University of Twente
Enschede - The Netherlands

Signals & Systems

Analyzing Joint-Measurements Using 3D Scans and Statistical Models

M.Sc. Thesis

Thijs Withaar

University of Twente
Department of Electrical Engineering,
Mathematics & Computer Science (EEMCS)
Signals & Systems Group (SAS)
P.O. Box 217
7500 AE Enschede
The Netherlands

Report Number: SAS032-05
Report Date: December 7, 2005
Period of Work: 10/01/2004 - 12/15/2005
Thesis Committee: Prof. dr. ir. C.H. Slump
Ir. J.A. Kauffman
Dr. ir. F. van der Heijden

Abstract

This thesis describes a method to analyze how hand positioning affects radiograph based joint space width (JSW) measurements. These measurements are used in scoring methods, to assess the progression of rheumatoid arthritis. While these measurements are being used in practice, the accuracy of these measurements is not known very well.

This thesis analyzes the effects of rotation of the hand with respect to the projection plane on the measurements. It also presents a method which can be used to estimate this rotation using a two-dimensional radiograph. Therefore, a statistical model of the radiograph is made. This model is then fitted to radiographs using an optimization method. Next, high-resolution 3D data is used to determine the effect of rotation on the measured JSW. Projections of the 3D data are made under varying projection angles. The JSW for each of the projections is then measured using an automated method. The results show that the projection angle has a large effect on the measured JSW.

Contents

Nomenclature

- A pixel is an element describing the intensity of an image at a 2D point.
- A voxel is an element describing the intensity of a volume at a 3D point.
- A set of N 2D coordinates is represented by a vector of $2N$ elements, containing the x and y coordinates.

$$c = [x_1, \ldots, x_N, y_1, \ldots, y_N] \tag{1}$$

- An image is considered to be a function, which outputs a vector of pixel values z for a given set of coordinates c. For an image I with a size of N pixels, this gives the following notations:

$$\begin{aligned} z &= I(c) \\ z &= [z_1, \ldots, z_N] \end{aligned} \tag{2}$$

- The shape model determines a set of coordinates, using the shape parameters $\mathrm{p_s}$. $\mathbf{U}$ is a set of eigenvectors, $\mathrm{c_0}$ is the mean shape of the model.

$$\mathrm{c} = \mathrm{c_0} + \mathbf{U}\mathrm{p_s} \tag{3}$$

- The texture model T consists of a mean texture T_0, the eigenvectors T_i and s texture modes. The modes are controlled by the texture parameters λ_i. The texture model can be represented by:

$$\mathrm{I} = \mathrm{T_0} + \sum_{i=1}^{s} \mathrm{T_i}\lambda_i = \mathrm{T_0} + \mathbf{T}\lambda \tag{4}$$

- An image can be deformed by a warp function. The warp function outputs a set of $2N$ coordinates, the x and y coordinates used for the image function. The function generates these coordinates using K shape parameters. The warped image I_w is generated by evaluating the image at the coordinates given by $W(p)$.

$$\begin{aligned} p &= [p_1, \ldots, p_K] \\ c &= W(p) \\ I_w &= I(W(p)) \end{aligned} \tag{5}$$

1

Introduction

Much research is done on medical image processing. One of the applications is the automatic analysis of the images. This thesis describes a method to analyze x-ray images of the hand. This is done to assess the progression of rheumatoid arthritis (RA).

Rheumatoid arthritis is an autoimmune disease where the immune system causes joint damages [17]. RA causes pain and stiffness which makes common tasks as walking and writing difficult and painful. It can eventually lead to permanent disability. There is no cure for RA, but there are various medicines that slow down the progression of the disease and relief the symptoms. For effective treatment, early and accurate diagnosis is necessary.

Currently, diagnosis is done with the use of radiographs of the hands and sometimes the feet. A physician diagnoses disease progression by giving a score for the progression of the disease. There are many different methods for scoring radiographs [3]. Most methods work by combining grades for bone erosion and joint space narrowing (JSN). A drawback of the analysis with these methods is that the results are dependent on the subjective reader. The results will vary in time and per physician, even the order in which the radiographs are analyzed is important [19]. If the grading can be done automatically, the results will be less observer dependent.

This thesis focusses on joint space width (JSW) measurements using radiographs. By measuring the JSW over time, the progression of the disease can be determined. Since a radiograph is a two-dimensional image of a three-dimensional object, direct measurement of a distance between two edges in the image could be affected by the projection angle of the radiographs. For healthy patients, the hand can be fixed into a particular position before taking the radiograph. For patients with RA this can be painful or impossible. Therefore, the effect of the projection angle on the joint space width measurement has to be investigated.

The first two chapters of this thesis describe a method to approximate an image using only a few parameters. With these parameters, different images

can be compared without the need of designing special algorithms for each different feature that should be extracted from an image. Chapter 2 covers the first part of this method. It describes a model that can approximate images of a certain type, using only a few parameters. After defining the model, it is possible to derive a method to fit the model to an image. Since a closed form solution does not exist, an iterative method is derived in chapter 3.

Chapter 4 shows how a 2D image can be calculated from 3D volume data. This is done by calculating projections of the 3D data. The chapter first describes the projection method, and then uses projections of a high-resolution data set to analyze the sensitivity of JSW measurements to rotation of the hand.

Next, chapter 5 presents test results for various parts of the previous chapters. The tests are done in the same order as the order in which the parts are covered in the previous chapters. First, the effect of the number of model parameters is analyzed. After the model parameters have been determined, the accuracy and sensitivity to initialization of the model is assessed. By fitting the model to a set of test images, the precision of the model fitting method is investigated. The last test is a short test on the accuracy of the projection algorithm. Finally, conclusions are drawn in chapter 6.

2

The image model

The method presented in this report is based on a model of the x-ray image of a bone. The purpose of the model is to characterize images of that bone from different patients, using only a few parameters. By fitting the model to an image, information can be extracted from these parameters.

In literature many different methods to generate a model are described, an overview is given in [5]. The models can be either models of edges or of models full images. Using a number of parameters, the models can be deformed. Most methods constrain the deformations to smooth deformations. The constraints are often based on the restriction of deformation to low order polynomials or physical models [20]. The model presented in this thesis has constraints based on a set of sample data. This has the advantage that deformations are restricted to deformations which are known to occur. This allows for strict constraints, while still allowing a deformation close to the optimal one. The model in this thesis models the full image, instead of separate edges. This way all image data is used, which can result in a higher precision than using only parts of the image to detect features.

The model in this thesis is an active appearance model (AAM) [6], with a modified fitting method [13]. The model is generated by collecting statistics from a set of training-images to determine common variations between the images. The AAM consists of two parts. The first part is the shape model. To generate this model, common points are marked in a set of images. This process is called annotation and is described in section 2.1. Using the annotations and the pixel data a model can be made. The model is generated using singular value decompositions (SVD), which will be explained in section 2.2. Using the SVD, the shape model can be calculated, as is shown in section 2.3. The second part of the AAM is the texture model. Section 2.4 presents a method to calculate the texture model using the annotation and the mean shape of the shape model.

When the model has been generated, it is possible to approximate an image by adjusting the model parameters. First, the texture is adjusted and

then the shape is deformed to match the image. This deformation is done with a piecewise affine warp, as shown in section 2.5.

2.1 Annotation

To generate a model, a number of sample shapes are required. These shapes are generated manually. This section shows how to generate a set of example shapes. The shapes are made by manually labelling features consistently across a set of images. The labelling process is called annotation. The annotation is done in two steps. First the annotation points of the model should be defined. Next, these points have to be marked in each image. After the annotation has been done, all annotations consisting of lines and points are converted into a set of points.

2.1.1 Defining the model

A problem with the first step of the annotation is that the set of images used in this thesis does not have many points that can be easily defined. The edges between bone and tissue, however, are easily recognized.

Figure 2.1 shows an example of an annotation. Most points are connected to form curves. One point, on the lower left on the image, is not connected. This point is placed on a corner that is easily recognized on most radiographs. The curves are made by performing a spline interpolation between the points.

The edges of a bone can be determined very accurately in one direction, but in the direction alongside the edge it is very hard to mark points consistently. The lines are therefore converted into a set of equidistant points, after they have been annotated. This set of points can be larger than the number of points used to define the line. This makes it possible to annotate smooth shapes using only a few number of points.

2.1.2 Converting the annotation to a set of points

Figure 2.1 shows an annotated image. Curves are put on top of all edges which are easily recognized in all images. A point on the lower left edge of the bone is also marked. The shape model, presented later on, can only be calculated from a set of points. All curves are therefore converted to sets of equidistant points.

The curves are made by calculating a cubic spline between the points. The x and y-coordinates of the curve are parameterized as a function of t. t is an intermediate variable ranging from 1 to the number of points defining the curve. The length of a part of the curve cannot be directly estimated from the values of t. Since there is no closed form solution to calculate the length of a spline [9], a numerical approximation is made.

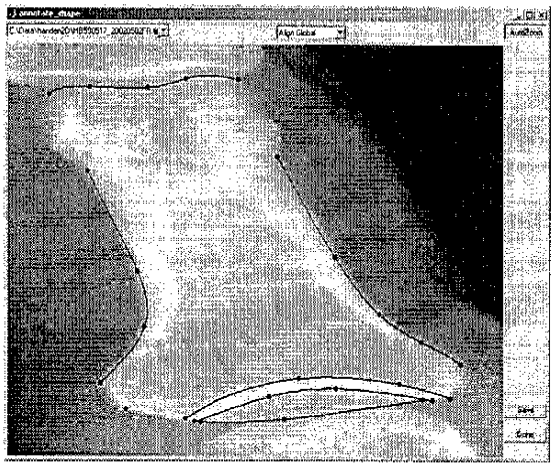


Figure 2.1: Example of an annotated image

The approximation starts by calculating a large set of points from a spline. These points are used to approximate the spline by a set of linear segments. By increasing the number of points, the approximation error can be arbitrarily small. The length of the spline can be estimated by summing the lengths of the line segments. For this thesis, 200 points are calculated on each spline.

Using the lengths of the line segments, a set of values for t can be calculated that gives a set of equidistant points. This is done to generate the desired number points for a curve. For each curve such a set of points is generated, these sets are combined and stored as the shape of the model for a particular image.

2.2 Singular value decomposition

Using the shape and pixel data from the annotated images, a model of an x-ray image can be made. The model for both the shape and the texture is generated using the singular value decomposition of the different shapes and pixel values. Next a short description of the SVD and its properties is given [21].

Figure 2.2 shows an example data set to illustrate the SVD. A two-dimensional data set is plotted as a cloud of points. It can be seen that the horizontal and vertical position of a point are not completely independent. The SVD can be used to determine this relation.

With an SVD a $M \times N$ matrix $\mathbf{X}$ can be written as a multiplication of the matrices $\mathbf{U}$, $\mathbf{S}$ and $\mathbf{V}$, as shown in equation 2.1. These three matrices have certain properties. $\mathbf{U}$ is an $M \times M$ matrix and $\mathbf{V}$ is of size $N \times N$.

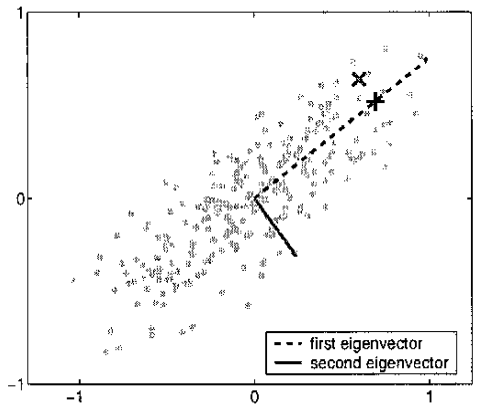


Figure 2.2: Illustration of the SVD properties for 2D data

Both $\mathbf{U}$ and $\mathbf{V}$ have orthogonal columns. $\mathbf{S}$ is a $M \times N$ diagonal matrix with the singular values on the diagonal. The singular values are sorted by their size, so the first eigenvector always has the largest singular value.

$$\mathbf{X} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^T \tag{2.1}$$

In figure 2.2, these properties are illustrated. A coordinate can be represented by scaling two vectors, one for the horizontal and one for the vertical axis. The scaling factors are collected in the matrix $\mathbf{X}$. For this example, an SVD is performed on this matrix. The columns of $\mathbf{V}$ are plotted as the first and second eigenvector. The first eigenvector is in the direction of the largest variation. One data point is plotted as $\times$, it's projection onto the first eigenvector is plotted as a $+$. The data point and the projection are quite close to each other. All data points can be approximated by scaling the first eigenvector, instead of scaling the two vectors parallel to the horizontal and vertical axes. The higher the horizontal and vertical coordinates of the points are correlated, the better this approximation is. This property is used to generate the models in this thesis; a training-set is used to calculate all the eigenvectors of the shape and the texture. By keeping only the largest few eigenvectors, all the data in the training-set can be approximated using only a few scaled eigenvectors.

The SVD is calculated using MATLAB. This software can also calculate an 'economy size' SVD. If $M > N$, only the first N columns of $\mathbf{U}$ are calculated. This is useful for high dimensional data, used in this thesis. The texture model can have over 10.000 pixels, but is generated using only tens of training images, giving only tens of non-zero eigenvalues. Performing an SVD on this data would result in a very large matrix. The matrix $\mathbf{U}$ resulting from the SVD on $\mathbf{X}^T$ is equal to $\mathbf{V}$ resulting from the SVD on $\mathbf{X}$ (equation 2.2). By calculating the 'economy size' SVD of $\mathbf{X}^T$, $\mathbf{V}$ can be

calculated as a matrix of size $M \times N$. If there are only N training-shapes, the last $M - N$ eigenvectors have an eigenvalue of zero, and are not used in the model.

$$\mathbf{X}^T = (\mathbf{U}\ \mathbf{S}\ \mathbf{V}^T)^T = \mathbf{V}\ \mathbf{S}\ \mathbf{U}^T \tag{2.2}$$

2.3 The shape model

Using the shapes of the annotated images, a shape model can be made. The model is made using the SVD. The previous section showed an example for two dimensional data, a shape with n points has $2n$ dimensions; two coordinates for each point.

To calculate the shape model, each shape is represented by a vector c. The vector consists of all x and y coordinates of the points in the shape. The vectors of the shapes are then combined into a matrix $\mathbf{X}$, as shown in equation 2.3.

$$\mathrm{c} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \\ y_1 \\ \vdots \\ y_N \end{bmatrix}, \quad \mathbf{X}^T = [\mathrm{c}_1, \ldots, \mathrm{c}_n] \tag{2.3}$$

All shapes should have an equal influence on the model. A shape with large coordinate values should not have a larger influence on the result of the SVD than a small shape. To prevent this, all shapes should be scaled to one another. To generate a texture model and to fit the model to an image, the mean shape should be known. The mean is calculated and removed from the data in $\mathbf{X}$, so it is a separate model property. To properly calculate the mean, all shapes are aligned to one another, which also removes rotation and translation from the data in $\mathbf{X}$.

Since the mean shape cannot be calculated without aligning the shapes, the scaling and rotation are removed by aligning all shapes to an arbitrarily chosen shape from the set. To simplify the notation of equation 2.4, a shape is represented as a vector of complex numbers, where each complex number represents the 2D coordinate of a point of that shape. Two shapes A and B are aligned by finding a transformation that minimizes the sum of squares of the differences of the coordinates. To calculate the transformation, the least-square solution x of equation 2.4 is calculated. The scale is given by the absolute value of x_1, the angle ϕ between the shapes is given by equation 2.5 and x_2 is the translation between the shapes.

$$\mathrm{B} = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} A_1 & \dots & A_n \\ 1 & \dots & 1 \end{bmatrix} \tag{2.4}$$

$$\phi = \arctan(\frac{\Im(x_1)}{\Re(x_1)}) \tag{2.5}$$

All the steps of generating the shape model have been described, the generation of the shape model can be summarized as follows:

1. Annotate a set of images (section 2.1)
2. Convert the annotations to sets of points
3. Calculate the mean shape
4. Subtract the mean shape from all shapes
5. Calculate the SVD of the resulting shapes
6. Keep only the largest few eigenvectors and eigenvalues

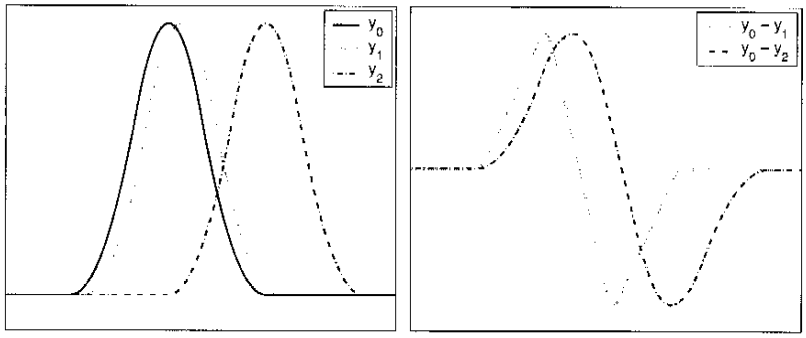
2.4 The texture model

To make a model of a set of images, the SVD is used in a similar way as for the shape model. Instead of coordinates of a shape, the pixel-values of each image are put into a vector.

Before the SVD is calculated, the images are aligned using the annotations. This is illustrated by figure 2.3. A pulse y_0 is plotted, together with y_1 and y_2, which are horizontally shifted copies of y_0. Figure 2.3(b) shows the differences between the shifted signals and the originals. Although the pulses all have the same shape and are equal up to a linear operation, their differences cannot be described by a single linear operation. This also occurs when the difference of an image and a shifted copy is calculated. A large set of eigenvectors would be necessary to represent a set of slightly shifted signals, whereas only one eigenvector would suffice if the signals were horizontally aligned. The images can be aligned using the annotations, by warping one image to match the other. This is further described in section 2.5.

After the images are aligned, the pixel intensities should be scaled to match each other. Without scaling, a brighter image, an image with greater pixel intensities, will have a larger influence on the mean texture and the model parameters than a less bright image. This can be seen from the example 2D data, which was shown in figure 2.2. Scaling different points with a different scaling factor will result in a different set of eigenvectors.

For the model fitting algorithm presented in the next chapter, the mean of the texture has to be known. It can be calculated right before the texture



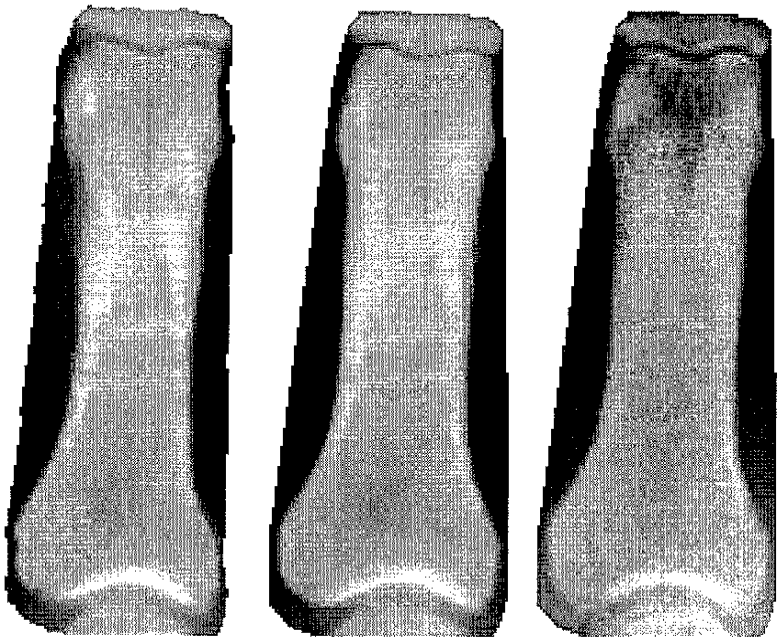
(a) Three pulses (b) The differences between the pulses

Figure 2.3: Effect of horizontal shift on the difference of two signals

parameters are calculated. So, after aligning the images, the texture model can be calculated in three steps. First, a mean image is calculated from the original images. Secondly, all images are scaled in brightness and contrast to fit the mean image as well as possible. The scaling parameters are determined by calculating the least square fit between the mean image and each original image. Finally, the mean of all the scaled vectors is calculated, and subtracted from all vectors. These vectors are then combined into a matrix, on which an SVD is performed.

Using the eigenvectors from the SVD, an image I, aligned to the mean texture, can be approximated using a linear combination of the mean texture T_0 and n eigenvectors $T_i ... T_n$ (equation 2.6). The values for λ are the texture parameters of the model. Figure 2.4(b) and 2.4(c) show how an image can change by texture variation.

$$I = \sum_{i=0}^{n} \lambda_i T_i \tag{2.6}$$



(a) shape variation (b) template image (c) texture variation

Figure 2.4: Shape and texture variation of the model

2.5 Warping

One part of the model is based on pixel values, the other part is based on shapes. The points from the shape are used to deform the texture model to match another image. This process of deforming an image is called warping. This section describes one warping method, the affine warp [1]. This warp is chosen because the partial derivative of the warp function is easily derived. Other warp methods can also be used, as long as the warp is invertible and the partial derivative can be computed. The warping method operates on coordinates. To calculate pixel intensities from these coordinates, filtering and interpolation are used.

2.5.1 Affine warping

The affine warp is applied to triangles. To divide an image into triangles, the points of the shape model are triangulated with the Delaunay algorithm [2]. The whole image is then warped triangle by triangle. The results can be summed to generate the complete image. An example is shown in figure 2.4(a). it was warped from figure 2.4(b). Figure 2.6 shows how the image is divided into triangles.

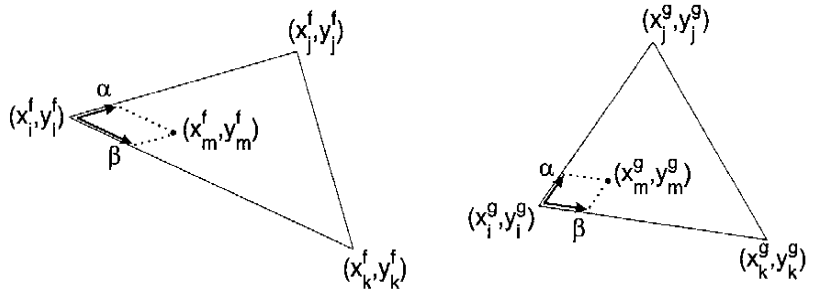


Figure 2.5: Affine warp from F to G

To warp a triangle, its coordinates must be known in the source image F and the target image G. Let $\mathbf{T_F}$ be a triangle in image F, and $\mathbf{T_G}$ the corresponding triangle in image G. Figure 2.5 shows the two triangles, and the names of the points. Each triangle has 3 points, i,j and k. Each point has an x and a y coordinate. So point i of triangle $\mathbf{T_F}$ is written as i^f. The coordinates of this point are written as (x_i^f, y_i^f). x_i^f is then the x coordinate of point i in image F. Using this notation, the triangles can be written as matrices.

$$\mathbf{T_F} = \begin{bmatrix} x_i^f & y_i^f & 1 \\ x_j^f & y_j^f & 1 \\ x_k^f & y_k^f & 1 \end{bmatrix}, \quad \mathbf{T_G} = \begin{bmatrix} x_i^g & y_i^g \\ x_j^g & y_j^g \\ x_k^g & y_k^g \end{bmatrix} \tag{2.7}$$

The vector with ones is concatenated to the coordinates of $\mathbf{T_F}$, this introduces a parameter for translation in the matrix $\mathbf{A}$ of the equation that describes the warp:

$$\mathbf{T_G} = \mathbf{T_F} \cdot \mathbf{A} \tag{2.8}$$

$\mathbf{A}$ is a 3×2 matrix with the warp coefficients. After this equation is used to determine the warp coefficients $\mathbf{A}$, warping a single point m from F to G can be done by multiplying the coordinates with the warp coefficients:

$$\mathrm{m_g} = \begin{bmatrix} x_m^f & y_m^f & 1 \end{bmatrix} \cdot \mathbf{A} \tag{2.9}$$

Again a constant should be included in the vector of coordinates to allow for translation in the warp.

2.5.2 The affine warp derivative

For the registration algorithm, it is necessary to compute the derivative of the warp function with respect to a single point. The input and output of the warp function are coordinates. The partial derivative of the warp function describes a change in coordinates for all pixels in the texture model as a function of a change in the coordinates of the shape model.

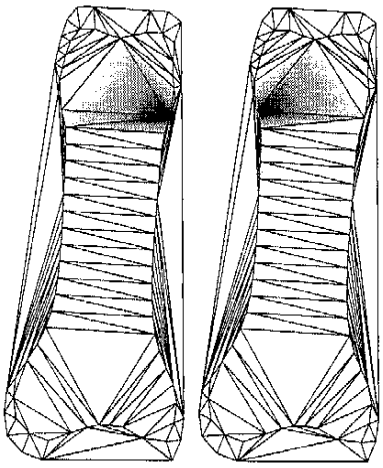
The derivative can be calculated more easily if the previous equations are rewritten. The point m can be expressed as a linear combination of the points of the triangle, both in F and in G.

$$\begin{aligned} x_m = \; & x_i + \alpha(x_j - x_i) + \beta(x_k - x_i) \\ y_m = \; & y_i + \alpha(y_j - y_i) + \beta(y_k - y_i) \end{aligned} \tag{2.10}$$

Since both the warp and equation 2.10 are linear, the order of operations does not matter. The symbols used in this equation are illustrated in figure 2.5, since the equation is valid for both F and G, the superscripts are dropped. To calculate the warp, the values for α and β are calculated in F. Using these values, the warp function W for a point m can be written as a linear combination of the elements of $\mathbf{T_G}$:

$$W(m) = i^g + \alpha(j^g - i^g) + \beta(k^g - i^g) \tag{2.11}$$

This warp function is only for one triangle, the total warp is made by summing the results of the warps for each triangle. The derivative of the warp function W with respect to i^f is then the sum of the derivatives for



(a) $\frac{\partial W(x)}{\partial x_2}$ (b) $\frac{\partial W(x)}{\partial x_{48}}$

Figure 2.6: Derivatives of the warp function

each triangle. Equation 2.11 can be used to calculate the derivative for one triangle:

$$\frac{\partial W(m)}{\partial x_i^g} = (1-\alpha-\beta, 0) \qquad \frac{\partial W(m)}{\partial y_i^g} = (0, 1-\alpha-\beta) \tag{2.12}$$

Note that from equation 2.12 it follows that $\frac{\partial W(x)}{\partial x_i} = \frac{\partial W(y)}{\partial y_i}$ and $\frac{\partial W(x)}{\partial y_i} = \frac{\partial W(y)}{\partial x_i} = 0$. The values for α and β can be calculated from $\mathbf{T_F}$ by solving equation 2.10, as shown in equation 2.13.

$$\begin{aligned} \alpha &= -(-y_i x_k + y_i x_m - x_m y_k + x_i y_k - x_i y_m + y_m x_k)/\text{denominator} \\ \beta &= (y_j x_i + y_i x_m - x_j y_i - y_j x_m - x_i y_m + x_j y_m)/\text{denominator} \\ \text{denominator} &= (x_j y_k - x_j y_i - x_i y_k - y_j x_k + y_j x_i + y_i x_k) \end{aligned} \tag{2.13}$$

Figure 2.6 shows the derivative of the warp function for two points. The triangulation of the point set is plotted on top for clarity. The partial derivative for point x_2 is shown on the left. Pixels near this point are colored dark, the partial derivative is close to one. The partial derivative decreases with the distance to x_2. A point only affects coordinates in vertices to which the point is connected. A movement of point x_2, for example, does not alter the warp for the lower half of the image.

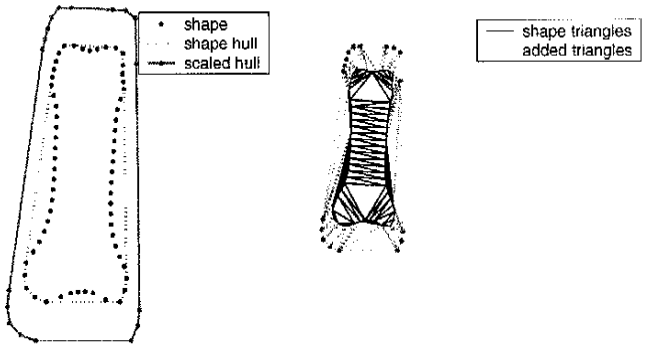
2.5.3 Warping outside the shape

The points of the shape model used in this thesis are located at the edge of the bone. By using the affine warping method, only points inside any of the triangles of the triangulation of the shape model are warped; the area inside the 'shape hull' of figure 2.7(a). Warping images with this shape will give results where the bone edge is barely visible. The bone edge is very useful for aligning images and should be included in the warped image.

The outer edge of an image which is deformed using affine warping is equal to the convex hull of the points of the shape model. To increase the area of pixels that is warped, the set of points used by the affine warp are altered so that its convex hull is larger.

To increase the warp area, the convex hull of the shape is calculated. The hull is then scaled, figure 2.7(a) gives an example. By adding the points of the scaled hull to the shape, the warp area can be extended. Figure 2.7(b) shows an example of the triangulation used for warping. The extra points can either be added to the shape model, or added just before warping. Section 5.1 will show that even though the coordinates of the extra points are based on the shape, adding these points to the shape model reduces the model accuracy greatly. To add the extra points to a target shape, the following algorithm is used:

1. Calculate the mean shape, without extra points
2. Calculate the indices K of the points part of the convex hull of the mean shape
3. Select the points with indices K from the target shape
4. Scale these points and add them to the target shape



(a) The convex hull (b) The extended shape

Figure 2.7: Extending the warp area

2.5.4 Implementation

By warping the image, the size of the image will change. To prevent anti-aliasing, the image should be low-pass filtered if the warp reduces the size of the image. The scaling can be calculated using the area of the convex hull of both shapes. The square root of the ratio of both areas is the scaling used for filtering.

Since no exact criteria for the filter are known, a standard filter is chosen. The filtering is done using the filter of MATLAB's *imresize* function. This function uses a Hanning filter with a cut-off frequency equal to the Nyquist frequency of the scaled image.

A filter width of 15 samples is used. As will be shown in chapter 5, a model with a texture resolution of 80 dpi can be used to fit the model to a test set which consist mainly of images with a resolution of 600 dpi, so the images are scaled down by a factor 7.5, in which case the hanning filter should be exactly of length 15. When the model is fitted, the image is warped multiple times. To reduce the computation time the image is filtered only once, using the initial scale estimate.

In the registration algorithm, an image is warped to match the template for each iteration. Since the template image is constant, many computations can be precomputed. For each pixel in the template image, it is known to which triangle it belongs. The triangle can be found by iterating over all triangles (for each pixel). The values for α and β can be calculated with the pixel coordinates. The pixel is part of the triangle if $\alpha \geq 0$, $\beta \geq 0$ and $\alpha + \beta \leq 1$. The triangle number for each pixel can be stored in memory. If F is the template image, and G the image to be warped, the warping consist of the following steps:

1. Calculate the warp coefficients $\mathbf{A}$ for each triangle, using equation 2.8
2. Lookup the triangle number for each pixel in F
3. Calculate the coordinates in G for each pixel of F, using equation 2.9
4. Interpolate the pixels, using the coordinates from the previous step

If the number of pixels is much larger than the number of triangles, step 3 and 4 require the most calculations. These steps can be performed quite fast on modern PC's. On an AMD Sempron running at 1.8 GHz, an image of 60.000 pixels can be warped 25 times per second. Optimizing the code for step 1 can reduce the computation time. The implementation as done for this thesis uses linear interpolation in step 4. Because the images are low-pass filtered before warping, more complex interpolation methods will not give much better results.

3

Model fitting

The previous chapter describes the model. The chapter showed how a model can be made using manual annotation. When the model is made, it should be possible to approximate other images than the ones used to generate the model. This chapter shows how the model parameters can be determined to approximate an image.

Since there is no closed form solution to this problem, an optimization algorithm is used. In the literature [5], a numerical approach is often used. The method presented here uses a Taylor approximation to estimate the gradient of an error function, similar to [13]. It is assumed that an initial estimate for the parameters is available, section 3.1 shows how an estimate can be given by the user.

Apart from the shape and texture parameters in sections 2.3 and 2.4, there are other parameters that have to be known to match the model to an image. In section 2.3 all shapes were aligned to one another using a global transformation, consisting of a scaling, a rotation and two translation parameters. In section 2.4 the brightness and contrast of the images was adjusted. Section 3.2 shows how these parameters can be included in the shape and the texture model. The parameters for the global transformation are modelled as shape parameters and the texture parameters are altered to include brightness and contrast.

Section 3.3 then shows how the shape and texture parameters can be optimized using the results from the previous section. Chapter 5 will show the results of a comparison of this method with the gradient method used in this thesis.

3.1 Initial estimate

The algorithm presented below needs an initial estimate of all the parameters. For the shape, the mean can be used as an estimate for the parameters. For the brightness, contrast and the global transformation, this will

not work.

The model fitting algorithm presented in this chapter will not converge when the initial estimate of the global transformation is not close to the optimal transformation. Solving the problem of finding the global transformation is beyond the scope of this report. Therefore, an estimate of the global transform is required from the user.

If the shape parameters are known, the texture parameters can be estimated. The image should be warped using the shape parameters, equation 2.6 then describes the relation between the image and the texture parameters. Using the pseudo-inverse $\mathbf{T}^+$, a least-square solution for the texture parameters can be calculated [21], as shown in equation 3.1.

$$\lambda = \mathbf{T}^+ \mathrm{I}_\mathrm{w} \tag{3.1}$$

The brightness and contrast can be estimated by adding two vectors to the matrix $\mathbf{T}$. For the contrast a vector with the mean texture should be added. To estimate the brightness a vector with all elements set to one should be added. By using the pseudo-inverse in equation 3.1, the least square solution for the texture parameter λ is found.

3.2 Combining parameters

When a new image is aligned, there will be a difference in translation, rotation and scaling. This section shows how a difference in these parameters can be combined into the shape model. The group of translation, rotation and scaling parameters will be referred to as pose. In Section 3.3 the model is fitted to an image, using the partial derivatives of the model parameters. Therefore, an expression of the shape as a function of the shape and pose parameters will be given. This expression is then used to calculate the partial derivatives necessary for section 3.3.

The pose consists of four parameters; the angle ϕ, the scale s and the translations in both dimensions Δx and Δy. The coordinates (x_p, y_p) for point (x, y) transformed with the pose parameters is given in equation 3.2.

$$\begin{aligned} x_p &= xs\cos(\phi) - ys\sin(\phi) + \Delta x \\ y_p &= xs\sin(\phi) + ys\cos(\phi) + \Delta y \end{aligned} \tag{3.2}$$

The gradient descent algorithm assumes the updates are small, so $x_p \approx x$. Because the origin is the center of rotation, a small angle can lead to large changes in coordinates if the shape is located far away from the origin. The rotation is therefore performed around the mean (c_x, c_y) of the coordinates of all pixels of the mean texture. This can be done by using equation 3.3 instead of equation 3.2.

$$\begin{aligned} x_p &= (x - c_x)s\cos(\phi) - (y - c_y)s\sin(\phi) + \Delta x + c_x \\ y_p &= (x - c_x)s\sin(\phi) + (y - c_y)s\cos(\phi) + \Delta y + c_y \end{aligned} \tag{3.3}$$

Using $c = (x, y)$ and $q = (\phi, s, \Delta x, \Delta y)$, the partial derivatives for equation 3.3 can be calculated. The partial derivative is dependent on ϕ and s. This is solved by using $x_p \approx x$, which means that $\phi \approx 0$ and $s \approx 1$.

$$\frac{\partial c}{\partial q} = \begin{bmatrix} -(y - c_y) & x - c_x & 1 & 0 \\ x - c_x & y - c_y & 0 & 1 \end{bmatrix} \tag{3.4}$$

This equation holds for all points c_i in the shape c. Next, the partial derivatives of the shape model are calculated. The shape is calculated by adding the eigenvectors $\mathrm{U}(i)$ to the mean shape c_0, using the shape parameters h.

$$\mathrm{c} = \mathrm{c}_0 + \mathbf{U}\mathrm{h} = \mathrm{c}_0 + \sum_{i=1}^{k} \mathrm{U}(i)\ h_i \tag{3.5}$$

The eigenvectors in $\mathbf{U}$ contain both the x and y coordinates of a point. If the eigenvector $\mathrm{U}(i)$ is split into parts for each axis, $\mathrm{U}(i) = [\mathrm{U_x}(i),\ \mathrm{U_y}(i)]$, the partial derivatives are:

$$\frac{\partial c}{\partial h} = \begin{bmatrix} \mathrm{U_x}(1) & \mathrm{U_x}(2) & \ldots & \mathrm{U_x}(k) \\ \mathrm{U_y}(1) & \mathrm{U_y}(2) & \ldots & \mathrm{U_y}(k) \end{bmatrix} \tag{3.6}$$

The pose and shape parameters are combined by concatenating the matrices:

$$\frac{\partial c}{\partial p} = \left[\frac{\partial c}{\partial q}\ ,\ \frac{\partial c}{\partial h}\right] \tag{3.7}$$

The texture model can be altered to include contrast and brightness. The texture model from section 2.4 does not describe these variations. The contrast variation can be added by adding the mean image T_0 as an eigenvector to the texture model. Brightness variation can be added by adding an eigenvector with all elements set to one.

3.3 Simultaneous optimization

With the initial estimate from section 3.1 it is possible to optimize all model parameters using an optimization algorithm. Since the model is a combination of various parts, a lot of symbols are used to describe various parameters. Throughout this chapter, the variables will be named according to the nomenclature.

3.3.1 The criteron

The goal of the optimization algorithm is to minimize the quadratic error by adapting the model parameters. Image I is the image to which the model is matched. It is warped to the mean texture T_0. The first set of parameters controls the shape variation, given by p. An image warped with parameters p is then written as:

$$I_w = I(W(p)) \tag{3.8}$$

After the image is warped, the contrast and brightness of the image are matched to the template:

$$I_s = \alpha I_w + \beta \tag{3.9}$$

Then, the texture variation is applied:

$$I_t = I_s + \sum_{i=1}^{s} \lambda_i T_i \tag{3.10}$$

Combining these steps into one equation gives a function for the image I, altered to match the template image.

$$I_t = \alpha I(W(p)) + \beta + \sum_{i=1}^{s} \lambda_i T_i \tag{3.11}$$

The error function e in equation 3.12 gives the quadratic error between the image and the template. It is the sum of the quadratic pixel error over all coordinates c. The square in this equation is an element-wise operator.

$$e = \sum_{x=1}^{N} \left[T_0 - \left(\alpha I(W(p)) + \beta + \sum_{i=1}^{s} \lambda_i T_i \right) \right]^2 \tag{3.12}$$

3.3.2 Efficient implementation

The criterion in equation 3.12 can be solved by using a Taylor expansion. This idea was first presented in [10] and applied to shape models in [13]. If equation 3.12 is used directly for Taylor expansion, the partial derivatives in the solution will depend on the current estimates of the parameters. The partial derivatives are then calculated using the pixel values from the image. This section shows a variation on this method where the partial derivatives are calculated using pixel values from the template image. This has two advantages: The partial derivatives only have to be calculated once and they are based on the template image. Since the template image is generated by averaging many radiographs, the noise in the template image is much lower, and the derivative of the pixel values is more accurate.

The parameter optimization is done in three steps. The first step is to apply the current parameter estimates to the image I, so it can be warped

to match the template T_0. This is done in equation 3.11. The next step is to calculate the parameter update relative to T_0 instead of the warped image I_t. Since T_0 is constant, a number of intermediate results can be pre-computed. The last step is to estimate how to update the parameters for I, given the estimated parameter update for T_0.

To calculate the parameter updates, the update to T_0 is calculated. Equation 3.12 is rewritten and two parameters to describe the update are added. These parameters, $\Delta\lambda$ and Δp, are the updates to the current parameter estimates. The summation is also dropped, so the error e is a vector. By minimizing the error of the elements of e the total error $\sum e$ is also minimized. To derive an update for the model parameters based on the error, it is not possible to use the scalar $\sum e$ directly, the different model parameters do have a different effect on the vector e, but not necessarily on the scalar $\sum e$.

$$e = \left[I_t - \sum_{i=0}^{s} (\lambda_0 + \Delta\lambda_i) T_i(W(p_0 + \Delta p)) \right]^2 \tag{3.13}$$

The vectors λ_0 and p_0 are chosen such that the error e is zero when I_t is equal to the template image T_0. All elements of these vectors are equal to zero, except for the first element of λ_0 and the second of p_0, which control the contrast of the image and the scaling of the shape. These parameters are equal to one.

By solving this equation for $\Delta\lambda$ and Δp, a change in parameters can be calculated which decreases the quadratic error. This is done in a few steps. First, a first order Taylor expansion of the warp function is performed.

$$e = \left[I_t - \sum_{i=0}^{s} (\lambda_{0i} + \Delta\lambda_i) T_i(W_0 + \frac{\partial W}{\partial p}\Delta p) \right]^2 \tag{3.14}$$

Next, a first order Taylor expansion to the template image is done.

$$e = \left[I_t - \sum_{i=0}^{s} (\lambda_{0i} + \Delta\lambda_i) \Big(T_i + \frac{\partial T_i}{\partial W}\frac{\partial W}{\partial p}\Delta p \Big) \right]^2 \tag{3.15}$$

The values for λ_0 are known, only the first element is non-zero. These values can be substituted into the equation:

$$e = \left[I_t - (T_0 + \frac{\partial T_0}{\partial p}\Delta p) - \sum_{i=0}^{s} \Delta\lambda_i \Big(T_i + \frac{\partial T_i}{\partial W}\frac{\partial W}{\partial p}\Delta p \Big) \right]^2 \tag{3.16}$$

By expanding this equation, the texture model T can be separated from the partial derivatives.

$$e = \left[I_t - T_0 - \frac{\partial T_0}{\partial p}\Delta p - T\Delta\lambda - \sum_{i=0}^{s} \Big(\Delta\lambda_i \frac{\partial T_i}{\partial W}\frac{\partial W}{\partial p}\Delta p \Big) \right]^2 \tag{3.17}$$

This equation should eventually be solved for both Δp and $\Delta\lambda$. It is therefore easier to rewrite the equation by grouping these two vectors.

$$q = \begin{pmatrix} \Delta p \\ \Delta\lambda \end{pmatrix}, \quad T_q = \begin{pmatrix} \frac{\partial T_0}{\partial p} & \mathbf{0} \\ \mathbf{0} & T \end{pmatrix} \tag{3.18}$$

A direct substitution results in a quadratic matrix equation. To solve the equation analytically, this second order term is assumed to be zero. Section 5.3 will show that this is a valid approximation.

$$e = [I_t - T_0 - T_q\ q]^2 \tag{3.19}$$

This equation can be solved by setting the partial derivative to zero.

$$\frac{\partial e}{\partial q} = -2T_q^T[I_t - T_0 - T_q q] = 0 \tag{3.20}$$

$$T_q{}^T T_q q = T_q{}^T[T_0 - I_t] \tag{3.21}$$

Since T_q is constant, the equation can be simplified by introducing a matrix $\mathbf{R}$, using $\mathbf{R} = (T_q{}^T T_q)^{-1} T_q{}^T$. Equation 3.22 shows how calculate the update q to the model parameters, using $\mathbf{R}$. Updating the parameters using this equation is equal to the update of the Gauss-Newton optimization method. Other research has shown that the Gauss-Newton method gives better results than most other optimization methods [1].

Since the error function is non-linear and the update is calculated using a linear approximation, the update will sometimes be too large, and sometimes too small. When the update is too large, the algorithm can 'bounce' around the optimal solution, without ever reaching it. The update is therefore multiplied by a step size. The step size μ is chosen as 0.5 for this thesis, this is small enough to prevent the 'bouncing' in nearly all occasions, while it is still large enough for the algorithm to converge quickly.

$$\mathrm{q} = \mu\mathbf{R} \cdot \mathrm{E}, \quad \mathrm{E} = (\mathrm{T_0} - \mathrm{I_t}) \tag{3.22}$$

The matrix $\mathbf{R}$ from this chapter is of size $(k + s) \times N$ for a model of k shape parameters and s texture model parameters. This matrix can be calculated once per model, since it is only dependent on the model. not on the image I. The partial derivative $\frac{\partial T_0}{\partial p}$ can be calculated in a few steps:

$$\frac{\partial T_0}{\partial p} = \frac{\partial T_0}{\partial W}\frac{\partial W}{\partial c}\frac{\partial c}{\partial p} \tag{3.23}$$

The partial derivative $\frac{\partial T_0}{\partial W}$ can be calculated with the gradient of the template image $\mathrm{T_0}$. Section 2.5.2 describes how to compute $\frac{\partial W}{\partial c}$. The matrix $\frac{\partial c}{\partial p}$ is covered in section 3.2.

3.3.3 Updating parameters

The previous section showed how to calculate a parameter update for the template image. However, the template image is constant, so the parameter update should be applied to the image to which the model is fitted. This should be done in reverse order. First the image is warped, and then the texture model is applied. This is similar to equation 3.12, only the α and β are part of the texture model, as in equation 3.13. So, the update is calculated with respect to T_0:

$$e = \left[\mathrm{I_t} - \sum_{i=0}^{s} (\lambda_{0\mathrm{i}} + \Delta\lambda_{\mathrm{i}}) T_i(W(\mathrm{p_0} + \Delta\mathrm{p})) \right]^2 \tag{3.24}$$

It is rewritten to fit equation 3.12, the updates $\Delta\lambda$ and Δp are applied to λ and p.

$$e = \left[\mathrm{T_0} - \left(\alpha I(W(\mathrm{p})) + \beta + \sum_{i=1}^{s} \lambda_i \mathrm{T}_i \right) \right]^2 \tag{3.25}$$

The next section shows how the warp parameters can be rewritten. After that, section 3.3.3 shows how the texture parameters can be rewritten.

Updating the shape parameters

An exact solution to calculate the new shape parameters from the updates calculated in section 3.3.2 does not exist. The parameter update for the shape can be calculated using a linear approximation to the update, as shown next. This linear approximation gives an error in the model parameters. The second part of this section shows a method that minimizes this error.

The updates for the shape and pose parameters in section 3.3.2 are updates to the shape of the template image, which should be rewritten as updates to the shape of the image to which the model is fitted. A first step is to calculate the updates to the shape of the warped image. This is done by multiplying the updates by -1. Using the updates for the warped image, the new parameters for the shape of the image can be calculated. The next step is to rewrite the updates to the shape of the warped image as updates to the shape of the image to which the model is fitted. The shape of an image is given by equation 3.26, which is a combination of equations 3.2 and 3.5. The updates calculated by the model fitting algorithm are updates to the mean shape c_0 and the shape parameters p_s. For the next iteration in the fitting algorithm, the updates have to be rewritten as updates to the shape of the image to which the model is fitted. So updates to c_0 and p_s should be rewritten as updates to p_s and the pose parameters (s,ϕ and Δc).

$$c = sR(c_0 + Up_s) + \Delta c, \quad R = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix}, \quad \Delta c = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{3.26}$$

To simplify the following formulas, a rotation by q_ϕ is written as a matrix multiplication by R_q, using equation 3.27. An update to the scale and rotation of c_0 cannot be exactly rewritten as an update to the shape and pose parameters. It is therefore necessary to make a few approximations. Equation 3.28 shows how a change in scale (q_s) of the mean shape c_0 can be approximated by a change in one of the pose parameters (s). Equation 3.29 shows how a rotation of c_0 by q_ϕ can be approximated by a change in one of the pose parameters (ϕ). A translation of the mean shape c_0 can be written as a change in Δc without any approximations, as is shown in 3.30.

$$R_q = \begin{bmatrix} \cos(q_\phi) & -\sin(q_\phi) \\ \sin(q_\phi) & \cos(q_\phi) \end{bmatrix} \tag{3.27}$$

$$c = sR[(1 + q_s)c_0 + Up] + \Delta c \approx (s(1 + q_s))R[c_0 + Up] + \Delta c \tag{3.28}$$

$$c = sR[R_q c_0 + Up] + \Delta c \approx s(RR_q)[c_0 + Up] + \Delta c \tag{3.29}$$

$$c = sR[c_0 + Up + q_{\Delta c}] + \Delta c = sR[c_0 + Up] + \Delta c + s\ R\ q_{\Delta c} \tag{3.30}$$

The parameter update is calculated by first multiplying the updates by -1, to get the update to the warped image. Next, equations 3.28, 3.29 and 3.30 are applied, to get the update to the shape of the image. Equation 3.31 shows how these two steps are combined to calculate the new parameters for the shape of the image (s^*, ϕ^* and Δc^*), based on the current parameters and the updates to the mean shape.

$$\begin{aligned} s^* &= s(1 - q_s) = s - sq_s \\ R^* &= RR_q, \quad \phi^* = \phi - q_\phi \\ \Delta c^* &= \Delta c - s\ R\ q_{\Delta c} \\ p^* &= p - \Delta p \end{aligned} \tag{3.31}$$

In equations 3.28 and 3.29 approximations were made. Next, a method is shown which minimizes the error in these approximations. Using the equations from the beginning of section 3.3.3, and ignoring texture variation, the following warps match the two images:

$$I(W(\mathrm{p})) \approx T_0(W(\Delta \mathrm{p})) \tag{3.32}$$

The image I is warped with the current parameter estimate. The template is warped with Δp to compensate for the errors in the current parameter estimate p.

Throughout the rest of this section the set of coordinates c, as defined in equation 5, will have the name of the corresponding warp parameters as subscript. The mean shape of the model is denoted as c_0. The warp function is defined slightly different than in the rest of the report. In the rest of the report the warp function is used to calculate the coordinates of pixels from the template T_0, here it is used to calculate coordinates of the shape model.

A warp $c_p = W(c, p)$ warps the coordinates of c from the model coordinates, shown in figure 3.1, to the image coordinates c_w, using the shape parameters p. The point-set resulting from the current shape parameters p is c_p. The point-set resulting from the parameter update applied to the mean shape is $c_{\Delta p}$.

$$\mathrm{c_p} = W(\mathrm{c_0}, \mathrm{p}), \quad \mathrm{c_{\Delta p}} = W(\mathrm{c_0}, \Delta \mathrm{p}) \tag{3.33}$$

In one iteration of the fitting algorithm, an image I is warped with the current parameters p. The update Δp is calculated which should be applied to the template to align both images. The problem is then to find the shape parameters p_n, that match $c_{\Delta p}$ to c_p.

$$c_p \approx W(c_{\Delta p}, p_n) \tag{3.34}$$

Figure 3.1 shows an example. For each warp, three points are shown. The two point-sets on the left are based on the image, the two on the right are based on the model. All point-sets except c_{pn} are known.

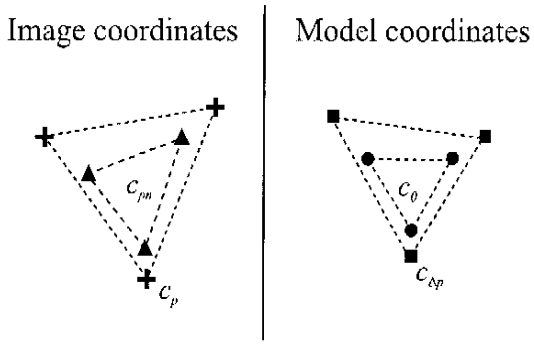


Figure 3.1: Overview of the different point-sets needed for updating the parameters

As mentioned before, the order of rotating and scaling the mean shape and adding the shape eigenvectors to this shape is reversed, so an exact solution for p_n does not always exist. Therefore the least square error of the approximation to p_n is minimized. The optimization starts with an initial estimate of p_n. The estimate is made using the linear approximation of equation 3.31. This introduces a small error dp. Using p_n it is possible to

calculate dp. Equation 3.35 shows the relation between these parameters, $(p_n + dp)$ is the set of parameters that matches $c_{\Delta p}$ to c_p.

$$c_p \approx W(c_{\Delta p}, p_n + dp) \tag{3.35}$$

dp is calculated by optimizing a square error e, based on the difference between the left and right hand side of equation 3.35:

$$e = [W(c_{\Delta p}, p_n + dp) - c_p]^2 \tag{3.36}$$

The square in this equation is an element-wise operator, just as the error optimized in section 3.13. To solve equation 3.36 a linear approximation of the warp function is made.

$$e = \left[W(c_{\Delta p}, p_n) + \frac{\partial W(c_{\Delta p}, p_n)}{\partial p} dp - c_p\right]^2 \tag{3.37}$$

The derivative of the warp function is calculated just as for section 3.3.2, $\frac{\partial W}{\partial p} = \frac{\partial W}{\partial c}\frac{\partial c}{\partial p}$. $\frac{\partial W}{\partial p}$ is calculated using $\frac{\partial W}{\partial c}$ from section 2.5.2 and $\frac{\partial c}{\partial p}$ from section 3.2. In section 3.3.2 the partial derivative $\frac{\partial W}{\partial p}$ is evaluated for the coordinates of a set of pixels. In this section, it should be evaluated for the coordinates $c_{\Delta p}$. For some shapes, it can occur that coordinates of $c_{\Delta p}$ are outside the convex hull of the mean shape c_0, which means that the warp function and it's derivative are not defined for that point. To calculate the partial derivative for these points, the coordinates of the vertex closest to this point is used to determine the partial derivative (using equation 2.12). The minimal error of equation 3.37 can be found by setting the derivative to zero.

$$\frac{\partial e}{\partial dp} = 2\frac{\partial W}{\partial p}^T \left[W(c_{\Delta p}, p_n) + \frac{\partial W}{\partial p} dp - c_p\right] = 0 \tag{3.38}$$

Using a few intermediate steps, the equation can be solved for dp:

$$\frac{\partial W}{\partial p}^T \frac{\partial W}{\partial p} dp = \frac{\partial W}{\partial p}^T [c_p - W(c_{\Delta p}, p_n)] \tag{3.39}$$

$$dp = \left[\frac{\partial W}{\partial p}^T \frac{\partial W}{\partial p}\right]^{-1} \frac{\partial W}{\partial p}^T [c_p - W(c_{\Delta p}, p_n)] \tag{3.40}$$

$$dp = \frac{\partial W}{\partial p}^{-1} [c_p - W(c_{\Delta p}, p_n)] \tag{3.41}$$

This gives the new estimate for p_n; $\tilde{p}_n = p_n + \mu dp$. Since a linear approximation was used for the calculation of dp, this new result is not exact, just as for the model fitting algorithm, a step size μ is introduced. μ is set to 0.5. $\tilde{p}_n$ should be better than the previous estimate p_n, so it can be used again to estimate a new value for dp. In a few iterations of updating dp and

p_n, this should give a better approximation. The value of the error, $\sum e$, can be checked to see if this gives indeed an improvement.

The partial derivatives are calculated using equations 2.12 and 2.13, which are only dependent on the values of $c_{\Delta p}$, not on c_{pn}. Thus, they are constant throughout the iterations and have to be computed only once. These derivatives were also used in section 3.3.2 and can be calculated in the same way.

The iterations will be very fast compared to most other steps in the complete algorithm. The implementation for this thesis updates p_n 15 times using equation 3.41. The result from the iteration with the lowest error is used as the final result. When the algorithm does not converge, the initial estimate will give the lowest error, so the solution is always at least as good as the initial linear approximation.

Updating the texture parameters

To calculate the update of the texture parameters, the warp is removed from the equation. By using $\tilde{\lambda} = \lambda_0 + \Delta\lambda$, the error function of equation 3.13 can be written as:

$$e = \left[I_t - \sum_{i=0} \tilde{\lambda} T_i \right]^2 \tag{3.42}$$

The square is an element-wise operator, just as in the previous section. Just as for the shape parameter update, this function has to be rewritten to equation 3.12. This is done in a few steps, the first is to expand a part of the summation.

$$e = \left[I_t - \tilde{\lambda}_0 T_0 - \tilde{\lambda}_1 T_1 - \sum_{i=2}^{s} \tilde{\lambda} T_i \right]^2 \tag{3.43}$$

The eigenvectors T_0 and T_1 are the eigenvectors added in section 3.2 to add contrast and brightness variation to the image model. The image I_t is made with the current best estimate of the texture parameters:

$$I_t = \alpha I + \beta + \sum_{i=2}^{s} \lambda_i T_i \tag{3.44}$$

Substituting this equation into equation 3.43 gives:

$$e = (-\tilde{\lambda}_0)^2 \left[T_0 + \frac{-1}{\tilde{\lambda}_0} (\alpha I + \beta + \sum_{i=2}^{s} \lambda_i T_i - \tilde{\lambda}_1 T_1 - \sum_{i=2}^{s} \tilde{\lambda}_i T_i) \right]^2 \tag{3.45}$$

By using the following substitutions, it is possible to rewrite this further.

$$\bar{\alpha} = \frac{\alpha}{\tilde{\lambda}_0}, \quad \bar{\beta} = \frac{\beta - \tilde{\lambda}_1 T_1}{\tilde{\lambda}_0}, \quad \bar{\lambda}_i = \frac{\lambda_i - \tilde{\lambda}_i}{\tilde{\lambda}_0} \tag{3.46}$$

All elements of T_1 have the same value, so $\bar{\beta}$ can be calculated by replacing T_1 with one of it's elements.

$$e = (\tilde{\lambda}_0)^2 \left[T_0 - \bar{\alpha} I - \bar{\beta} - \sum_{i=2}^{s} \bar{\lambda}_i T_i \right]^2 \tag{3.47}$$

This function is now very similar to the criterion used to fit the model. The criterion is given in 3.12. Without the warp function, it simplifies to:

$$e = \left[T_0 - \left(\bar{\alpha} I + \bar{\beta} + \sum_{i=2}^{s} \bar{\lambda}_i T_i \right) \right]^2 \tag{3.48}$$

This only differs from the previous equation by a factor. This means that both equations have their minimum at the same parameter values.

The warped image I_w is updated by combining all the separate substitutions. The update is given by q, in equation 3.22. Using the substitution from equation 3.18, $\Delta\lambda$ is determined. Finally, with $\tilde{\lambda} = \lambda_0 + \Delta\lambda$ and equation 3.46 the new texture parameters are given.

3.3.4 Summary of the parameter update

Since the parameter calculation involves a lot of steps, a short overview of all the steps is given in table 3.1. This table lists the steps of one iteration of the model fitting. The model fitting is started as described in section 3.1. After each iteration the new model parameters are used as the model parameters for the next iteration.

	Description	**Equation number**
1.	Apply the shape parameters to the shape model	3
2.	Apply the pose parameters to this shape	3.3
3.	Warp the image using this shape, and apply the texture parameters	3.11
4.	Calculate the update q to the template T_0	3.22
5.	Split q into Δp and $\Delta\lambda$	3.18
6.	Calculate the new shape parameters p	3.41
7.	Calculate $\tilde{\lambda} = \lambda + \Delta\lambda$	
8.	Calculate the new texture parameters	3.46

Table 3.1: summary of the model fitting algorithm

3.4 Active appearance models

In [5] another method for determining the update matrix R is described. In the sections above, an approximation was made analytically. In [5] a

numerical approximation is used. This section will give an overview of the numerical method, section 5.6 will compare the performance of both update methods.

Using the shape model, small variations to the mean texture are made. The shape parameters and the global transformation are randomly varied in a certain range. The resulting images are stored as vectors in a matrix V. The variations applied to the shape model are stored as vectors in a matrix C. To estimate parameters from an image, a matrix R is determined, such that $C = RV$. As shown in [4], this can be done in two steps:

$$\frac{dr}{dp} = VC^{+}, \quad R = \frac{dr}{dp}^{+} \tag{3.49}$$

A direct calculation, $R = CV^{+}$, is also possible, but according to [4] it does not give good results. The matrix R can be used exactly the same as R from equation 3.22.

The range over which the model is varied, is an important set of parameters is . With a large range, the algorithm should be less sensitive to the initial estimate, but the accuracy of the final solution could decrease. The sensitivity to the initial model can also be decreased by a multi-resolution approach, the chosen range of the parameter variation used in this thesis is therefore relatively small.

3.5 Parameter estimation

As shown in the previous section, the model is fitted to an image by adjusting the model parameters. The method adjusts these parameters based on the pixel values of the image. The better the model matches the image, the more accurate these adjustments are estimated. In cases where an image does not match the model very well, parameters can keep increasing in the model fitting algorithm, even though the error criterion does not improve.

This section presents a method to estimate the uncertainty of the model parameters. In [23] this is done using only a few pixels per point in the shape model. The method presented in this section uses all pixels of the warped image.

When the model is generated, the mean values and the covariances for the parameters can be calculated. By using this information in the model-fitting algorithm, it should be possible to limit parameters values to realistic values, while larger parameter values are still possible if this leads to a good match of the model.

3.5.1 Estimating image noise

The model parameters are determined using an optimization algorithm. The difference between a template image T_0 and a warped image $\mathrm{I_t}$ is used to determine the update to the model parameters. These model parameters can be estimated by using equation 3.22 from section 3.3.2:

$$\mathrm{q} = \mathbf{R} \cdot \mathrm{E}, \quad \mathrm{E} = (T_0 - I_t) \tag{3.50}$$

The vector E is the difference between the template image $\mathrm{T_0}$ and the warped image $\mathrm{I_t}$. It is assumed that this difference is only caused by noise and an error in the model parameters. Since the template is generated using a large number of images, it's noise is negligible compared to the noise in the warped image. E can then be written as:

$$\mathrm{E} = \mathrm{T_0} - (\mathrm{I_t} + \mathrm{N_t}) \tag{3.51}$$

Where $\mathrm{N_t}$ is the noise of the warped image. The noise can be split up into two parts, a part that has influence on the estimated model parameters, and a part that has no influence.

$$\mathrm{N_t} = \mathrm{q_n}\mathbf{R} + \mathrm{R_t} \tag{3.52}$$

If $\mathrm{R_t}$ has no influence on the estimated parameters, it is orthogonal to the parameter update matrix $\mathbf{R}$. $\mathrm{R_t}$ can thus be calculated using E and an orthonormal base $\mathbf{V}$ of $\mathbf{R}$. The unbiased estimate of the variance is then given by equation 3.54.

$$\mathrm{R_q} = \mathbf{V}\mathrm{E}, \quad \mathrm{R_t} = \mathrm{E} - \mathbf{V}^T\mathbf{R_q} \tag{3.53}$$

$$\mathbf{C}_t = \frac{\mathbf{R_t}^T \cdot \mathbf{R_t}}{N-1} \tag{3.54}$$

If the amount of noise varies spatially, the variance can be estimated in small blocks of pixels. The block size should be small enough to accurately capture the variation in noise. However, if the block size is chosen to small the noise estimate will be inaccurate, since it is based on only a few samples.

To estimate the noise in the image, it is assumed that the noise is independent for all pixels. This assumption is necessary to estimate the parameter variance, the estimate is based on only one vector, so covariances cannot be estimated. Within one block of pixels, the noise is assumed to be equally large for all pixels. If these assumptions hold, the noise in $\mathrm{N_t}$ is equally distributed over both subspaces and it's variance is equal to $\mathbf{C}_t$ scaled by $\frac{N-1}{N-k-1}$. The variance of E is equal to the variance of $\mathrm{N_t}$. If the noise is considered independent for all pixels, the covariance matrix for E is a diagonal matrix with the estimated variances on the diagonal. Using the covariance matrix $\mathbf{C_E}$ of E and equation 3.50, the covariance matrix $\mathbf{C_q}$ for the parameter update q can be calculated.

$$\mathbf{C_q} = \mathbf{R} \cdot \mathbf{C_E} \cdot \mathbf{R}^{\mathrm{T}} \tag{3.55}$$

3.5.2 Updating the model parameter covariance

The previous section showed how to estimate the uncertainty of the parameter update. The parameter updates and the covariances are calculated in the coordinate space of the template image. The model parameters, however, are estimated in the coordinate space of the image. The parameter updates for the model were calculated in section 3.3.3. This section will show how the variance of these model parameters can be estimated.

Linear shape update

In section 3.3.3 a linear approximation for the parameter update is given. The parameter update was calculated using equation 3.31:

$$\begin{aligned} s^* &= s(1-q_s) = s - sq_s \\ R^* &= RR_q, \quad \phi^* = \phi - q_\phi \\ \begin{pmatrix} \Delta x^* \\ \Delta y^* \end{pmatrix} &= \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} - s\, R \begin{pmatrix} q_{\Delta x} \\ q_{\Delta y} \end{pmatrix} \\ p^* &= p - \Delta p \end{aligned} \tag{3.56}$$

In order to write the parameter updates as a linear matrix equation, the model parameters of this equation are combined into a vector p. To update the shape parameters, there are three vectors with shape parameters. the current parameters are stored in p, the update to the template image in Δp

and the new shape parameters are given by p^*. These are listed in equation 3.57.

$$\begin{aligned} p^* &= [\phi^*, s^*, \Delta x^*, \Delta y^*, {p_{s1}}^*, \ldots, {p_{sk}}^*] \\ p &= [\phi, s, \Delta x, \Delta y, p_{s1}, \ldots, p_{sk}] \\ \Delta p &= [q_\phi, q_s, q_{\Delta x}, q_{\Delta y}, \Delta p_{s1}, \ldots, \Delta p_{sk}] \end{aligned} \tag{3.57}$$

To calculate the covariance of the parameter updates, these equations should be linearized, to fit equation 3.58.

$$p^* = \frac{\partial p^*}{\partial p} p + \frac{\partial p^*}{\partial \Delta p} \Delta p + P_0 \tag{3.58}$$

Using the equations 3.31 and 3.57, the solution to equation 3.58 is given by equation 3.59.

$$P_0 = \begin{bmatrix} 0 & sq_s & 0 & \ldots & 0 \end{bmatrix}, \quad \frac{\partial p^*}{\partial p} = \begin{bmatrix} 1 & & & & \\ & (1-q_s) & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix},$$

$$\frac{\partial p^*}{\partial \Delta p} = \begin{bmatrix} -1 & 0 & & \ldots & & 0 \\ 0 & -s & & & & \\ & & -s\cos(\phi) & +s\sin(\phi) & & \\ \vdots & & -s\sin(\phi) & -s\cos(\phi) & & \\ & & & & -1 & \\ & & & & & \ddots \\ 0 & & & & & -1 \end{bmatrix} \tag{3.59}$$

Linear texture update

To estimate the uncertainty for the texture parameters, equation 3.46 is linearized, as shown in equation 3.60.

$${\lambda_i}^* = \frac{\lambda_i - \tilde{\lambda}_i}{\tilde{\lambda}_0} \approx c_1 \lambda_i + c_2 \tilde{\lambda}_i + c_{3i} \tilde{\lambda}_0 + c_{0i} \tag{3.60}$$

The constants $c_0 \ldots c_3$ are equal to the partial derivatives, shown in equation 3.61.

$$
\begin{aligned}
c_1 &= \frac{\partial \lambda_i^*}{\partial \lambda_i} = \frac{1}{\tilde{\lambda}_0} \\
c_2 &= \frac{\partial \lambda_i^*}{\partial \tilde{\lambda}_i} = -\frac{1}{\tilde{\lambda}_0} \\
c_{3i} &= \frac{\partial \lambda_i^*}{\partial \tilde{\lambda}_0} = -\frac{\lambda_i - \tilde{\lambda}_i}{\tilde{\lambda}_0^2} \\
c_{0i} &= \frac{\lambda_i - \tilde{\lambda}_i}{\tilde{\lambda}_0} - c_1\lambda_i - c_2\tilde{\lambda}_i - c_3\tilde{\lambda}_0
\end{aligned}
\tag{3.61}
$$

The update equation for the contrast parameter α is different from the others, the values for c can be calculated by setting $\tilde{\lambda}_i$ to zero.

$$
\begin{aligned}
c_{20} &= \frac{\alpha}{\tilde{\lambda}_0}\frac{\partial}{\partial \tilde{\lambda}_i} = 0 \\
c_{30} &= \frac{\alpha}{\tilde{\lambda}_0}\frac{\partial}{\partial \tilde{\lambda}_0} = \frac{-\alpha}{\lambda_0^2}
\end{aligned}
\tag{3.62}
$$

The update for the texture parameters can be written as one equation, by combining the results from equation 3.61 and 3.62 into a matrix equation.

$$
\lambda^* = K_0 + K_1\lambda + K_2\tilde{\lambda} \tag{3.63}
$$

$$
K_0 = \begin{bmatrix} c_{01} \\ \vdots \\ c_{0k} \end{bmatrix}, \quad K_1 = \begin{bmatrix} c_1 & & 0 \\ & \ddots & \\ 0 & & c_1 \end{bmatrix}, \quad K_2 = \begin{bmatrix} c_{30} & & 0 \\ c_{31} & c_2 & \\ \vdots & & \ddots \\ c_{3k} & & c_2 \end{bmatrix} \tag{3.64}
$$

With the linearized updates for the shape and texture parameters, the updated parameters p^* and λ^* can be written as:

$$
\begin{bmatrix} p^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} P_0 & \mathbf{0} \\ \mathbf{0} & K_0 \end{bmatrix} + \begin{bmatrix} \frac{\partial p^*}{\partial p} & \mathbf{0} \\ \mathbf{0} & K_1 \end{bmatrix} \begin{bmatrix} p \\ \lambda \end{bmatrix} + \begin{bmatrix} \frac{\partial p^*}{\partial \Delta p} & \mathbf{0} \\ \mathbf{0} & K_2 \end{bmatrix} \begin{bmatrix} \Delta p \\ \tilde{\lambda} \end{bmatrix} \tag{3.65}
$$

Now the parameter update is written as a linear equation, the covariance of the updated parameters can be calculated. Any error in the current parameters should be compensated for by the parameter update. The covariance of the updated parameters is therefore only dependent on the covariance of the updates Δp and $\tilde{\lambda}$. The covariance matrix $\mathbf{C}$ for the updated parameters is then calculated with equation 3.66.

$$
\mathbf{C} = \begin{bmatrix} \frac{\partial p^*}{\partial \Delta p} & \mathbf{0} \\ \mathbf{0} & K_2 \end{bmatrix} \mathbf{C_q} \begin{bmatrix} \frac{\partial p^*}{\partial \Delta p} & \mathbf{0} \\ \mathbf{0} & K_2 \end{bmatrix}^T \tag{3.66}
$$

3.5.3 Combining model and image parameters

When the model is generated, the mean and covariances of all model parameters can be calculated. This section will show how the covariances of the model can be used while fitting the model to an image. By using the covariances of the model, the model parameters can be restricted to realistic values when the model does not match the image very well, while still allowing larger parameter values if the model does match well.

To combine the information from the model with the information of the fitting algorithm, the Mahanalobis distance is used [23]. Both the model and the fitting algorithm provide an estimate of the parameters, and for both a covariance matrix is determined. If the parameter estimates have a gaussian distribution, the maximum-likelihood estimate is equal to the solution which has the minimal sum of mahalanobis distances [23]. The mahalanobis distance for two measurements is given in equation 3.67.

$$d^2 = (x - x1)^T C_1^{-1}(x - x_1) + (x - x_2)^T C_2^{-1}(x - x_2) \tag{3.67}$$

This equation can be solved by setting the derivative to zero. The maximum likelihood estimate x is then given by equation 3.68.

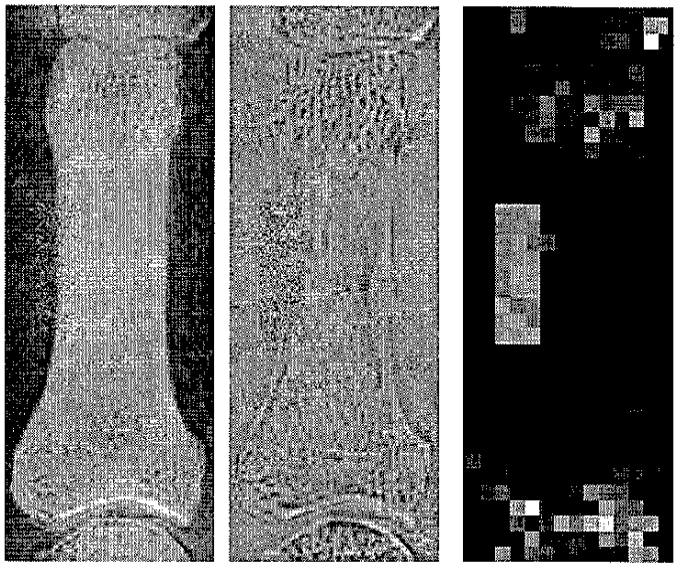
$$x = C(C_1^{-1}x_1 + C_2^{-1}x_2), \quad C = (C_1^{-1} + C_2^{-1})^{-1} \tag{3.68}$$

In each iteration of the model fitting algorithm, this equation is used to calculate the new model parameters.

3.5.4 Test Results

The covariance of the parameter update is calculated using a few approximations. To determine the validity of these approximations, a test is performed. A model is fitted to an image, using it's annotation for the initialization. Figure 3.2 shows three intermediate results from the noise estimation. The warped image is shown in figure 3.2(a). A small amount of gaussian noise was added to the left of this image. The other two images represent R_w and var_E. R_w is the estimate of the noise in the image, var_E is the estimated variance of the difference image E. Brighter colors indicate a higher variance. The variance is estimated in blocks of 10 by 10 pixels. The noise added to the image is clearly seen as a gray rectangle in the left of figure 3.2(c).

The added noise has a variance of $5 \cdot 10^{-6}$. The estimated variance for the pixels to which the noise was added is $5.2 \cdot 10^{-6}$ on average. When comparing the variances for the image without and with the added noise, the variance in q for the horizontal shift increased by 33%, while the variance for the vertical shift increased by less than 6%. Since the noise is added to a part of an image with mainly horizontal gradients, the estimate for the horizontal shift should indeed be less accurate when noise is added.



(a) A warped image, with added noise (I_t)

(b) image used to estimate variance (R_t)

(c) Estimated variance (var_E)

Figure 3.2: Examples for the variance estimation

The method presented above is tested on a set of 5 images. For each image, the model was fitted both with and without using equation 3.68. For all images the estimated covariance for the parameter update is much smaller than the covariance from the model. For most parameters, the variances differed by a factor 100 or more. The model parameters are thus hardly influenced by the model covariances.

The difference in shape between the results of both model fits was lower than 0.1 pixel. It seems that noise in the image is not the largest cause of errors in the parameter update, and the estimated covariances cannot be used to estimate the error in the parameter update. By looking at the difference image, shown in figure 3.2(b), it can be seen that near the edges of the bone, there are edge-like features, so the difference between model and warped image is not independent for all pixels. This assumption was made to be able to determine the covariance of the estimated parameters, but it does not seem to hold very well.

4

Projecting 3D data

One goal of this project is to find a relation between the shape and position of bones, and the measured JSW. The bones are 3-dimensional objects, while the JSW is measured from a 2-dimensional projection. This chapter shows how a 2D projection can be calculated from a 3D data set. Using this method, a series of projections is calculated and the JSW is determined for each projection. The measured JSW's are then used to analyze the sensitivity of the JSW measurement to rotation of the 3D data set.

In this chapter, the elements of a 2D image are called pixels. The 3D data set is called a volume. The volume is a set of voxels, where each value of a voxel represents the attenuation of the material per unit length.

4.1 The volume rendering integral

To calculate a 2D projection from a 3D data set, the effects of all light sources and all attenuations in the path from the light sources to the projection plane should be calculated. This can be done by calculating the volume rendering integral [15]:

$$I(x,r) = \int_0^L C_\lambda(s)\mu(s)e^{-\int_0^s \mu(t)dt}ds \tag{4.1}$$

This integral only takes attenuation of the material into account, other effects, such as reflection, are ignored. I is the amount of light coming from ray direction r to location x at the image plane. C_λ is the amount of light emitted from location s in the direction r. μ is the attenuation of the material (or light extinction coefficient [14]), it has a unit of $1/length$. L is the length of ray r. The integral over t calculates the total attenuation from position s to the image plane at position x.

If there is only one light source (at distance L), the outer integral vanishes. If there is no attenuation at the location of the light source, $\mu(s) = 1$, the volume rendering integral reduces to equation 4.2, where I_0 is the intensity of the light source. This equation calculates the integral of the atten-

uation form the light source at position L, to the image plane at position x.

$$I(x,r) = I_0 e^{-\int_0^L \mu(t)dt} \tag{4.2}$$

The rest of this chapter will only describe how to approximate the integral over t. After calculating this integral, equation 4.2 should be used to calculate the final image I. In the next section, a short overview of projection methods will be given. The integral from equation 4.2 can be approximated with any of these methods. To compare the speed of the algorithms, the 3D data is assumed to have a size of $N \times N \times N$. The resulting 2D image is assumed to have size $M \times M$. After the overview, The projection method used for this thesis is described in more detail.

4.2 Projection methods

This section summarizes three common projecting methods. A method often used for CT images, is the Fourier-slice projection [11]. This method is based on the Fourier slice theorem. This theorem states that a projection of a 3D volume can be calculated by taking the inverse Fourier transform of a 2D plane perpendicular to the projection axis. The 2D plane is calculated by interpolating the Fourier transform of the 3D volume.

The calculation time is bounded by the 2D inverse FFT. For a volume of size N^3, this is in the order of $N^2 \log(N)$ operations. A drawback of this method is, that it cannot generate a perspective projection. A set of parallel-ray projections can be used to generate a set of perspective projections as a post-processing step, but the algorithm itself cannot generate these projections directly. When arbitrary projection angles are handled, the 2D FFT slice should be interpolated from a 3D FFT. This interpolation should be done carefully to obtain good results [8].

Another method to generate projections is ray-casting. Ray-casting is a technique that produces good results, but is very slow compared to the previous method. Ray-casting works by casting a ray into the data volume for each pixel. This ray is sampled along a number of steps, so a line integral can be calculated. Since these samples need to be calculated at non-integer positions, interpolation of the volume data is required. For each pixel a ray is cast, and each ray is evaluated at least N times to sample all voxels. So the number of operations for one projection is at least M^2N.

The third method is called splatting [22]. This method is just the opposite of ray-casting. It iterates over all voxels instead of over all pixels. For each voxel the contribution of the voxel to the 2D image is determined. The time complexity is determined by the number of voxels and the number of pixels. If M is lower or equal to N, the time complexity is in the order of

Method	speed	perspective projection
Fourier-Slice	$N^2 \log(N)$	-
ray-casting	M^2N	+
splatting	M^2N	+

Table 4.1: comparison of three projection methods

N^3, else it is NM^2. The latter follows from the extent of the footprint, as is explained in the next section.

Although the Fourier-slice method is faster, it is not used in this report, since it cannot generate a perspective projection. If the image and volume size are comparable, thus if M is close to N, then splatting and ray-casting have a similar performance. Table 4.1 summarizes the properties of the projection methods.

4.3 Splatting

The projections used in this paper are generated with the splatting method [16]. This method determines the effect of each voxel on the projected image. The effect is calculated by 'splatting' onto the 2D image plane. Figure 4.1 shows an example. The black voxel on the left is 'splatted' onto the projection. This leaves a footprint on the projection; the bell shape on the right. The size the 'splatted' voxel is dependent on the chosen footprint, the volume and the image resolution [22].

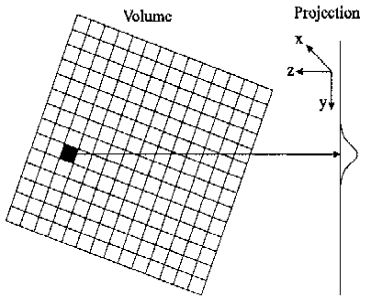


Figure 4.1: Illustration of the splatting algorithm

For the algorithm it is necessary to determine the coordinates $(m,\ n)$ in the projection for each of the voxel's coordinates $(x,\ y,\ z)$ in the 3D volume. First, the data volume is shifted to the origin, the mean value of the coordinates is zero. Next, the voxel coordinates are rotated using rotation matrices for all three axes:

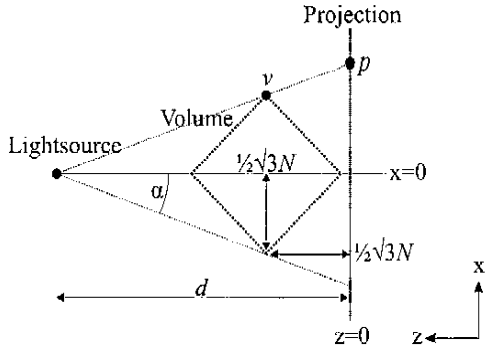


Figure 4.2: Geometry used to generate perspective projections

$$M_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix}, \quad M_y = \begin{bmatrix} \cos(\psi) & 0 & -\sin(\psi) \\ 0 & 1 & 0 \\ -\sin(\psi) & 0 & \cos(\psi) \end{bmatrix},$$

$$M_z = \begin{bmatrix} \cos(\phi) & \sin(\phi) & 0 \\ -\sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

The results are dependent on the order in which these rotations are applied. For this thesis, the volume is first rotated around the x-axis, then the y-axis, and finally the z axis.

$$\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = M_z M_y M_x \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{4.4}$$

After rotation, the volume is shifted by $\frac{1}{2}\sqrt{3}N$ in the z-direction, so all voxels have a positive z-coordinate, independent of the rotation applied to the volume.

When the volume is rotated, the 2D coordinates of each voxel are calculated using perspective projection. Figure 4.2 gives an overview of the relation between the lightsource, the volume data and the resulting projection. The lightsource is assumed to be a point source. The point v is a point in 3D space, for which it's coordinates in the projection plane, p, should be calculated. From figure 4.2, an equation for the x and y coordinates in the projection plan is derived:

$$p_x = \frac{d + v_z}{d} v_x, \quad p_y = \frac{d + v_z}{d} v_y \tag{4.5}$$

These equations are equal to the parallel-ray projection when d equals infinity. The implementation used in this thesis determines the lightsource distance d from the angle α. The relation between these parameters is given by equation 4.6.

$$\tan\alpha = \frac{\frac{1}{2}\sqrt{3}N}{d - \frac{1}{2}\sqrt{3}N} \tag{4.6}$$

Using the distance d, the 2D coordinates p_x and p_y can be shifted and scaled, allowing the projections to be larger or smaller than the 3D volume.

$$\begin{aligned} m &= p_y \cdot zoom + \Delta y \\ n &= p_x \cdot zoom + \Delta x \end{aligned} \tag{4.7}$$

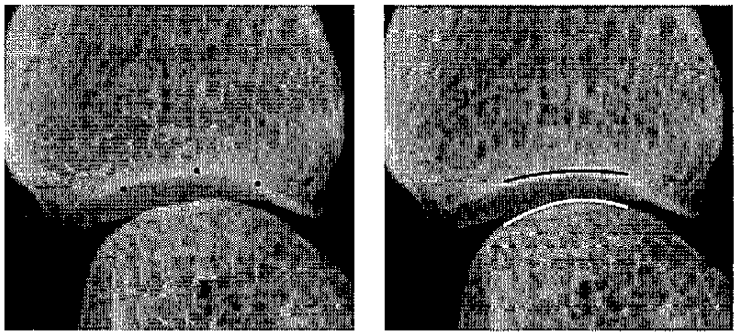
With the calculated 2D coordinates, it is possible to add the footprint to the 2D image at the correct location. As a footprint a separable cubic interpolation kernel is used, it is shown to give better results [12] than gaussian footprints used elsewhere [22]. The kernel is defined by equation 4.8. This kernel is separable, so the kernel values for the x and y-axis are multiplied to get the 2D kernel value. Gaussian footprints are often calculated offline and stored in a lookup-table for rendering. The cubic kernel is simple enough to evaluate directly.

$$h(t) = \begin{cases} 3/2|t|^3 - 5/2|t|^2 + 1 & |t| < 1 \\ -1/2|t|^3 + 5/2|t|^2 - 4|t| + 2 & 1 < |t| \leq 2 \\ 0 & otherwise \end{cases} \tag{4.8}$$

To prevent aliasing, the sampling rate should be at least as large as the voxel and pixel sampling rate. If the size of a voxel is larger than the pixel size, the footprint should increase in size. However, if the size of a voxel is smaller than the pixel size, the footprint size should not be decreased, but it's intensity should be adjusted. Without scaling the intensity, smaller voxels would appear to have a higher attenuation.

To determine the scaling of the interpolation kernel, the size of the kernel in pixels has to be calculated. Multiplying equation 4.5 by equation 4.7 gives the change in size by the transformations, which is $\frac{d}{d+v_z}zoom$. The footprint should be scaled in both dimensions by this factor. If the voxel size is larger than the image size, the intensity of the kernel should be scaled with the square of this value, since the scaling is proportional to the area of the voxel on the image plane.

In literature many extensions to the splatting method are described. Most of these extensions are related to the speed of the algorithm. Since projecting data is not the main part of this thesis, those extensions are not researched. The quality of this projection method is tested in chapter 5.



(a) Initialization (b) Detection

Figure 4.3: semi-automatic JSW detection

4.4 Analyzing 3D data

With the projection method from the previous section, it is possible to analyze the influence of rotation on JSW-measurements. This is done using a set of four 3D images of the MCP-joint. These images were acquired using a μCT-scanner and have been provided by J. Duryea [7]. The datasets have a resolution of 66 μm, which is similar to the resolution of a radiograph.

For rigid bodies, there are 6 degrees of freedom in 3D space, three rotations and three translations. When projecting a 3D dataset, one of these rotations and two of these translations are equal to the rotation and translations of the 2D projection. The third translation, perpendicular to the image plane, causes a change in scale for a perspective projection. This leaves only two rotations which cannot be directly related to a rigid transformation of the projection. For each of these two rotations, the splatting algorithm is used to generate a set of 30 projections. This is done for all four datasets. For each projection, the angle of rotation is slightly different. An example of such a projection is shown in figure 4.3. By measuring the JSW for each projection, the influence of rotation on the JSW measurements can be determined. Using the projections, the JSW is measured with a semi-automatic method. The method is very similar to the method used in [18], which is shown to give good results.

4.4.1 Measuring the JSW

The semi-automatic method needs initialization from the user. First the horizontal center of the joint is indicated. Next three points on the lower edge of the phalangeal bone are indicated. And finally one point on the upper edge of the metacarpal bone has to supplied. Figure 4.3(a) shows these points.

Since only three points are given as initialization, a second order polynomial function is fitted through the points on the phalangeal edge. Using the point on the metacarpal edge, this function is copied and shifted vertically to fit the metacarpal edge. This gives an initial estimate for the edges. Next, an optimization algorithm is used to improve the fit.

Just as in [18], the optimization algorithm matches a fourth order polynomial curve to the bone edges. The criterion for the optimization is based on pixel intensities. The pixel intensities under the polynomials are sampled and summed. To fit the phalangeal edge, the sum of the intensities is maximized. To fit the metacarpal edge, the vertical gradient of the image is calculated and the sum of this gradient, sampled along the polynomial, is maximized. The polynomials are evaluated horizontally across 7 mm, around the horizontal center of the joint, as shown in figure 4.3(b).

A direct search method is used for the optimization algorithm. Each of the parameters is optimized separately. To make the optimization more robust, the image is low-pass filtered to 1/20th of the original sample rate. The polynomial function is parameterized as a set of 13 points. For each point the vertical position is varied and the polynomial is fitted to the 13 points. The vertical positions are varied over a range of .5 mm. Although the re-parametrization does add computational complexity, the search becomes more robust and cannot produce results far from the initial values given by the user.

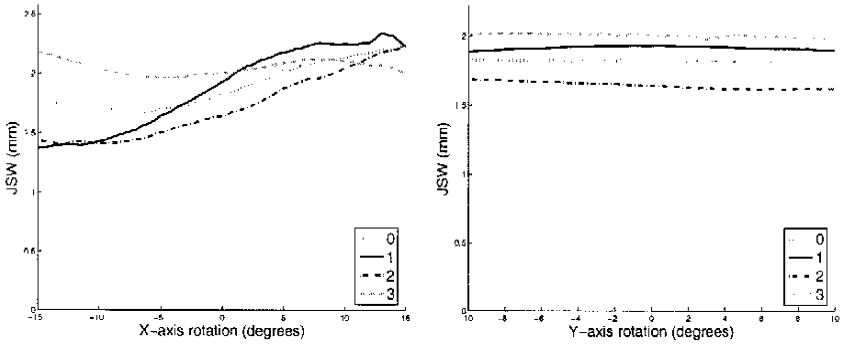
After the optimization, the mean distance between the two polynomials is calculated. This value is used as the JSW. To start the optimization for the next projection, the results from the current projection are used. Since the projections only have small differences, this gives a good initialization.

4.4.2 Analyzing rotation

Figure 4.4 shows the results of the JSW measurements. For each projection, the JSW is plotted against the rotation. The rotation around the X- and Y-axis is varied slightly for each projection. The X-axis is in the horizontal direction of figure 4.3, the Y-axis is in the vertical direction.

As can be seen from the plots, the Y-angle of the projection has little influence on the measured JSW. The X-angle, however, does have a large influence. The difference in JSW between a rotation of -5 and +5 degrees can be up to 0.5 mm, resulting in a difference of 50μm per degree rotation.

Other research [18] shows that a precision higher than 0.1 mm can be achieved for JSW measurements. A study conducted for the 'Nederlandse Vereniging voor Reumatologie' showed that the change in JSW can be as low as 50 μm per year. The results from the measurements in this section show that a difference in alignment of only a few degrees results in a much larger error. It can be concluded that the positioning of the hand is very important for reliable JSW measurements.



(a) rotation around the X-axis (b) rotation around the Y-axis

Figure 4.4: JSW as function of rotation

5

Test results

This chapter will show results of tests of the various parts of this thesis. First, the annotation and the number of model parameters are chosen. Next, two methods for updating the model parameters in the gradient algorithm are compared. For a chosen number of model parameters, it is shown how sensitive the algorithm is to the initial shape estimate. This test is also done for the active appearance update method from section 3.4. Next, it is tested whether the fitting method is sensitive enough to detect small changes in rotation. Finally, the quality of the projection method is tested.

5.1 Model accuracy

Before the fitting algorithm is tested, it is useful to know how well the model can be fitted to another image. It is also important to know how much training data is needed for an accurate model. A test is performed using the annotations of a proximal phalanx of which an example is shown in figure 5.2(a). A test set is used to generate the shape model, the model is then matched to a set of different annotations as close as possible. The mean absolute difference between all shape points and all annotated points is calculated for all annotations in the test set, which contains 20 annotations. For one model, the shape model is extended as described in section 2.5.3. Figure 5.1 shows that this model performs far worse than the others.

The difference between a model generated using 40 and 60 annotations is very small, it seems that 40 annotations are enough to generate an adequate shape model. The accuracy of the model is limited, however. Approximately 40 parameters are necessary for an error lower than 50 μm. This is a large number of parameters; the annotations consist of only 64 2D coordinates, so there are only 128 possible shape parameters.

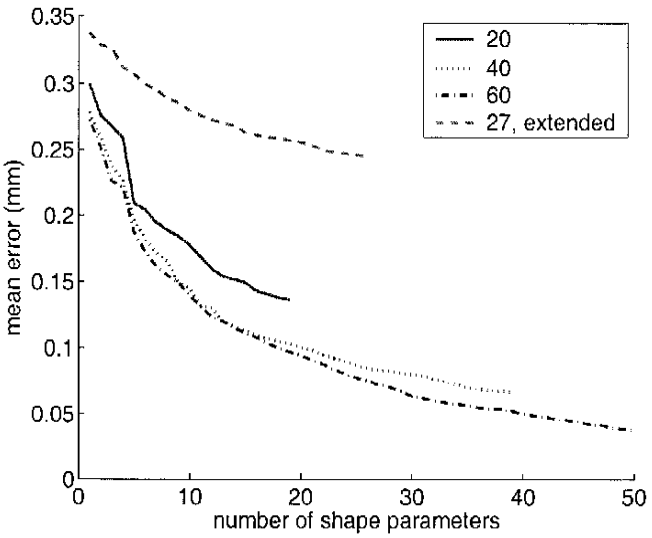


Figure 5.1: Error between annotation and the model, for models generated with a different number of annotations

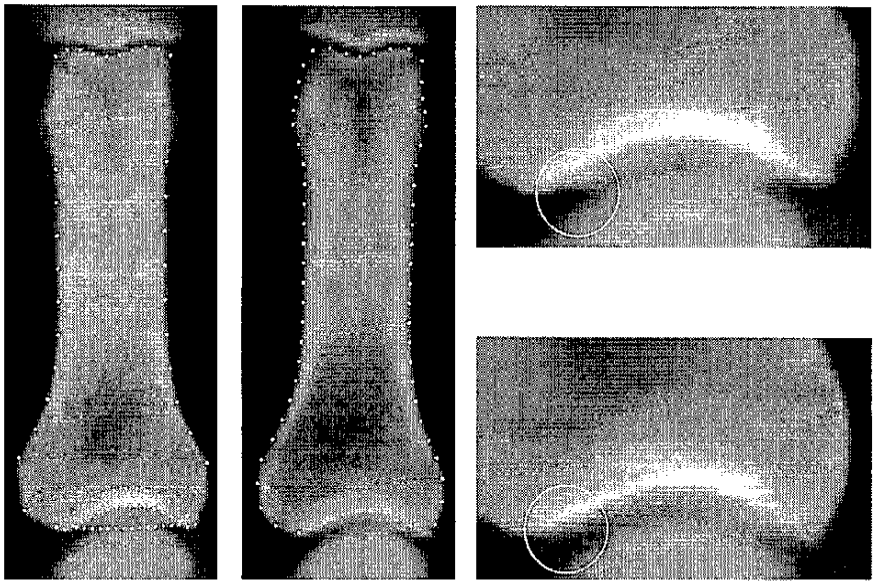
5.2 Model parameters

The image model is generated using a set of annotations. For this thesis two different sets of annotations were used, shown in figure 5.2. The figures show the annotated points as dots on top of the mean texture.

The annotation in figure 5.2(a) has more points, especially near the joint at the bottom. Since the variation is quite large between images at this position, using the annotation labelled 'T' should give a better model. Figure 5.2(c) shows a close up of both annotations, 'T' at the top and 'J' at the bottom. For the area inside the circle, the texture generated using the annotation 'T' is a bit sharper, but the difference is small.

Besides the annotation, the number of parameters should also be chosen to generate a model. The number of parameters can be chosen for both the shape and the texture. The model is made to approximate an image using only a few parameters. Therefore, the total number of model parameters is used when comparing models to each other.

Figure 5.3 shows the error and convergence rate as functions of the total number of shape and texture parameters. The six parameters for pose, brightness and contrast are excluded from this figure, since they are always needed. To test the convergence rate, a set of 54 images was used. Each image was annotated manually. If the mean difference between the annotated points and the result from the model fit was more than 1.8 mm, the result was considered divergent. The error in figure 5.3(b) is the mean quadratic image error between the model and all images of the test set. The test



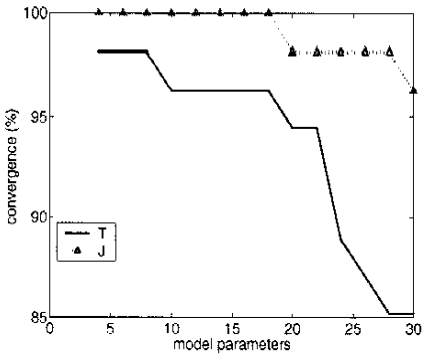
(a) annotation 'T' (b) annotation 'J' (c) close up view of both annotations

Figure 5.2: The annotations used for testing the model

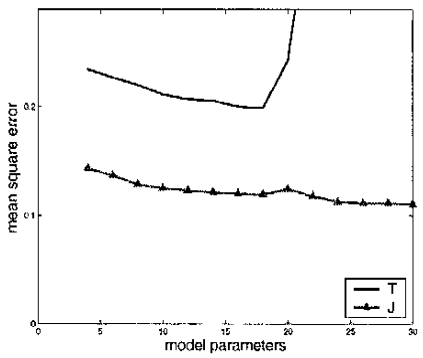
is done for two different annotations. For each annotation 27 images were used to generate the model. The number of iterations for fitting the model is limited to 150 and the number of pixels in the texture model was 20.000. The shape of the last iteration was used as the result of the model fitting algorithm.

The annotation labelled 'J' gives better results. For a high accuracy of the shape model, the number of model parameters should be as high as possible. Figure 5.3(a) however, shows that fitting the model works better for a low number of parameters. For the rest of the tests, 14 shape and 4 texture parameters will be used. Since annotation 'J' gives both a better convergence rate and a pixel error, this annotation will be used for the rest of the tests. There are 81 images and annotations available. 27 are used to generate the models, the other 54 are used to test the model. Using more images to generate the model would give better, but more unreliable results, since there would be less images in the test set.

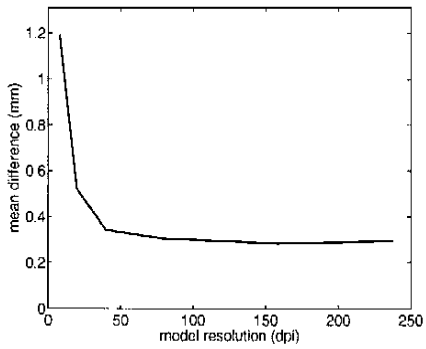
After the number of parameters is determined, a test is done to determine the required resolution of the texture model. A set of texture models is generated with different resolutions. Each model is fitted to the test set using the annotations as initial values. The model fitting algorithm is then run for 80 iterations. Figure 5.3(c) shows the mean difference with the annotation after fitting. When the texture resolution is too small, the edges at the top and bottom of the bone are not sharp enough for the algorithm.



(a) The convergence rate as a function of the number of model parameters



(b) The mean square pixel error as a function of the number of model parameters



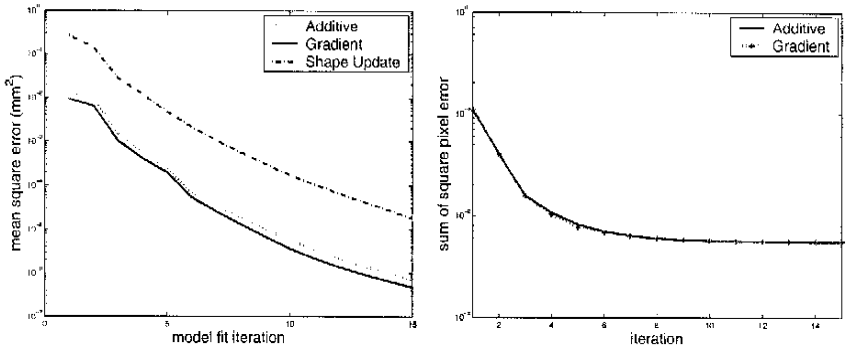
(c) The error between annotation and fitted shape as a function of the texture resolution

Figure 5.3: Effects of the number of model parameters on the results of the model fitting method

For the following tests a resolution of 80 dpi, which equals 20.000 pixels, is used.

5.3 The approximated error criterion

In section 3.3.2 a quadratic term was assumed to be zero in equation 3.17. This was done to derive an analytical expression for the parameter update. While this term is not equal to zero, a test has been performed to determine the effect of this approximation. Using MATLAB's *lsqnonlin* optimizer, equation 3.17 can be solved numerically. A test set of 15 images is used, for which the model fitting is done once using the linear error function (equation 3.22), and once using the error function with the quadratic term. The step size μ was set to 0.5 for both tests. For one image, the optimization using the quadratic error function did not converge. For the other tests, the



(a) Average shape error for both parameter update methods

(b) Average square pixel error for both parameter update methods

Figure 5.4: Comparison of the linear and gradient parameter update

results were compared to each other. The shapes calculated by both fitting methods were compared to each other. The average difference between the linear and square method for all points in the shape set was 1.27μm, while the pixels of the texture model have a size of approximately 300μm when they are warped to the images. So, the linear error function can be used without losing significant accuracy.

5.4 Updating the model parameters

In section 3.3.3 an optimization method for updating the warp parameters is described. This section will show test results for the linear update and this method.

Figure 5.4 shows the average result for a set of 5 images. The annotation of these images is used as an initial estimate for the shape. The annotations are rotated by 1 degree to show the difference between both methods. In figure 5.4(a) the parameter update is done with the gradient descent method. In every iteration the approximation error for both the linear update and the gradient descent update is calculated.

The horizontal axis of figure 5.4(a) shows the warp iterations as described in section 3.3. The vertical axis shows the error in the update of the model points in the image frame. This error is the error that is minimized in section 3.3.3. For reference, the difference between the current and the new shape is also plotted as 'shape update'. The gradient update does give better results, but the difference between both methods is very small.

Figure 5.4(b) shows the pixel error for every iteration of the model fit algorithm. One line shows the image error using additive update, the other shows the image error for the gradient update. The differences are barely

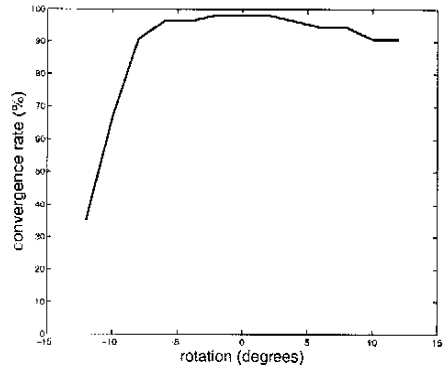
visible, the gradient method does not give a smaller error for every iteration. The differences are so small, that both methods give nearly identical results. Since the linear update can be calculated faster, it should be preferred over the gradient update.

5.5 Initialization

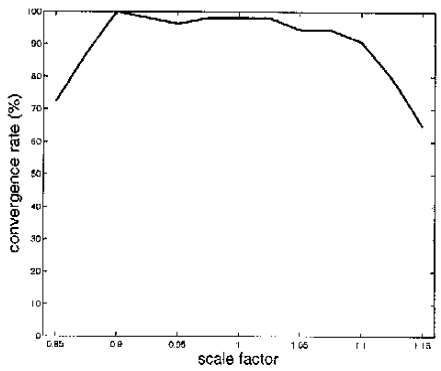
The model parameters can only be estimated when a fairly good initial estimate of the parameters is known. The number of parameters is chosen using the results from section 5.2. There are 14 shape and 4 texture parameters and the resolution of the texture model is 80 dpi. Figure 5.5 shows the convergence rate of the algorithm as a function of an offset in the model initialization. The annotation is changed with either a small rotation, a scaling or a shift. For each of these changes a test is done for 54 images. The point set is used to determine if the algorithm is convergent. A point set is compared with the original annotation, the mean change in position is calculated over all points. If the mean change is less than 1.8 millimeters, the algorithm is considered convergent. The algorithm only converges if the initialization of the horizontal position is very good. When the initialization is off by more than a few millimeters, the contrast estimation described in section 3.1 does not work any more. Since most bright pixels of the template are compared with dark pixels of the warped image and vice versa, a negative value for the contrast is estimated.

Figure 5.5(d) and 5.5(e) show the results for two different models. The model of the proximal phalanx (PP) is the same as used in the previous tests, the other model is shown in figure 5.6. After fitting, all points which are not part of the PP model are removed from the other model. The mean differences in shape between both fitted models and the annotation are then compared in figure 5.5(e). The model of the index finger is less accurate, but is it also less sensitive to initialization. This can be explained by the fact that the position and shape of different bones is correlated. The shape model uses this correlation to restrict variation to likely deformations. By including more bones in the model, the shape of one bone can be estimated from all bones, making the fitting more robust.

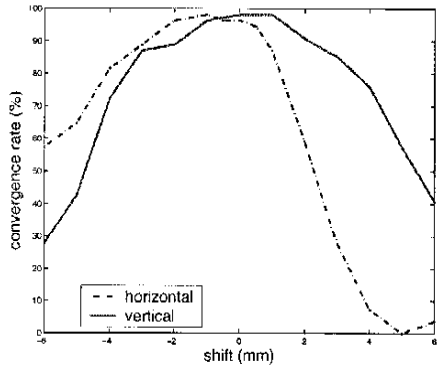
The results indicate that a fairly good initial estimate is necessary. In a typical image the proximal phalanx is roughly 40 by 11 millimeters. The error in position can only be a few millimeters for reasonable convergence, so the algorithm is quite sensitive to the initial estimate. By using a model which covers multiple bones, these results can be improved.



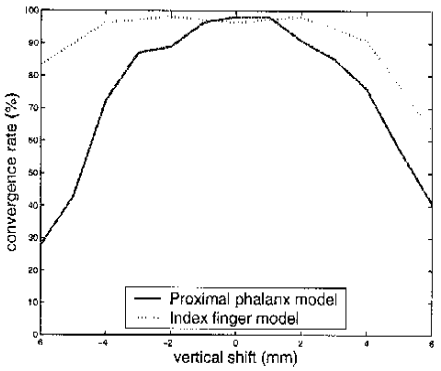
(a) sensitivity to rotation



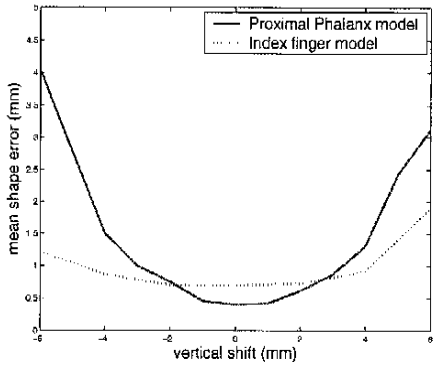
(b) sensitivity to scaling



(c) sensitivity to shift, for the model of the proximal phalanx



(d) sensitivity to shift, for two models



(e) Difference between results of the model fitting and the annotation

Figure 5.5: The effect of model initialization on the convergence rate and the shape error. A fitting result is considered convergent if the difference with the annotation is less than 1.8 mm on average. The shape error is the mean difference between the fitting result and the annotation.

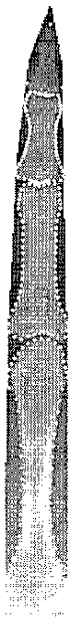


Figure 5.6: Model of the index finger. The mean shape is plotted as white dots on top of the mean texture

Parameter	Range
Shift	± .4 pixels
Rotation	± .2 degrees
Scaling	0.996 .. 1.004
Shape parameters	± 0.4 standard deviations

Table 5.1: Ranges of parameter variation used for calculating the update matrix **R** for Cootes' numerical model fitting method.

5.6 Comparison with active appearance models

The model parameters are estimated from an image using an update matrix. This matrix can be made using an analytical approximation, or using a numerical method. The numerical method is part of the active appearance models described in section 3.4.

To generate the update matrix, it is important to choose the range over which the shape parameters are varied. The range is chosen by hand and listed in table 5.1. The range of variations is chosen such that the resulting update matrix **R** looks similar to matrix generated using the analytical method. The actual images are generated by applying a set of random variations with a uniform distribution to the shape and texture model. The

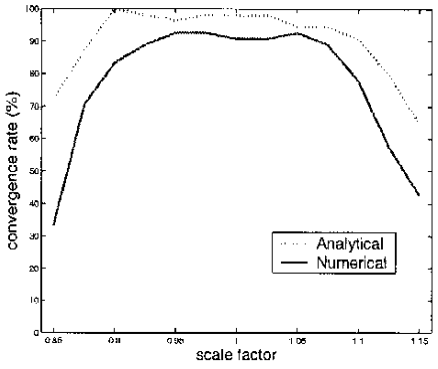


Figure 5.7: Comparison of the analytical and numerical methods

texture model has a size of 20.000 pixels, just as in the previous section. This corresponds to a resolution of roughly 80 dpi, when the mean texture is warped using the annotations.

A set of 300 images is generated to determine the update matrix. Figure 5.7 shows the results. The results of the gradient method are also plotted, for comparison. The analytical method performs better than the numerical method. The convergence rate of the analytical method is higher than the numerical method for every scale factor.

The range of parameter variation, given in table 5.1, is specific for a certain model and it's texture resolution. So even though the results of the numerical method might improve somewhat by adjusting the range of variation that is used to generate the model, the analytical method gives better results, without having to determine a set of parameters empirically.

5.7 Sensitivity of the fitting method

To determine whether the model and fitting method are sensitive enough to detect small changes of rotation, a set of 6 x-ray images was taken of a human hand skeleton. The x-ray source was rotated in steps of 1 degree. For each rotation step, an image of the skeleton was taken.

The skeleton is held together by a set of metal wires. The wires are visible as bright lines in the image. To make the model fitting possible, the wires are removed from the image by using image processing. Pixels above a certain intensity were marked as invalid. All invalid pixels were then interpolated from neighboring pixels using MATLAB's *griddata* function. In figure 5.8(c) the result of the processing can be seen. Near the horizontal center of the bone, at the location of the metal wire, the image is very vague, but the wire itself is not visible.

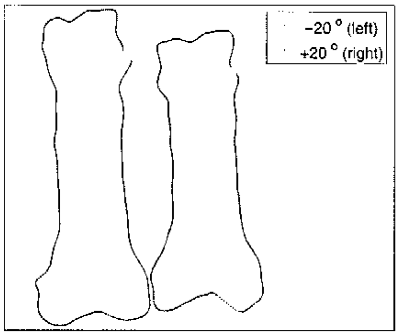
After removing the wires, the model from section 5.5, shown in figure

5.2(b), is fitted to each of the images. The initialization was done by manually entering the pose parameters, as described in section 3.1. Using this initialization, the model fitting is initialized using the mean shape. Any change in shape parameters is the result of the fitting algorithm. Fitting the model gives 6 sets of shape parameters, one set for each image. Using the rotations and the model parameters, it can be determined whether a relation between the two exists. Since there are only six images, fitting a high order polynomial through the data would always give a 'perfect' relation for the test set, but it would be nearly certain that this relation will not hold for any other test set. Therefore a linear relation is assumed, and correlation is used to determine the precision of the linear relation.

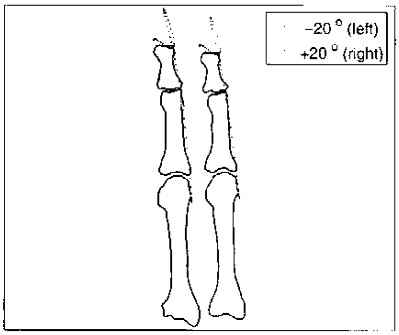
Four shape parameters have a correlation of which the absolute value is higher than 0.95. This indicates that the model and the fitting algorithm are sensitive enough to detect small variations in the image caused by a change of the projection angle. By calculating a linear regression between the angle and each of the parameters, a shape can be estimated from the angle. Figure 5.8(a) shows the estimated shapes for two different projection angles. The shapes are calculated for very large changes of the projection angles, for which the linear relation probably does not hold, but it does give an impression on how the shape changes. While the width of the bone hardly varies, the variation in height is clearly visible. This relation between the height of a bone and the projection angle may be general. Although the height of a bone varies per patient, it remains nearly constant for one patient over time. This measure may be used to determine whether two images of one patient are taken under similar projection angles.

In section 5.5 the model of the single bone is shown to be very sensitive to initialization. The other model used in that section (figure 5.6) was less sensitive, but also less accurate. This model is also fitted to the 6 images, giving 10 shape parameters per image. The correlation between one shape parameter and the rotation was -0.992, for four shape parameters the absolute value of the correlation was higher than 0.9. This shows that this model can also be fitted precisely enough to detect small changes in the projection angle.

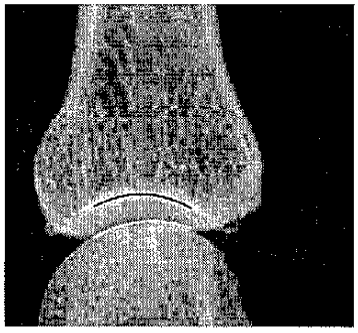
For each of the images, the JSW is measured. This is done using the semi-automatic method of section 4.4.1. Figure 5.8(c) shows the detected edges for one of the images. Figure 5.8(d) shows the measured JSW as a function of the projection angle. The correlation between the two variables is 0.82. Because the JSW measurement is initialized manually for each of the images, the results are less clear than those in section 4.4.2. However, there is clearly a relation between the two. The relation is also large compared to the variation in JSW over time, which can be as low as 50 μm per year.



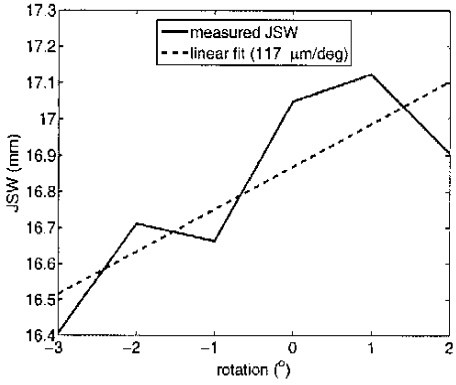
(a) Estimated shape for two projection angles, for the single bone model.



(b) Estimated shape for two projection angles, for the index finger model.



(c) Detected edges used for JSW measurement.



(d) Measured JSW for different projection angles.

Figure 5.8: Results from a set of 2D images, with different projection angles

5.8 The projection algorithm

To test the quality of the projection method, it should be compared with some ground truth data. A simple test is to sum the volume data in one dimension and then compare it to an parallel (non-perspective) projection with the same rotation parameters. For this special case the interpolation kernel is only evaluated at integer positions. Negative effects caused by interpolation are then invisible in the output. The output of the splatting algorithm is shown in figure 5.9. This result can be compared with the summation of voxels in one dimension. The maximum relative error for this example is less than 10^{-13}. This error can be explained by the limited numerical precision of the calculations. Even though this test is not very thorough, it shows that the calculation of coordinates in the 2D projection plane is implemented correctly.

Another test can be done with a simple geometric shape, such as a cube. It is fairly straightforward to calculate the line-integrals of a cube for a non-perspective projection. To test the interpolation of the projection method, the cube is rotated slightly such that splats for non-integer voxel coordinates are calculated.

Figure 5.10 shows the results of this test. A cube with a length of 55 is rotated by 15 degrees. In figure 5.10(a) it's projection is shown. The horizontal line indicates the position of the slice shown in figure 5.10(b). This figure shows the intersection for both the analytical and the projection method. The difference between these two is shown as the error. The error is much larger than in the previous test. Even though the projection method is not perfect, objects can be rendered with an error that is small compared to the amplitude of the output.

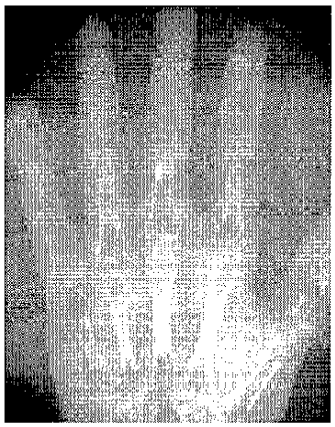
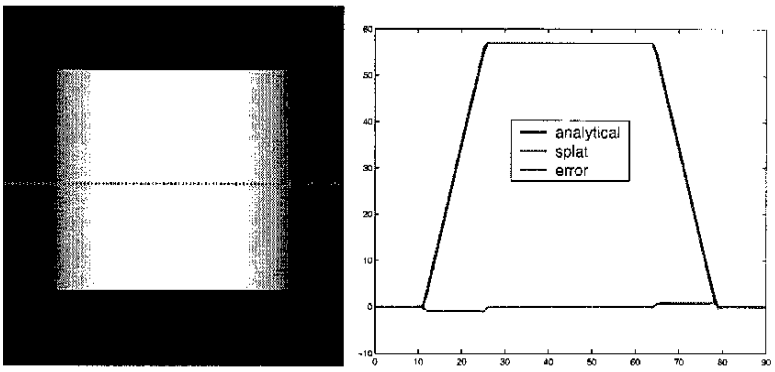


Figure 5.9: Parallel (non-perspective) projection without rotation



(a) Projection of a cube (b) Slice of the projection

Figure 5.10: Test results for the projection of a 3D-volume

6

Conclusions and recommendations

In this thesis volume data is used to analyze the sensitivity of JSW measurements to rotation. A two-dimensional projection of the volume data is made under varying angles of rotation. The JSW is then measured for each projection, using an automatic method. By comparing the sensitivity with the change in JSW caused by progression of rheumatoid arthritis, it is shown that positioning of the hand is very important for reliable JSW measurement. This result is confirmed by the JSW measurements done on six 2D images of which the change in projection angle is known.

To estimate the positioning of the hand from a radiograph a model of an X-ray image of a hand is made. Also a method is presented to fit the model to an image. This methods works well for a simple shape model, but it performs less for a more detailed model. It is also shown that the fitting method is more reliable for models that cover a larger area of hand. The analytical fitting method is compared to an often used numerical approach and the analytical method is shown to have a higher convergence rate. Using a set of 6 images with varying projection angles, it is shown that the model and the fitting method can be used to detect small changes caused by a varying projection angle.

While a complex shape model which includes multiple edges at the bottom of the proximal phalanx cannot be fitted very well, a simpler shape model can be fitted well, as long as the initialization is good. Since the quantitative effect of rotation on the measured JSW differs per bone, it is unknown whether the model can be used to improve JSW-measurement accuracy. More test data is needed to determine this.

There may be a general relation between a change in rotation and the ratio between height and width of a bone. More testing is needed to determine how general this relation is. If such a relation exists, the model parameters can be used to determine if there is a change in rotation between two ra-

diographs. This information could then be used to estimate the accuracy of the JSW measurements.

To generate projections from 3D data, a splatting algorithm was used in this thesis. When generating projections where each projected voxel is at least as large a pixel, as is done in this thesis, the ray-casting method is simpler to implement, since the interpolation kernel does not need to be scaled. When a projection method has to be implemented again, it is recommended to investigate the ray-casting method.

There are various extensions which could be made to the model and the fitting method. The model presented in this paper does not handle overlap of different objects, extending this model to handle overlap could very well improve the accuracy. In literature another warping method is often used, the thin plate spline warp. Using this method might give better results. Calculating the partial derivative of this function, as is needed for the model fitting method, however, might be difficult. The step size in the model fitting method is constant. For non-linear optimization methods many algorithms for adapting the step size are described in literature. A good method could greatly reduce the number of iterations needed in the fitting algorithm. Another useful addition would be automatic initialization of the model. Currently, model initialization is done manually. By adding automatic initialization, the results would be completely independent of the user, which makes the results more consistent.

Finally, it is recommended to position the hand careful and consistent before making a radiograph. Even though the method presented in this thesis seems useful for analyzing radiographs, a lot of work has to be done before JSW measurements can be corrected for a difference in rotation. Consistent positioning of the hand would immediately give more precise JSW measurements.

Bibliography

[1] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework: Part 1. Technical Report CMU-RI-TR-02-16, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2002.

[2] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483, 1996.

[3] S. Boini and F. Guillemin. Radiographic scoring methods as outcome measures in rheumatoid arthritis: properties and advantages. *Annals of the Rheumatic Diseases*, 60:817–827, 2001.

[4] T. Cootes and P. Kittipanya-ngam. Comparing variations on the active appearance model algorithm. In *Electronic Proceedings of The 13th British Machine Vision Conference*, pages 837–846, 2002.

[5] T. Cootes and C. Taylor. Statistical models of appearance for computer vision. `http://www.isbe.man.ac.uk/~bim/Models/app_models.pdf`, march 2004.

[6] Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. Active appearance models. *IEEE Transactions on pattern analysis and machine intelligence*, 23:681–685, 6 2001.

[7] J. Duryea, J. T. Dobbins III, and J. A. Lynch. Digital tomosynthesis of hand joints for arthritis assessment. *Medical Physics*, pages 325–333, March 2003.

[8] John I. Jackson et al. Selection of a convolution function for fourier inversion using gridding. *IEEE Transactions on medical imaging*, 10(3):471–478, September 1991.

[9] Joseph Kearney Hongling Wang and Kendall Atkinson. Arc-length parameterized spline curves for real-time simulation. `http://www.`

cs.uiowa.edu/~kearney/pubs/CurvesAndSurfacesArcLength.pdf, 2002.

[10] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision (darpa). In *Proceedings of the 1981 DARPA Image Understanding Workshop*, pages 121–130, April 1981.

[11] Tom Malzbender. Fourier volume rendering. *ACM Trans. Graph.*, 12(3):233–250, 1993.

[12] Stephen R. Marschner and Richard J. Lobb. An evaluation of reconstruction filters for volume rendering. In *VIS '94: Proceedings of the conference on Visualization '94*, pages 100–107, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[13] Iain Matthews and Simon Baker. Active appearance models revisited. *International Journal of Computer Vision*, 60(2):135 – 164, November 2004. In Press.

[14] Nelson Max. Optical models for direct volume rendering. *IEEE Trans. on vis. and comp. graph.*, 1(2):99–108, June 1995.

[15] Michael Meissner, Jian Huang, Dirk Bartz, Klaus Mueller, and Roger Crawfis. A practical evaluation of popular volume rendering algorithms. In *VVS '00: Proceedings of the 2000 IEEE symposium on Volume visualization*, pages 81–90, New York, NY, USA, 2000. ACM Press.

[16] Klaus Mueller and Roni Yagel. Fast perspective volume rendering with splatting by utilizing a ray-driven approach. In *VIS '96: Proceedings of the 7th conference on Visualization '96*, pages 65–ff., Los Alamitos, CA, USA, 1996. IEEE Computer Society Press.

[17] P. A. Ory. Interpreting radiographic data in rheumatoid arthritis. *Annals of the Rheumatic Diseases*, 62:597–604, 2003.

[18] John T. Sharp, Jill C. Gardner, and Earl M. Bennett. Computer-based methods for measuring joint space and estimating erosion volume in the finger and wrist joints of patients with rheumatoid arthritis. *Arthritis and Rheumatism*, 43:1378–1386, June 2000.

[19] D. van der Heijde et al. Reading radiographs in chronological order, in pairs or as a single films has important implications for the discriminative power of rheumatoid arthritis clinical trials. *Rheumatology*, 38:1213–1220, 1999.

[20] Yongmei Wang and L. H. Staib. Physical model-based non-rigid registration incorporating statistical shape information. *Medical Image Analysis*, 4:7–20, March 2000.

[21] David S. Watkins. *Fundamentals of Matrix Computations.* John Wiley & Sons, 1991.

[22] Lee Westover. Footprint evaluation for volume rendering. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 367–376, New York, NY, USA, 1990. ACM Press.

[23] Xiang Sean Zhou, Dorin Comaniciu, and Alok Gupta. An information fusion framework for robust shape tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(1):115–129, 2005.