Optimization of 3-D Active Appearance Models Using the Inverse Compositional Approach: Applications to Cardiac MRI Segmentation

Alexander Andreopoulos

John K. Tsotsos

Technical Report CS-2004-08

October 28, 2004

Department of Computer Science and Engineering

4700 Keele Street North York, Ontario M3J 1P3 Canada

Optimization of 3-D Active Appearance Models Using the Inverse Compositional Approach: Applications to Cardiac MRI Segmentation

Alexander Andreopoulos and John K. Tsotsos
Dept. of Computer Science and Engineering
York University
Toronto, Canada
M3J 1P3

Abstract

We present an efficient algorithm for the fitting of three dimensional (3-D) Active Appearance Models (AAMs) on short axis cardiac MRI, using an extension of the inverse compositional image alignment algorithm that was recently introduced by Matthews and Baker [8]. We demonstrate its applicability for the segmentation of the left ventricle in short axis cardiac MR images. We perform experiments to evaluate the speed and segmentation accuracy of our algorithm on a total of 477 MR images, acquired from 11 patients. We observe a 60 fold increase in fitting speed compared to a brute force Gauss-Newton optimization and a segmentation accuracy which agrees well with an independent standard. We conclude that this is an efficient and robust algorithm for left ventricle segmentation using 3-D Active Appearance Models.

1. Introduction

In 2001, Cardiovascular Disease (CVD) contributed to almost one third of global deaths. CVD is the leading cause of death in the developed world and by 2010, CVD is estimated to be the leading cause of death in developing countries. Heart disease has no gender, geographic or socioeconomic boundaries [1].

Three dimensional imaging of the heart using imaging modalities such as Ultrasound, Magnetic Resonance Imaging (MRI), and X-ray Computed Tomography, is a rapidly developing area of research in medical imaging [7]. The manual segmentation of short axis cardiac MRI provide clinically useful indicators of the heart function, such as the Ejection Fraction (EF) ratio. However, the manual segmentation is a slow and error prone procedure. Fully automated methods for performing this, are highly desirable.

The main reason why existing segmentation algorithms perform worse than human experts do, is because most methods do not incorporate as an integral part of their functionality a sufficient amount of a-priori knowledge about the 3-D cardiac structure and its temporal deformation [9]. Most methods do not attempt to locate the true anatomical boundaries using expert knowledge, and instead try to perform the segmentation by positioning the segmentation boundaries at the locations of strongest local image features, such as edges.

Active Appearance Models (AAMs) are a promising method for the interpretation of medical images [4, 5, 6]. Recently, there has been a lot of interest in the application of 3-D Active Appearance Models for the automatic segmentation of the left ventricle from short axis cardiac MRI [9, 10], due to their robustness against noisy medical images, and their ability to learn the 3-D structure of the heart and not lead to unlikely segmentations.

Standard numerical optimization methods for the fitting of AAMs, such as gradient descent, are very slow, mainly due to the high number of parameters needing to be optimized. This problem is exacerbated even more when moving from 2-D AAMs to 3-D AAMs. When using 3-D AAMs for the segmentation of the left ventricle, it is not uncommon for such models to use 50-100 parameters. In an effort to deal with this, efficient algorithms for fitting AAMs have been developed [5]. However, the fitting accuracy and the convergence rates of such algorithms are known to be suboptimal in many applications.

Recently, a novel algorithm for the fitting of 2-D AAMs was introduced by Matthews and Baker [8]. Its applicability was demonstrated on artificial data and on real life data for face tracking. In this paper we present an extension of this algorithm for the fitting of 3-D AAMs, when used for the segmentation of short axis cardiac MRI. By definition, short axis cardiac MR images are such that the long axis of the heart is perpendicular to the acquisition image plane. In practice, this means that during the AAM fitting we need to rotate our model only around the long axis of the heart. We take advantage of this fact to design an efficient fitting algorithm. To the best of our knowledge, this is the first effort at extending the inverse compositional image alignment algorithm to 3-D AAMs, and testing its applicability to the

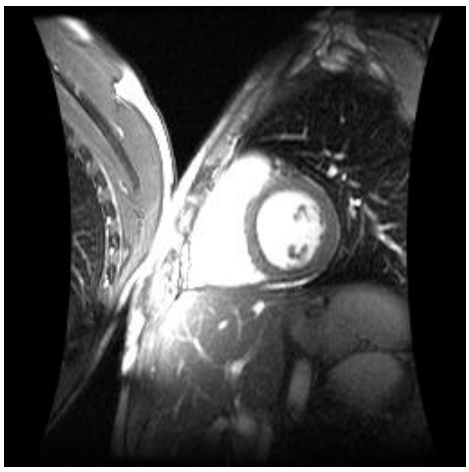

Figure 1: Short axis cardiac MRI.

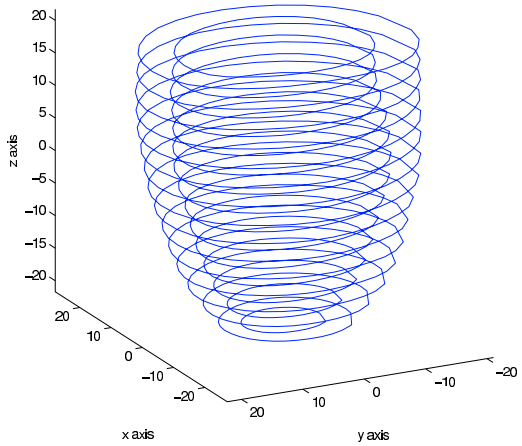

Figure 2: Endocardial and epicardial landmarks stacked on top of each other. Displayed as curves for greater clarity.

interpretation of medical images.

The algorithms described in the literature for fitting AAMs, can be classified as either robust but inefficient gradient descent type algorithms, or as the efficient but ad-hoc algorithms described next. The original AAM formulation uses regression to find a constant matrix $\mathbf{R}$, such that if the current fitting error between the AAM and the image is $\delta\mathbf{t}$, the updated AAM parameters are $\delta\mathbf{p}=\mathbf{R}\delta\mathbf{t}$. In more recent implementations, the estimation of matrix $\mathbf{R}$ is superseded by a faster and simpler method which regards $\mathbf{R}$ as a jacobian matrix of the error function between the AAM and the image [5]. In general there is no reason why the error measure $\delta\mathbf{t}$ should uniquely identify the update parameters $\delta\mathbf{p}$. Such methods lack a sound theoretical basis. Moreover, it has been shown that using a constant matrix $\mathbf{R}$ to estimate the update parameters can lead to incorrect results [8]. However, the constant matrix technique is widely used due to its fitting speed. Later, Matthews and Baker [8] showed how to use the inverse compositional image alignment algorithm to fit 2-D AAMs.

The algorithm we describe in this paper is an extension of [8] to 3-D, under the constraint that all rotations take place around one axis. Our experimental results show it is an accurate, robust and efficient algorithm. Our border positioning errors are significantly smaller than the errors reported for other 3-D AAMs [9] which use the constant matrix approach for the fitting, and are comparable to the errors of Gauss-Newton based fitting. This is good evidence that under thorough clinical validation on a big data set, our algorithm might be proven to be superior than most, if not all, fitting algorithms for 3-D AAMs.

2. 3-D AAMs

We describe our implementation of the 3-D AAM of the left ventricle. It has many similarities to the methodology used in [9, 10]. We begin by giving a quick overview of point distribution models for 3-D AAMs. We then describe how we aligned the landmarks which made up our training set. We end by giving an overview of how we handled appearance variation.

2.1. The Point Distribution Model

Figure 1 shows a short axis cardiac MR image. A stack of such images gives us a volumetric representation of the heart. Manual segmentations of the left ventricle provide us with contours representing the endocardium and epicardium of the left ventricle. By uniformly sampling each of these contours at i_0 points along their arclength, each contour is represented by a set of landmarks. By stacking the landmarks on top of each other we obtain a 3-D representation of the left ventricle's endocardium and epicardium, as shown on figure 2. However, the number of images intersecting the left ventricle is not the same across every patient. Therefore, we need to interpolate between the contours so that every 3-D model is made up of the same number of slices. If we want to create a slice at height z_0 located between two slices, we can simply do the following: From the line segment joining the i^{th} landmark in the two slices, we find the location with height z_0. This gives us the i^{th} landmark in the new slice. In our implementation, we created 15 contour slices, evenly sampled along the z-axis, located between the apex and basal contours.

So, assuming that we have a set of N sample shapes, each sample made up of m landmarks, we can represent each

shape sample as a 3m dimensional vector, since each landmark is made up of 3 coordinates. By applying principal component analysis (PCA) on the distribution of the shape vectors, any shape $\mathbf{s}$ out of the N shapes can be represented as

$$\mathbf{s} = \mathbf{s}_0 + \sum_{i=1}^{n} p_i \mathbf{s}_i \quad (1)$$

for some $\mathbf{p} = (p_1, ..., p_n) \in \Re^n$, where $\mathbf{s_0}$ is the mean shape vector (a.k.a base mesh), and $\mathbf{s_i}$ indicates the i^{th} eigenvector. Notice that we are summing over n eigenvectors $\mathbf{s_i}$ with eigenvalues $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n \geq 0$. These are the n eigenvectors with highest eigenvalues that we found with PCA. We chose n such that it explained around 95% of the variation. Empirically, it has been shown that this is a good value. Higher values tend to lead to PDMs which overfit the training set, and much smaller values lead to PDMs which cannot generalize to new shapes.

2.2. Landmark Alignment

When building a 2-D AAM, we need to make sure that the shapes are aligned with each other so that we remove any difference between two shapes due to a similarity transform. See [4] for more details. This leads to more compact AAMs, which can be described by a smaller number of parameters. In [8] and in our algorithm, this is a necessary step since, as we will see below in the 3-D case, by removing the similarity transform from the training set shapes, the $\mathbf{s}_i$ vectors will be orthogonal to a subspace of the 3-D similarity transforms of $\mathbf{s}_0$. In our algorithm, this was accomplished by using an iterative alignment procedure as described in [4], only that in this case we aligned the shapes so that they were as close to each other with respect to translation, scaling and rotation around only the z-axis. We did not align the shapes with respect to x and y axis rotations since we only wanted our model to handle rotations around the z-axis. The reason will become clearer later on.

2.3. Appearance Variation

We need to model the appearance variation of the shape. We first manually tetrahedrize $\mathbf{s}_0$, as shown in figure 3. This splits the left ventricular volume enclosed by $\mathbf{s}_0$ into tetrahedra. The same landmark connectivity defining the tetrahedra of $\mathbf{s}_0$ can be used to define the tetrahedrization of any shape variation resulting from Eq. (1). Then, we uniformly sample the texture enclosed by each training shape using the same methodology as in [9]. Let the mean texture we get by averaging the sampled textures be $A_0(\mathbf{x})$ and the k eigenvectors we found by PCA, describing around 95% of the texture variation, be $A_1(\mathbf{x}), A_2(\mathbf{x}), ..., A_k(\mathbf{x})$ (where $\mathbf{x}$ denotes the texture coordinate in the base model $\mathbf{s}_0$ coordinate system. Notice that we are abusing terminology here.

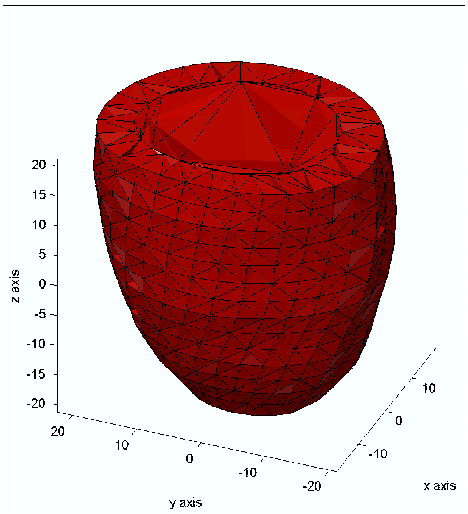

Figure 3: Tetrahedrization of base mesh $\mathbf{s}_0$. Every tetrahedron represents a part of the myocardial muscle or a part of the left ventricle's blood pool.

$A_i(\mathbf{x})$ can be viewed as a geometric vector or as a function of $\mathbf{x}$, the context will make it clear which case it is). Then

$$A(\mathbf{x}) = A_0(\mathbf{x}) + \sum_{i=1}^{k} b_i A_i(\mathbf{x}) \quad (2)$$

defines the different texture variations the model has learned from our training data.

3. The Inverse Compositional Approach to Image Alignment

There is a wealth of literature on image alignment algorithms and the reader is referred to [2, 3] and the references therein for an overview. The inverse compositional approach has been shown to be a fast and reliable image alignment algorithm.

Assume we have a template $A_0(\mathbf{x})$ that we want to align with an image $I(\mathbf{x})$. In the compositional framework for image alignment we compute a warp $\mathbf{W}(\mathbf{x}; \mathbf{\Delta p})$ that is composed with the current warp $\mathbf{W}(\mathbf{x}; \mathbf{p})$(where $\mathbf{p}$ are warp parameters and $\mathbf{x}$ denotes pixel/voxel coordinates), in order to find the update parameters $\mathbf{\Delta p}$ minimizing

$$\sum_{\mathbf{x} \in Domain(A_0)} [I(\mathbf{W}(\mathbf{W}(\mathbf{x}; \mathbf{\Delta p}); \mathbf{p})) - A_0(\mathbf{x})]^2. \quad (3)$$

In the inverse compositional approach we are trying to minimize

$$\sum_{\mathbf{x} \in Domain(A_0(W))} [I(\mathbf{W}(\mathbf{x}; \mathbf{p})) - A_0(\mathbf{W}(\mathbf{x}; \mathbf{\Delta p}))]^2. \quad (4)$$

It can be shown [2] that the solution to this least squares problem is approximately

$$\mathbf{\Delta p} = \mathbf{H^{-1}} \sum_{\mathbf{x} \in Domain(A_0)} [\bigtriangledown A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}}]^T [I(\mathbf{W}(\mathbf{x};\mathbf{p})) - A_0(\mathbf{x})] \quad (5)$$

where

$$\mathbf{H} = \sum_{\mathbf{x} \in Domain(A_0)} [\bigtriangledown A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}}]^T [\bigtriangledown A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}}] \quad (6)$$

and $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ is evaluated at $(\mathbf{x};\mathbf{0})$. It can be shown that within first order, under the condition of a fairly "smooth" warping function $\mathbf{W}$, (4) is equal to

$$\sum_{\mathbf{x} \in Domain(A_0)} [I(\mathbf{W}(\mathbf{W^{-1}}(\mathbf{x};\mathbf{\Delta p});\mathbf{p})) - A_0(\mathbf{x})]^2. \quad (7)$$

This means that the $\Delta \mathbf{p}$ in (5) can also be used to minimize (7). Notice that (7) is an image alignment error measure of the form (3). So once we have found $\Delta \mathbf{p}$ we can update the warp by

$$\mathbf{W}(\mathbf{x};\mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}^{-1}(\mathbf{x};\mathbf{\Delta p}) \quad (8)$$

and go to Eq. (5) to perform another iteration of the image alignment algorithm. This is a very efficient algorithm, known as the inverse compositional algorithm [3]. It is efficient because we can precompute $\nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ and $\mathbf{H}$. This algorithm can easily be used for fitting templates $A(\mathbf{x})$ with appearance variation as described in section 2.3 Eq. (2). In that case, if in Eq. (4) above, $A_0(\mathbf{x})$ is replaced by $A(\mathbf{x})$, with appearance variation as described in section 2.3, then (7) can be shown to be equal to [3]:

$$\sum_{\mathbf{x} \in Domain(A_0)} [proj_{span(A_i)^\perp}(I(\mathbf{W}(\mathbf{x};\mathbf{p})) - A_0(\mathbf{x}))]^2 + \quad (9)$$

$$\sum_{\mathbf{x} \in Domain(A_0)} [proj_{span(A_i)}(I(\mathbf{W}(\mathbf{x};\mathbf{p})) - A(\mathbf{x}))]^2. \quad (10)$$

In practice we estimate (9) by projecting $\nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ onto $span(A_i)^\perp$ in Eq. (5) and (6) above. Then, since the minimum of expression (10) is zero (by choosing the b_i in Eq. (2) appropriately), the error is equal to the value of expression (9).

4. 3-D AAM Fitting Using the Inverse Compositional Algorithm

In this section we show how to fit 3-D AAMs using the inverse compositional approach. In section 4.1 we show how to extend [8] to fit 3-D AAMs with no global similarity transformations. In section 4.2 we show how to fit a 3-D AAM when we allow translations, rotations around only one axis - by convention we will be rotating around the z-axis - and scaling of the coordinate axes.

4.1. Fitting Without Global Shape Transform

We now show how the inverse compositional algorithm can be used for fitting 3-D AAMs without taking care of any global shape similarity transformations (translation,scalings and rotations).

We first need to define $\mathbf{W}(\mathbf{x};\mathbf{p})$, where $\mathbf{p}$ denotes the current landmarks model parameters from Eq. (1), and the $\mathbf{x} = (x, y, z)^T$ parameter denotes a point in the base mesh $\mathbf{s_0}$. Then $\mathbf{W}(\mathbf{x};\mathbf{p})$ denotes the warping of $\mathbf{x}$ under the current warp parameters $\mathbf{p}$. As mentioned above, every base mesh voxel $\mathbf{x}$ lies in a tetrahedron $\mathbf{T_0}$ defined by the vertices $(x_i^0, y_i^0, z_i^0), (x_j^0, y_j^0, z_j^0), (x_k^0, y_k^0, z_k^0), (x_l^0, y_l^0, z_l^0)$. If the current shape parameters of our point distribution model are $\mathbf{p}$, then let the vertices of the deformed tetrahedron $\mathbf{T_1}$ be $(x_i, y_i, z_i), (x_j, y_j, z_j), (x_k, y_k, z_k), (x_l, y_l, z_l)$ which were computed from Eq. (1). $\mathbf{W}(\mathbf{x};\mathbf{p})$ computes the affine transformation of $\mathbf{x}$ from $\mathbf{T_0}$ to $\mathbf{T_1}$. If $\alpha_i, \alpha_j, \alpha_k, \alpha_l$ denote the barycentric coordinates of $\mathbf{x}$ in $\mathbf{T_0}$ given by

$$\begin{pmatrix} \alpha_i \\ \alpha_j \\ \alpha_k \\ \alpha_l \end{pmatrix} = \begin{pmatrix} x_i^0 & x_j^0 & x_k^0 & x_l^0 \\ y_i^0 & y_j^0 & y_k^0 & y_l^0 \\ z_i^0 & z_j^0 & z_k^0 & z_l^0 \\ 1 & 1 & 1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad (11)$$

(remember that by the definition of barycentric coordinates $\alpha_i + \alpha_j + \alpha_k + \alpha_l = 1$ and $0 \leq \alpha_i, \alpha_j, \alpha_k, \alpha_l \leq 1$), then the affine transformation of $\mathbf{x}$ is $\mathbf{W}(\mathbf{x};\mathbf{p})$ and is given by:

$$\alpha_i \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} + \alpha_j \begin{pmatrix} x_j \\ y_j \\ z_j \end{pmatrix} + \alpha_k \begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix} + \alpha_l \begin{pmatrix} x_l \\ y_l \\ z_l \end{pmatrix}. \quad (12)$$

To compute $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$ in Eq. (5) we do the following: For every point $\mathbf{x}$ in the mean tetrahedrization $\mathbf{s_0}$ compute $\mathbf{W}(\mathbf{x};\mathbf{p})$ and sample image I at that location by trilinear interpolation. By a straightforward extension of [8] from 2-D to 3-D, it can be shown that - we skip the proof due to lack of space, see [8] for details -

$$\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \sum_{i=1}^{m} [\frac{\partial \mathbf{W}}{\partial x_i} \frac{\partial x_i}{\partial \mathbf{p}} + \frac{\partial \mathbf{W}}{\partial y_i} \frac{\partial y_i}{\partial \mathbf{p}} + \frac{\partial \mathbf{W}}{\partial z_i} \frac{\partial z_i}{\partial \mathbf{p}}] \quad (13)$$

where

$$\frac{\partial \mathbf{W}}{\partial x_i} = (\alpha_i, 0, 0)^T \pi(\mathbf{x}, i) \quad (14)$$

$$\frac{\partial \mathbf{W}}{\partial y_i} = (0, \alpha_i, 0)^T \pi(\mathbf{x}, i) \quad (15)$$

$$\frac{\partial \mathbf{W}}{\partial z_i} = (0, 0, \alpha_i)^T \pi(\mathbf{x}, i) \quad (16)$$

and

$$\frac{\partial x_i}{\partial \mathbf{p}} = (\mathbf{s}_1^{x_i}, \mathbf{s}_2^{x_i}, ..., \mathbf{s}_n^{x_i}) \quad (17)$$

$$\frac{\partial y_i}{\partial \mathbf{p}} = (\mathbf{s}_1^{y_i}, \mathbf{s}_2^{y_i}, ..., \mathbf{s}_n^{y_i}) \quad (18)$$

$$\frac{\partial z_i}{\partial \mathbf{p}} = (\mathbf{s}_1^{z_i}, \mathbf{s}_2^{z_i}, ..., \mathbf{s}_n^{z_i}). \quad (19)$$

Notice that $\pi(\mathbf{x}, i)$ equals 1 if $\mathbf{x}$ is in a tetrahedron of $\mathbf{s}_0$ having landmark i as its vertex, and is 0 otherwise. Also $\mathbf{s}_j^{x_i}, \mathbf{s}_j^{y_i}, \mathbf{s}_j^{z_i}$ denote the element of $\mathbf{s}_j$ that corresponds to x_i, y_i and z_i respectively. Notice that the summation in (13) is nonzero only for the 4 vertices of the tetrahedron enclosing the current point $\mathbf{x}$ where we are evaluating the jacobian.

By the argument in [8] we see that within first order, $\mathbf{W^{-1}}(\mathbf{x}; \mathbf{\Delta p}) = \mathbf{W}(\mathbf{x}; -\mathbf{\Delta p})$. From (8) we conclude that we need to find a parameter $\mathbf{p}'$ such that $\mathbf{W}(\mathbf{x}; \mathbf{p}') = \mathbf{W}(\mathbf{x}; \mathbf{p}) \circ \mathbf{W^{-1}}(\mathbf{x}; \mathbf{\Delta p})$. We can approximate this quantity by finding a $\mathbf{p}''$ such that $\mathbf{W}(\mathbf{x}; \mathbf{p}'') = \mathbf{W}(\mathbf{x}; \mathbf{p}) \circ \mathbf{W}(\mathbf{x}; -\mathbf{\Delta p})$. The problem here is that piecewise affine warping does not form a group under the operation of composition. In other words the composition of two piecewise affine warps cannot necessarily be described by another piecewise affine warp. We compensate for this in the following way. We first estimate a new position for the landmarks $\mathbf{s}_0$ under the composition of the two warps. So for every one of the landmarks $\mathbf{a}$ in vector $\mathbf{s}_0$ we do the following: We estimate $\mathbf{W}(\mathbf{a}; -\mathbf{\Delta p})$ simply by using Eq. (1) with $-\mathbf{\Delta p}$ as parameter and looking at the resulting vector indices corresponding to landmark $\mathbf{a}$. Then, to compute $\mathbf{W}(\mathbf{W}(\mathbf{a}; -\mathbf{\Delta p}); \mathbf{p})$ we use the following heuristic procedure, which gives good results for the 2-D case in [8] and our 3-D case. For each one of the tetrahedra in $\mathbf{s}_0$ having $\mathbf{a}$ as a vertex, we estimate the destination of $\mathbf{W}(\mathbf{a}; -\mathbf{\Delta p})$ under that tetrahedron's affine warp. Then, we define the value of $\mathbf{W}(\mathbf{W}(\mathbf{a}; -\mathbf{\Delta p}); \mathbf{p})$ to be the average value of the destination of $\mathbf{W}(\mathbf{a}; -\mathbf{\Delta p})$ under those tetrahedra's affine warps. Once we have done this for all landmarks in $\mathbf{s}_0$ we can then estimate $\mathbf{p}''$ by finding the closest vector $\mathbf{p}''$=$\mathbf{p}$ in Eq. (1) satisfying the new landmarks.

4.2. Fitting Without X and Y Axes Rotations

Assume $\mathbf{N}(\mathbf{x}; \mathbf{q})$ is a global transform, which does not rotate the shape around the x and y axes, defined as follows. Let $\mathbf{x} = (x, y, z)^T$ and $\mathbf{q} = (a, b, c, t_x, t_y, t_z)^T$. Then define $\mathbf{N}(\mathbf{x}; \mathbf{q})$ by

$$\mathbf{N}(\mathbf{x}; \mathbf{q}) = \begin{pmatrix} 1+a & -b & 0 \\ b & 1+a & 0 \\ 0 & 0 & 1+c \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}. \quad (20)$$

Notice that $\mathbf{q} = \mathbf{0}$ gives the identity transformation. If we let $a = kcos(\theta) - 1$, $b = ksin(\theta)$ and $c = s - 1$ then

$$\mathbf{N}(\mathbf{x}; \mathbf{q}) = \begin{pmatrix} kcos(\theta) & -ksin(\theta) & 0 \\ ksin(\theta) & kcos(\theta) & 0 \\ 0 & 0 & s \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}. \quad (21)$$

This performs a rotation by an angle θ around the z-axis followed by a scaling by k of the x,y coordinates and a scaling by s of the z-coordinates, followed by a translation by $(t_x, t_y, t_z)^T$. Notice that if we replace the s above by k, then we are performing a three-dimensional similarity transform where we are not rotating around the x,y axes. In other words (20) above has slightly more expressive power than a typical similarity transform since it allows to scale the z-coordinate values by a value different than the scaling of the x,y coordinates. Then $\mathbf{N} \circ \mathbf{W}$ is a warp which can perform both the piecewise affine warp of section 4.1 and the global transform $\mathbf{N}$.

Some may argue against scaling along the z axis independently from the x,y axes scalings, since this is not a similarity transform. But from the test cases we performed later on, this does not lead to incorrect model instances, and instead adds more expressive power to our model, making it easier to fit heart models with different z-axes scalings than the ones in our training set.

As noted above, our base mesh is $\mathbf{s_0} = (x_1^0, y_1^0, z_1^0, ..., x_m^0, y_m^0, z_m^0)^T$. Let

$$\mathbf{s_1^*} = c_1 \mathbf{s_0} = c_1 (x_1^0, y_1^0, 0, ..., x_m^0, y_m^0, 0)^T \quad (22)$$

$$\mathbf{s_2^*} = c_2 (-y_1^0, x_1^0, 0, ..., -y_m^0, x_m^0, 0)^T \quad (23)$$

$$\mathbf{s_3^*} = c_3 (0, 0, z_1^0, ..., 0, 0, z_m^0)^T \quad (24)$$

$$\mathbf{s_4^*} = c_4 (1, 0, 0, ..., 1, 0, 0)^T \quad (25)$$

$$\mathbf{s_5^*} = c_5 (0, 1, 0, ..., 0, 1, 0)^T \quad (26)$$

$$\mathbf{s_6^*} = c_6 (0, 0, 1, ..., 0, 0, 1)^T \quad (27)$$

where c_i is a constant such that $\mathbf{s_i^*}$ is of unit length. Notice that then

$$\mathbf{N}(\mathbf{s_0}; \mathbf{q}) = \mathbf{s_0} + \sum_{i=1}^{6} q_i \mathbf{s_i^*} \quad (28)$$

where $q_1 = \frac{a}{c_1}$, $q_2 = \frac{b}{c_2}$, $q_3 = \frac{c}{c_3}$, $q_4 = \frac{t_x}{c_4}$, $q_5 = \frac{t_y}{c_5}$, $q_6 = \frac{t_z}{c_6}$. If during the shape alignment we aligned the training data such that their center of gravity was at point (0,0,0) (ie:for any training shape their average x,y and z coordinate was 0) then $S^* = \{\mathbf{s_1^*}, \mathbf{s_2^*}, \mathbf{s_3^*}, \mathbf{s_4^*}, \mathbf{s_5^*}, \mathbf{s_6^*}\}$ is an orthonormal set.

4.2.1 Orthonormalizing $\mathbf{S^*}$ and $\mathbf{S}$ and computing the jacobian

Remember that set $S = \{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_n\}$ from Eq. (1) is the set of eigenvectors of the covariance matrix (the permissible modes of deformation). For reasons which will become obvious later, we must make sure that every vector in S^* is orthogonal to every vector in S. The shape alignment that we performed before, should in theory take care of this. In practice, however, this is not usually the case and due to various sources of errors they are not fully orthogonal. We, therefore, have to orthonormalize the two sets S^* and S. In our test cases we did it as follows. For every vector in $\mathbf{v} \in S$ we take its projection $\mathbf{v}'$ onto the space orthogonal to S^* by

$$\mathbf{v}' = \mathbf{v} - \sum_{\mathbf{v}^* \in S^*} (\mathbf{v}^T \mathbf{v}^*)\mathbf{v}^* \tag{29}$$

We call this new set S'. Then, by using an orthogonalization algorithm such as the Gram-Schmidt algorithm we can transform S' into S", where now S" is an orthonormal set. Then every vector in S^* is orthogonal to every vector in S''.

There are two remaining issues that need to be explained. We need to show what is the Jacobian of $\mathbf{N} \circ \mathbf{W}$ which will be used instead of $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ in Eq. (5)-(6), and how to update the parameters from Eq. (7). By a straightforward extension of [8] it can be shown that the jacobian of $\mathbf{N} \circ \mathbf{W}$ is $(\frac{\partial \mathbf{N} \circ \mathbf{W}}{\partial \mathbf{q}}, \frac{\partial \mathbf{N} \circ \mathbf{W}}{\partial \mathbf{p}}) = (\frac{\partial \mathbf{N}}{\partial \mathbf{q}}, \frac{\partial \mathbf{W}}{\partial \mathbf{p}})$

4.2.2 Updating the Parameters

In the same way that we noted in section 4.1, to within first order $(\mathbf{N} \circ \mathbf{W})^{-1}(\mathbf{x}; \mathbf{\Delta q}, \mathbf{\Delta p}) = \mathbf{N} \circ \mathbf{W}(\mathbf{x}; \mathbf{-\Delta q}, \mathbf{-\Delta p})$. We can use this approximation to calculate

$$\mathbf{N} \circ \mathbf{W}((\mathbf{N} \circ \mathbf{W})^{-1}(\mathbf{s_0}; \mathbf{\Delta q}, \mathbf{\Delta p}); \mathbf{q}, \mathbf{p}) \tag{30}$$

(the new locations of the landmarks $\mathbf{s}_0$) by using the method of section 4.1 for composing two warps. Once we have estimated the new landmark positions $\mathbf{s}^\dagger$ from Eq. (30), we need to find new values for $\mathbf{p}$ and $\mathbf{q}$ such that $\mathbf{N} \circ \mathbf{W}(\mathbf{s_0}; \mathbf{q}, \mathbf{p}) = \mathbf{s}^\dagger$. First notice that

$$\mathbf{N} \circ \mathbf{W}(\mathbf{s_0}; \mathbf{q}, \mathbf{p}) = \mathbf{N}(\mathbf{s_0} + \sum_{i=1}^{n} p_i \mathbf{s_i}; \mathbf{q}) \tag{31}$$

which can be rewritten as

$$\mathbf{N}(\mathbf{s_0}; \mathbf{q}) + \begin{pmatrix} 1+a & -b & 0 \\ b & 1+a & 0 \\ 0 & 0 & 1+c \end{pmatrix} \sum_{i=1}^{n} p_i \mathbf{s_i} \tag{32}$$

where the summations above are taking place over all vectors $\mathbf{s_i} \in S''$. The matrix multiplication in (32) above with the 3m dimensional vector $\mathbf{s_i}$, indicates the result of multiplying every triple of adjacent x,y,z coordinates by the matrix.

By using (32), we see that $\mathbf{N} \circ \mathbf{W}(\mathbf{s_0}; \mathbf{q}, \mathbf{p}) = \mathbf{s}^\dagger$ can be rewritten as

$$\mathbf{s_0} + \sum_{i=1}^{6} q_i \mathbf{s_i^*} + [(1+a) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \sum_{i=1}^{n} p_i \mathbf{s_i}] +$$
$$[b \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \sum_{i=1}^{n} p_i \mathbf{s_i}] +$$
$$[(1+c) \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \sum_{i=1}^{n} p_i \mathbf{s_i}] = \mathbf{s}^\dagger. \tag{33}$$

We observe that the three terms in Eq. (33) above that are multiplied by 1+a, b and 1+c, are orthogonal to the vectors in S^*, since S^* and S'' are orthogonal. This is most difficult to see for the fourth term in (33) that is multiplied by b. This term is orthogonal to the vectors in S^* because for each vector in S^*, if we switch the values of the x and y coordinates and change the sign of one of the two coordinates, the resulting vector still belongs to span(S^*). Therefore, from (33) we get that

$$q_i = \mathbf{s_i^*} \cdot (\mathbf{s}^\dagger - \mathbf{s_0}). \tag{34}$$

Once we know the parameters $\mathbf{q}$, we can find the exact inverse of $\mathbf{N}(\mathbf{s}^\dagger; \mathbf{q})$ as

$$\mathbf{N^{-1}}(\mathbf{s}^\dagger; \mathbf{q}) =$$
$$\begin{pmatrix} 1+a & -b & 0 \\ b & 1+a & 0 \\ 0 & 0 & 1+c \end{pmatrix}^{-1} [\mathbf{s}^\dagger - \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}] \tag{35}$$

where $\mathbf{q} = (a, b, c, t_x, t_y, t_z)$ and where we have again abused terminology in (35) so that the subtraction by $(t_x, t_y, t_z)^T$ implies subtracting this vector from every corresponding triple of x,y,z coordinates in $\mathbf{s}^\dagger$, and the matrix multiplication has the same meaning as in (32). Then

$$p_i = \mathbf{s_i} \cdot (\mathbf{N^{-1}}(\mathbf{s}^\dagger; \mathbf{q}) - \mathbf{s_0}) \tag{36}$$

and we have estimated the new parameters $\mathbf{p}$ and $\mathbf{q}$.

5. Experimental Results

We trained and fitted the model on a data set of 11 short axis cardiac MR image sequences with a temporal resolution of 20 time samples. The slices were acquired under breathhold in end-expiration. Each image's resolution was 256×256 pixels. The number of slices intersecting the left ventricle ranged between 7 and 10. The voxel dimensions ranged between $1 \times 1 \times 5.76$ and $1 \times 1 \times 7.67$. The implementation and all test cases were done on an Intel Xeon 3.08Ghz with 3GB RAM using MATLAB 6.5.1.

A manual tracing of the left ventricular endocardium and epicardium was acquired from each patient during the 6 temporal samples closest to diastole. The manual tracing was performed on all slices where both the endocardium and epicardium of the left ventricle was visible. As it is common in clinical practice, the endocardial contours were drawn behind the papillary muscles and trabeculae. In each slice, two reference points were placed on the endocardial and epicardial contours, at the point closest to the posterior junction of the right and left ventricles. This point served as a reference point from which we started sampling the contour, to get its corresponding landmarks, as described in section 2.1.

We used the 6 temporal samples closest to the end diastole of each patient to train our model, which gave us a total of 66 training examples. We trained the model using a leave-one-out approach and by using the full 66 training examples data set. In the leave-one-out approach, if we wanted to fit an AAM on a certain time sample of an MR sequence, we trained our model on the 60 training examples not belonging to the same patient. This gave us a total of 66 test cases. We also trained our model on the entire data set and then fitted the model on each of the 66 samples. This gave us an indication of our algorithms accuracy in the presence of a large training set.

We compared the segmentation accuracy of the algorithm described in this paper and of a standard Gauss-Newton optimizer which simultaneously optimized the model parameters $\mathbf{p}$, the global shape parameters $\mathbf{q}$ and the appearance parameters b_i. Other algorithms that are commonly used for the fitting of AAMs, as described in the introduction, are too dependent on the implementation parameters and their results could easily be biased, so we decided not to test them here. We compared our algorithm with a Gauss-Newton optimizer for two reasons: To emphasize the gain in speed that our algorithm can provide compared to brute force approaches and to investigate our algorithms fitting accuracy against an exhaustively tested algorithm whose convergence properties are well understood and validated, such as Gauss-Newton optimization.

5.1. Quantitative Validation Method

We manually translated, rotated and scaled $\mathbf{s_0}$ until our model fitted well the median MR slice of the left ventricle. The same initialization was used when fitting on an image stack a model trained with the leave-one-out approach and the model trained with all the data. We estimated the fitting accuracy of our algorithm in the following way. For each of the image slices intersected by our model, we extracted the endocardial and epicardial contours of our models intersection with the image. If the MR slice had been manually segmented, we sampled the two extracted contours at 50 evenly spaced points on the endocardial contour and 50 points on the epicardial contour, and for each one of those points, we found the smallest distance of the point from the manually traced contour of the endocardium and epicardium respectively. The mean distance of the 100 extracted distances gives the average distance of a point on a contour to the manually traced contour and is a metric indicating how well our model has segmented the particular image. We also visually inspected all the segmented images to verify the correctness of the segmentation.

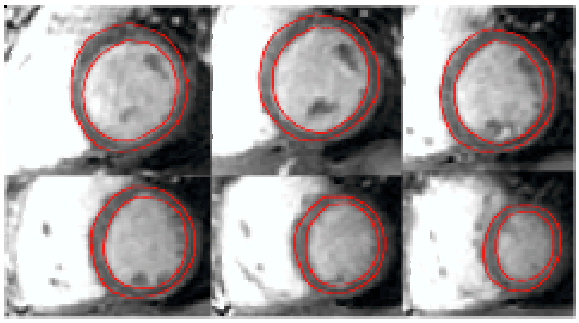

Figure 4: Resulting segmentation of a few slices.

5.2. Results

Table 1 gives the fitting speed, as well as the average segmentation error for the images that had been manually segmented and which were also intersected by the AAM model. Figure 4 shows the resulting segmentations for some slices. The total number of slices segmented varied slightly depending on the fitting algorithm, the training method used, and the subsequent z-axis extension, and ranged between 468 and 477.

The inverse compositional algorithm described in this paper fits the model approximately 60 times faster than a typical brute force Gauss-Newton optimization. The error of the inverse compositional algorithm is comparable or even significantly less than the error of standard Gauss-Newton optimization, and is well below previously reported errors for 3-D AAMs which use different fitting algorithms [9]. We suspect that the reason why Gauss-Newton gives a slightly greater error in one case is because Gauss-Newton has to optimize the appearance parameters also, while the inverse compositional algorithm projects them out. Our models were made up of 30 shape parameters, 20 appearance parameters and the 6 global shape parameters $\mathbf{q} = (a, b, c, t_x, t_y, t_z)$, for a total of 56 parameters. The inverse compositional algorithm proved to be extremely sensitive to global intensity changes in the images. We handled this by normalizing the left ventricle's intensity in each image to a common mean and standard deviation before training and fitting our models. It should be noted that Gauss-Newton optimization was not very sensitive to global inten-

Results	Average Slice Error in Millimeters	Standard Deviation of Error in Millimeters	Average Slice Error in Pixels	Standard Deviation of Error in Pixels	Average Number of Seconds till Convergence
Leave 1 out with ic algorithm	1.60	0.71	1.28	0.59	10.59
Leave 1 out with Gauss-Newton	1.58	0.69	1.25	0.55	626.40
Full data set with ic algorithm	0.88	0.45	0.71	0.39	8.06
Full data set with Gauss-Newton	1.21	0.49	0.95	0.37	465.15

Table 1: Fitting errors and fitting speed of inverse compositional (ic) and Gauss-Newton algorithm.

sity changes. The AAM never got confused by the papillary muscles and trabeculae, which tend to be a problem for most cardiac segmentation algorithms. Most cases of erroneous segmentations involved the epicardial contour, which didn't extend fully to the edge of the epicardium and got stuck inside the myocardium. The reason for this is most likely because the model has no knowledge of the surrounding texture. A multiscale fitting scheme might fix this.

6. Summary and Conclusions

We presented an efficient and robust algorithm for fitting 3-D AAMs on short axis cardiac MRI. To the best of our knowledge this is the first attempt at using the inverse compositional algorithm for fitting 3-D AAMs on medical images. It gives rapid segmentation results with a precision at least as good as that of previously reported results and comparable to brute force Gauss-Newton optimization. Clinical validation of the method on a much larger training set is necessary, if we want to have more reliable conclusions. Improvements to the algorithm include using a multiscale approach to do the fitting, incorporating knowledge about the texture variation in the neighborhood surrounding the left ventricle, extending the algorithm to handle rotations around the x and y axes, and making the model extend fully along the z axis so that it segments all image slices. In conclusion, we believe that 3-D AAMs are one of the most promising methods for solving the segmentation problem of cardiac MRI. The optimization algorithm presented here has given us encouraging results, leading us to believe that it has the potential to become the method of choice for solving this problem.

Acknowledgments

We thank Dr. Paul Babyn and Dr. Shi-Joon Yoo of the Department of Diagnostic Imaging at the Hospital for Sick Children in Toronto for providing us with the MRI data. JKT holds a Canada Research Chair in Computational Vision and acknowledges its financial support. AA holds an NSERC PGS-M and acknowledges its financial support.

References

[1] American Heart Association, "International Cardiovascular Disease Statistics," [Online]. Available: http://www.americanheart.org, 2004.

[2] S. Baker, R. Goss, and I. Matthews, "Lucas-Kanade 20 Years on: A Unifying Framework," *International Journal of Computer Vision*, Vol. 56, No. 3, pp. 221-255, 2004.

[3] S. Baker, and I. Matthews, "Equivalence and Efficiency of Image Alignment Algorithms," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* Vol. 1, pp. 1090-1097, 2001.

[4] T.F. Cootes, "Statistical Models of Appearance for Computer Vision," [Online]. Available: http://www.isbe.man.ac.uk/ bim/Models/app_models.pdf

[5] T.F. Cootes, G. Edwards, and C. Taylor, "Active Appearance Models," In *Proceedings of the European Conference on Computer Vision,* Vol. 2, pp. 484-498. 1998.

[6] T.F. Cootes, and C.J. Taylor, "Statistical Models of Appearance for Medical Image Analysis and Computer Vision," In *Proceedings of SPIE Medical Imaging,* Vol. 4322 pp. 236-248, 2001.

[7] A.F. Frangi, W.J. Niessen, and M.A. Viergever, "Three-Dimensional Modeling for Functional Analysis of Cardiac Images: A Review," *IEEE Transactions on Medical Imaging,* Vol.20, No. 1, pp. 2-25, 2001.

[8] I. Matthews, and S. Baker, "Active Appearance Models Revisited," *International Journal of Computer Vision*, Vol. 60, No. 2, pp. 135-164, 2004.

[9] S. C. Mitchell, J. G. Bosch, B. P. F. Lelieveldt, R. J. van der Geest, J. H. C. Reiber, and M. Sonka, "3-D Active Appearance Models: Segmentation of Cardiac MR and Ultrasound Images," *IEEE Transaction on Medical Imaging,* Vol.21, No. 9, pp. 1167-1178, 2002.

[10] M.B. Stegmann, "Generative Interpretation of Medical Images," doctoral dissertation, Dept. of Informatics and Mathematical Modelling, Technical University of Denmark, 2004.