i \tag{9}$$

Using $\hat{\mathbf{P}}_k \mathbf{H}^T \mathbf{R}^{-1}$ as the Kalman gain, the updated state and error covariance estimate becomes:

$$\hat{\mathbf{x}}_k = \bar{\mathbf{x}}_k + \hat{\mathbf{P}}_k \mathbf{H}^T \mathbf{R}^{-1} \mathbf{v} \qquad \hat{\mathbf{P}}_k = (\bar{\mathbf{P}}_k^{-1} + \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H})^{-1} \tag{10}$$

2.3 End-systolic and end-diastolic volumes

The volume of the mesh is calculated for every frame using equation (11) [9]. This equation calculates the signed volume of tetrahedrons formed by each triangle in the mesh and the origin as depicted in Fig. 5. The end-systolic (ES) and end-diastolic (ED) phases are identified using the minimum and maximum volumes respectively from the volumes of the meshes in all image frames.

$$V = \left| \sum_i \frac{1}{6} (x_{i2} y_{i3} z_{i1} - x_{i3} y_{i2} z_{i1} + x_{i3} y_{i1} z_{i2} - x_{i1} y_{i3} z_{i2} - x_{i2} y_{i1} z_{i3} + x_{i1} y_{i2} z_{i3}) \right| \tag{11}$$

The pseudo-code below describes the complete implementation which is written in C++. The entire Kalman filter procedure is repeated 10 times per image frame.

Algorithm 1 Implementation

```
Set initial state $x_0$ and $x_1 = x_0$
$k \leftarrow 1$
for each image frame do
    for a number of iterations do
        Predict state and error for the current frame using Eq. (4) and (5).
        Perform transformations to create the predicted shape $\mathbf{p} = T_g(T_l(\mathbf{p}_0, \boldsymbol{x}_l), \boldsymbol{x}_g)$
        Perform edge detection for each vertex in the mesh $\mathbf{p}$
        Assimilate the measurements using Eq. (8) and (9).
        Estimate the state and error for the current frame using Eq. (10).
        $k \leftarrow k + 1$
    end for
    Calculate volume size $V$ of the current mesh model defined by $\boldsymbol{x}_k$ using Eq. (11).
    if $V < V_{\min}$ then
        $V_{\min} \leftarrow V$
        $\boldsymbol{x}_{ES} \leftarrow \boldsymbol{x}_k$
    end if
    if $V > V_{\max}$ then
        $V_{\max} \leftarrow V$
        $\boldsymbol{x}_{ED} \leftarrow \boldsymbol{x}_k$
    end if
end for
Save the meshes defined by $\boldsymbol{x}_{ES}$ and $\boldsymbol{x}_{ED}$ to disk
```

3 Results

The method was evaluated as part of the Challenge on Endocardial Three-dimensional Ultrasound Segmentation (CETUS). In this challenge, a dataset of 30 sequences of 3D ultrasound volumes of one cardiac cycle was provided. The sequences were collected from both healthy subjects and subjects with a history of myocardial infarction and dilated cardiomyopathy using three different ultrasound probes. The same parameters were used for all subjects. Results for the training and test dataset are gathered in tables 1 and 2 respectively. The tables contain measures such as mean absolute difference (MAD), hausdorff distance (HD) and min and max error all expressed in millimeters. The correlation of the ejection fraction (EF) and stroke volume (SV) was 0.91 and 0.92 for the training dataset and 0.91 and 0.55 for the test dataset. A Bland-Altman analysis of the EF and SV gave the 95% limits of agreements intervals 1.52 ± 12.37 and 0.34 ± 19.14 for the training dataset and 3.87 ± 8.15 and 0.75 ± 20.31 for the test dataset. Fig. 6 show two ultrasound images where the border of the result mesh of the proposed method and the ground truth is illustrated. The average runtime per subject was measured to be 2.1 seconds with a standard deviation of 0.6 seconds. This includes everything from reading data, processing and storing the result meshes to disk. The average runtime per image frame was measured to be 65 ms which enables real-time tracking of the LV. The runtime was measured on a system with an Intel i7-3770 CPU running at 3.4 GHz, 16 GB RAM and a solid-state drive.

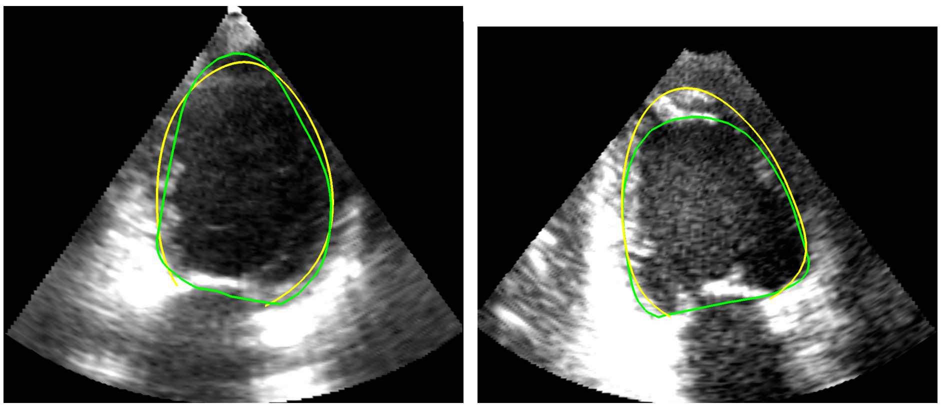


Fig. 6. Result of subject 5 ES to the left and subject 10 ED to the right. The yellow line is the mesh border of the ground truth given by the CETUS organizers and the green line is the mesh border of the proposed method.

4 Discussion

The results show that the presented method is able to automatically and robustly track the LV in 3D ultrasound with a mean mesh difference at about 2.5 mm. However, the results for the training set is slightly better than for the test set which may indicate that the parameters have been overtuned for the training set. Also, the max error was high (~10mm) on some sequences. The image to the right in Fig. 6 illustrates such a case. The experts have included bright parts of the heart's apex in the image, while the proposed method track the inside edge. Thus, to deal with this problem, different edge detection methods may be needed for different parts of the mesh model. The implementation achieved speeds that enable real-time tracking and it is mainly the number of model and control mesh vertices (M and N) that affect the speed. The proposed mesh model which use mean value coordinates is able to model a complex shape with few control points. This may prove useful when tracking more complex shapes in which traditional methods such as B-spline and subdivision surfaces will have to use many control points which reduces speed significantly. Thus, our future work will focus on applying this method to other applications such as tracking the ventricle of the brain in 3D ultrasound for guidance of ventricular drainage procedures.

5 Conclusion

A method for fully automatic real-time tracking of the LV in 3D+t ultrasound was presented. The method was able to track the LV in all 30 sequences and achieved a mean mesh difference of about 2.5 mm. However, the max error was high ($\sim$ 10 mm) on some of the sequences due to failure to detect the LV border in some areas.

References

1. Kalman, R.: A new approach to linear filtering and prediction problems. Journal of Fluids Engineering **82**(1) (1960) 35–45
2. Jacob, G., Noble, J.A., Behrenbruch, C., Kelion, A.D., Banning, A.P.: A shape-space-based approach to tracking myocardial borders and quantifying regional left-ventricular function applied in echocardiography. IEEE transactions on medical imaging **21**(3) (March 2002) 226–38
3. Orderud, F.: A Framework for Real-Time Left Ventricular Tracking in 3D+T Echocardiography , Using Nonlinear Deformable Contours and Kalman Filter Based Tracking. Computers in Cardiology **33** (2006) 125–128
4. Orderud, F., Hansgå rd, J.g., Rabben, S.I.: Real-time tracking of the left ventricle in 3D echocardiography using a state estimation approach. Medical Image Computing and Computer-Assisted Intervention (MICCAI) **10**(Pt 1) (January 2007) 858–865
5. Orderud, F., Rabben, S.I.: Real-time 3D segmentation of the left ventricle using deformable subdivision surfaces. In: Computer Vision and Pattern Recognition, Ieee (June 2008) 1–8
6. Garland, M., Heckbert, P.S.: Surface simplification using quadric error metrics. In: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, New York, New York, USA, ACM Press (1997) 209–216
7. Ju, T., Schaefer, S., Warren, J.: Mean Value Coordinates for Closed Triangular Meshes. ACM Transactions on Graphics **24**(3) (2005) 561–566
8. Rabben, S., Torp, A., Stø ylen, A., Slø rdahl, S., Bjø rnstad, K., Haugen, O., Angelsen, B.: Semiautomatic contour detection in ultrasound M-mode images. Ultrasound in medicine and biology **26**(2) (2000) 287–296
9. Zhang, C., Chen, T.: Efficient feature extraction for 2D/3D objects in mesh representation. In: Image Processing. (2001) 935–938

Subject	MAD	HD	Modified dice	Min error	Max error	EF	Reference EF	SV	Reference SV
1 ES	2.35	6.82	0.1	0	6.82	-	-	-	-
1 ED	1.39	4.01	0.05	0.01	4.01	41.4	44.2	122.1	126.6
2 ES	1.66	6.22	0.08	0.01	6.05	-	-	-	-
2 ED	1.33	4.73	0.06	0	4.59	22.9	19.3	46.3	39.6
3 ES	4.47	8.88	0.26	0.01	8.88	-	-	-	-
3 ED	3.53	7.42	0.16	0.02	7.42	43.6	55	72.2	67.7
4 ES	2.95	9.69	0.15	0.02	9.34	-	-	-	-
4 ED	2.07	8.06	0.1	0.01	7.61	35	24.1	69	46.8
5 ES	1.31	3.97	0.07	0	3.97	-	-	-	-
5 ED	1.58	5.24	0.08	0	4.98	26.4	25.9	32.2	32.5
6 ES	2.33	7.7	0.12	0.01	7.7	-	-	-	-
6 ED	2.43	7.34	0.1	0.01	7.34	54.9	53.6	72.4	64.9
7 ES	3.96	10.55	0.22	0.03	10.55	-	-	-	-
7 ED	2.08	6.67	0.1	0.01	6.67	33.1	47.2	44.9	56.8
8 ES	3.31	9.27	0.17	0.01	9.27	-	-	-	-
8 ED	2.41	7.52	0.1	0.01	7.52	42	48.8	67.3	72.2
9 ES	1.69	6.26	0.08	0	6.26	-	-	-	-
9 ED	1.44	5.21	0.07	0	5.21	47.5	51	66.9	67.9
10 ES	2.32	10.34	0.11	0	9.86	-	-	-	-
10 ED	1.96	11.36	0.09	0.01	10.68	17	16.9	30.3	32.6
11 ES	1.96	6.66	0.13	0	6.31	-	-	-	-
11 ED	1.51	6.46	0.08	0.01	6.36	38.5	34.9	64.5	65.4
12 ES	3.13	9.67	0.1	0.02	9.67	-	-	-	-
12 ED	2.83	8.32	0.09	0.02	7.87	17.1	23.2	57.5	75.5
13 ES	1.9	6.77	0.06	0	6.28	-	-	-	-
13 ED	1.84	5.79	0.05	0.02	5.79	14.1	13.6	58.9	55.3
14 ES	2.86	7.86	0.11	0	7.86	-	-	-	-
14 ED	3.09	7.82	0.11	0.01	7.82	18.3	15.1	48.9	37.1
15 ES	1.98	6.19	0.06	0	6.19	-	-	-	-
15 ED	2.19	7.28	0.07	0	7.28	22.5	24.3	85.7	93
Mean	2.33	7.34	0.10	0.01	7.21	31.62	33.14	62.60	62.26
Std. Dev.	0.80	1.87	0.05	0.01	1.80	12.81	15.25	22.55	24.88

Table 1. Results for the training dataset.

Subject	MAD	HD	Modified dice	Min error	Max error	EF	Reference EF	SV	Reference SV
16 ES	1.48	6.53	0.08	0	6.53	-	-	-	-
16 ED	1.37	5.19	0.07	0	4.89	43	47.3	49.9	56.1
17 ES	1.71	4.2	0.11	0	4.2	-	-	-	-
17 ED	1.5	4.75	0.08	0.01	4.33	38.7	46.9	44.9	54.3
18 ES	3.25	8.02	0.12	0.01	8.02	-	-	-	-
18 ED	2.96	6.77	0.1	0.02	6.77	22.7	26	59.9	57.3
19 ES	3.27	8.63	0.17	0	8.63	-	-	-	-
19 ED	3.61	9.12	0.15	0.01	9.12	39.6	38.7	66.5	49.2
20 ES	2.53	8.09	0.17	0	7.46	-	-	-	-
20 ED	2.01	9.12	0.1	0	8.83	48	55	53.4	61.8
21 ES	2.51	7.98	0.12	0	7.89	-	-	-	-
21 ED	1.79	4.32	0.07	0.01	4.22	32.3	36.1	63.2	67.1
22 ES	3.86	12.95	0.18	0.02	12.95	-	-	-	-
22 ED	2.79	6.93	0.12	0.02	6.93	29.1	38.2	50.5	53.7
23 ES	3.66	12.97	0.2	0	12.97	-	-	-	-
23 ED	4.54	12.3	0.19	0	12.3	49.8	48.5	64.9	46.9
24 ES	2.76	7.59	0.14	0.05	7.59	-	-	-	-
24 ED	1.79	7.06	0.08	0	6.7	27.1	31.8	39.6	44.3
25 ES	2.21	7.96	0.13	0.01	7.13	-	-	-	-
25 ED	2.37	11.68	0.11	0.04	10.82	51.6	56.8	69.8	75
26 ES	3.35	11.4	0.15	0	10.74	-	-	-	-
26 ED	3.52	13.04	0.14	0.04	12.53	30.8	33.9	65.4	78.3
27 ES	3.03	6.37	0.23	0.04	6.37	-	-	-	-
27 ED	2.54	5.3	0.15	0.02	5.3	40	51.9	40.6	43
28 ES	1.96	6.11	0.12	0	6.11	-	-	-	-
28 ED	2.75	7.71	0.13	0.02	7.71	49	52.6	46.7	42.9
29 ES	2.29	5.69	0.16	0	5.69	-	-	-	-
29 ED	2.06	4.67	0.1	0.01	4.67	56.2	55	50.2	43.9
30 ES	4.47	13.19	0.19	0.01	13.19	-	-	-	-
30 ED	5.28	12.84	0.18	0.01	12.84	34.6	31.9	70.7	51.1
Mean	2.77	8.28	0.13	0.01	8.11	39.50	43.37	55.74	54.99
Std. Dev.	0.97	2.94	0.04	0.01	2.91	10.01	10.01	10.60	11.29

Table 2. Results for the test dataset.