ACTIVE APPEARANCE MODELS

Theory, Extensions & Cases

2nd edition

Mikkel Bille Stegmann

LYNGBY 2000

Master Thesis IMM-EKS-2000-25

IMM

Printed by IMM, Technical University of Denmark

Preface

This thesis has been prepared over six months at the Section for Image Analysis, Department of Mathematical Modelling, IMM, at The Technical University of Denmark, DTU, in partial fulfillment of the requirements for the degree Master of Science in Engineering, M.Sc.Eng.

To supplement this thesis refer to the produced web-site on Active Appearance Models at `http://www.imm.dtu.dk/~aam/`

It is assumed that the reader has a basic knowledge in the areas of statistics and image analysis.

Lyngby, August 2000

2nd edition preface

Minor errors in grammar and mathematical notation have been corrected. Further has the notation regarding pose-transfomations in the regression and optimization part been clarified a bit. Thanks to Hans Henrik Thodberg for drawing my attention to this.

Lyngby, September 2000

Mikkel Bille Stegmann
[email: mikkel@stegmann.dk]

Acknowledgements

Though this thesis is done by a one-man-band, the result would never have been the same without the support, encouragement and assistance from the following people.

Hans Henrik Thodberg, Pronosco A/S for establishing a partly sponsorship during the thesis period, with which I could fully concentrate on my thesis and without which my sparetime would have been somewhat less colorful. I would also like to thank Hans Henrik for his interest and fruitful discussions during the thesis period. Pronosco A/S digitized and annotated the metacarpal radiographs.

Cardiac MRIs were provided and annotated by M.D. Jens Christian Nilsson and M.D. Bjørn A. Grønning, H:S Hvidovre Hospital. They are also both thanked for their exciting discussions and good comments on my thesis work in general. M.Sc. Torben Lund is also gratefully acknowledged for providing the initial contact and all his practical help during the collaboration.

Cross-section images of pork carcasses were provided by the Danish Slaughter-Houses and annotated by M.Sc. Rune Fisker, DTU and clustered by M.Sc. Nicolae Duta, Michigan State University.

M.D. Lars Hyldstrup, H:S Hvidovre Hospital provided all metacarpal radiographs.

Home ground thanks goes out to my academic advisors, Ph.d. student Rune Fisker and Dr. Bjarne K. Ersbøll, without whom I wouldn't have done my master thesis in image analysis and with whom it has become magnitudes better due to your huge encouragement and support. Thank you.

Further more I would like to thank the whole image analysis section for providing a pleasant and inspiring atmosphere. In particular I would like to thank my office-mate Klaus Baggesen Hilger for all his help and fruitful discussions, Dr. Rasmus Larsen for being substitute advisor during the holidays of Dr. Bjarne K. Ersbøll, Lars Pedersen for lending me his computer during his stay at Yale. Finally Henrik Aanæs is thanked for proofreading my manuscript.

Hans P. Palbøl for proofreading and for doing courses and projects with me for the bulk part of my years at DTU. It has been great fun studying with you.

The indirect support of Karlheinz Brandenburg and Justin Frankel is also gratefully acknowledged. Keep up the good work.

My family for all your love, support and encouragement. I am sorry that my thesis work coincided with your wish to move to another place. Help is outstanding :-)

A heartfelt thanks goes out to my girlfriend Katharina for all your love, support and patience when I was only thinking about strange formulas.

At last, I would like to thank Poul Rose for your initial encouragement to do mathematical-based research.

Mikkel Bille Stegmann

Abstract

This thesis presents a general approach towards image segmentation using the learning-based deformable model Active Appearance Model (AAM) proposed by Cootes et al. The primary advantage of AAMs is that a priori knowledge is learned through observation of both shape and texture variation in a training set. From this, a compact object class description is derived, which can be used to rapidly search images for new object instances.

A thorough treatment and discussion of the theory behind AAMs is given, followed by several extensions to the basic AAM, which constitutes the major contribution of this thesis. Extensions include automatic initialization and unification of finite element models and AAMs. All of these have been implemented in a structured and fast C++ framework; the AAM-API.

Finally, case studies based on radiographs of metacarpals, cardiovascular magnetic resonance images and perspective images of pork carcass are presented. Herein the performance of the basic AAM and the developed extensions are assessment using leave-one-out evaluation.

It is concluded that AAMs – as a data-driven and fully automated method – successfully can perform object segmentation in challenging and very different image modalities with very high accuracy. In two of three cases subpixel accuracy were obtained w.r.t. object segmentation.

Keywords: Deformable Template Models, Snakes, Principal Component Analysis, Shape Analysis, Non-Rigid Object Segmentation, Non-Rigid Object Detection, Initialization, Optimization, Finite Element Models.

Resumé

I denne afhandling præsenteres en general metode til segmentering af billeder; den indlæringsbaserede deformable template model Active Appearance Model (AAM), introduceret ved Cootes et al. Hovedbidraget i AAM metoden er, at forhåndsviden omkring form og tekstur er indlært igennem et givent træningssæt. Fra dette opbygges en kompakt beskrivelse af den klasse af objekter modellen repræsenterer. Beskrivelsen anvendes efterfølgende til at gennemsøge nye billeder for forekomster af objekttypen.

Der gives en grundig beskrivelse og diskussion af det matematiske fundament for AAM metoden, efterfulgt af adskillige udvidelser af AAM-formuleringen. Dette udgør det væsentlige bidrag i denne afhandling. Blandt de udformede udvidelser er automatisk initialisering samt en kombinering af AAM og finit-element metoder. Alt arbejde er udført som et struktureret og effektivt C++ bibliotek (AAM-API).

Afslutningsvis præsenteres case-studier af røntgenbilleder af mellemhåndsknogler, magnetisk resonans billeder af det menneskelige hjerte samt perspektiviske billeder af svinekød. Baseret på disse, er der udført en grundig evaluering af AAM metodens præcision vha. leave-one-out evaluering.

Det er konkluderet at AAM – som en fuldautomatisk og data-drevet metode – succesfuldt og med høj præcision, kan udføre segmentering i endog meget udfordrende og forskelligartede billed-modaliteter. I to af de tre cases er der opnåedet subpixel præcision mht. objektsegmentering.

Nøgleord: Deformable Template Models, Snakes, Principal komponent analyse, Formanalyse, Ikke-rigid object segmentering, Ikke-rigid objekt detektion, Initialisering, Optimering, Finit-element modeller.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis deals with a core problem within computer vision research, namely the segmentation of non-rigid objects in digital images.

Several decades of research in computer vision and pattern recognition have resulted in fast, robust and accurate methods for the detection of rigid objects. However, until about a decade ago most methods would fail in presence of objects with great variability regarding shape and appearance. Nevertheless humans would have no problems in classifying these into equivalence classes – i.e. faces, hands, fish etc.

To overcome these limitations a whole family of methods is spun off to address the problem of variability. Many of these also address the problems of partial image evidence, occlusion and severe noise. This family is called the *deformable template models*.[1]

A novel and fairly sophisticated deformable template model – of which this thesis is dedicated to – is the learning-based *Active Appearance Model* [10].

The constructivist theorists of cognitive psychology believe that the process of seeing is an active process in which our world is constructed from both the retinal view and prior knowledge [56]. This constitutes the motivation of all learning-based computer vision methods such as the Active Appearance Models (AAMs). To stress this point try to see what is depicted at figure 1.1 without reading the upside-down caption.

[1] Alternatively: deformable templates or deformable models.

Figure 1.1: Image interpretation using a priori knowledge. What is depicted here? Courtesy of Preece et al. [56].

Tip: Try looking for a Dalmatian dog sniffing leaves in a park.

Without a priori knowledge, it would never have been possible to decipher the black blobs of figure 1.1. This is the main assumption behind the constructivist approach [56]. Namely, that visual perception involves the intervention of representations and memories such as "dog", "park" etc. Mundy [53] also stress this point (pp. 1213, l. 5-8):

> *"... This process of* recognition*, literally to RE-cognize, permits an aggregation of experience and the evolution of relationships between objects based on a series of observations."*

These thoughts are essential to fully grasp the motivation and design of learning-based models in computer vision.

1.1 Motivation and Objectives

The Active Appearance Model was proposed by Cootes et al. [10] in 1998 as one of the more sophisticated deformable template models. This is

primarily due to a unique and effective combination of techniques that enables searching of images with a flexible, compact and complete model representation feasible in the millisecond range.

To our knowledge only one group beside Cootes' namely the vision group at University of Iowa, has published work on AAMs [52]. Due to this fact and the overall elegance of AAMs, work in this area constituted a suitable relevant and challenging topic for a master thesis. Thus, the main objectives set forth was:

- Discuss, document and explore the basic AAM.
- Design general extensions to the AAM approach.
- Evaluate AAMs through a set of relevant and varying cases.

An additional aim was to provide a platform for further development on AAMs through an open, free and well-documented application programmers interface (API).

1.2 Thesis Overview

The thesis is structured into five parts where each part requires knowledge from the preceding parts.

Part I: Statistical Models of Shape and Texture Presents the statistical and mathematical foundations for AAMs.

Part II: Active Appearance Models Combines the statistical models into performance effective AAMs and presents various extensions to the basic AAM.

Part III: Implementation Introduces the developed application programmers interface on AAMs.

Part IV: Experimental Results Assess AAM performance and problems on real-life cases.

Part V: Discussion Proposes ideas for further work on AAMs and draws conclusions from the thesis work.

Some of the techniques and preliminary results can be found in abbreviated form in a paper prepared during the thesis period [67]. The paper is attached as appendix B.

1.3 Mathematical Notation

To ease reading and understanding; the used notation conventions are enumerated below.

Vectors are viewed upon as column vectors and typeset in non-italic lowercase boldface using commas to separate elements: $\mathbf{v} = [a, b, c]^{\mathbf{T}}$

Vector functions are typeset in non-italic boldface: $\mathbf{f}(\mathbf{v}) = \mathbf{v} + \mathbf{v}$

Matrices are typeset in non-italic boldface capitals as:

$$\mathbf{M} = \begin{bmatrix} a & b \\ c & c \end{bmatrix}$$

Matrix diagonals are manipulated using the $\mathrm{diag}(a)$ operator. If a is a vector of lenght n an $n \times n$ diagonal matrix is produced. If a is an $n \times n$ matrix the diagonal is extracted into a vector of lenght n.

Dot-product operator is typeset as: $\mathbf{a} \cdot \mathbf{b} = \sum_i a_i b_i$

Sets are typeset using curly braces: $\{\alpha\ \beta\ \gamma\}$

"Unit vectors" are typeset as: $\mathbf{1} = [1\ \cdots\ 1]^{\mathbf{T}}$

Unit matrices are typeset as:

$$\mathbf{I} = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}$$

1.4 Nomenclature

Variables used without an explicit denotation conform to the nomenclature below.

$\mathbf{I}$	An image (or the unit matrix).
E	The error energy in model to image fit.
k	The number of Euclidean dimensions. In the planar case $k = 2$.
n	The number of points on a shape.
N	The number of shapes in a training set.
m	The number of texture samples inside a shape.
$\mathbf{x}$	A normal vector, or a planar shape.
Σ	The covariance matrix (also called the dispersion matrix).
$\mathbf{\Lambda}$	A diagonal matrix of eigenvalues.
$\mathbf{\Phi}$	A matrix of eigenvector columns.
λ_i	The i^{th} eigenvalue.
$\boldsymbol{\phi}_i$	The i^{th} eigenvector.
θ	A 2D shape rotation given in radians.

Chapter 2

Background

In recent years, the model-based approach towards image interpretation named *deformable template models* has proven very successful. This is especially true in the case of images containing objects with large variability.

As the precise definition of a deformable template model we will use the one of Fisker [26]:

> **Definition 1**: *A deformable template model can be characterized as a model, which under an implicit or explicit optimization criterion, deforms a shape to match a known object in a given image.*

Among the earliest and most well known deformable template models is the Active Contour Model – known as *Snakes* proposed by Kass et al. [46]. Snakes represent objects as a set of outline landmarks upon which a correlation structure is forced to constrain local shape changes. In order to improve specificity, many attempts at hand crafting a priori knowledge into a deformable template model have been carried out. These include Yuille's et al. [73] parameterization of a human eye using ellipses and arcs.

In a more general approach, while preserving specificity Cootes et al. [15] proposed the Active Shape Models (ASM) where shape variability is learned through observation. In practice, this is accomplished by a training set of annotated examples followed by a Procrustes analysis [35] combined with a principal component analysis.

A direct extension of the ASM approach has lead to the Active Appearance Models [10]. Besides shape information, the textual information, i.e. the pixel intensities across the object, is included into the model. The AAM has been further developed in [13, 14, 22].

Jain et al. [44, 45] classifies deformable template models as either being *free form* or *parametric* where the former denotes model deformation dependent on *local* constraints on the shape and the latter *global* shape constraints. By building statistical models of shape and texture variation from a training set, AAM qualifies as being a parametric deformable template model.

Quite similar to AAMs and developed in parallel herewith, Sclaroff & Isidoro proposed the Active Blob approach [43, 58]. Active Blobs is a real-time tracking technique, which captures shape and textual information from a prototype image using a finite element model (FEM) to model shape variation. Compared to AAMs, Active Blobs deform a static texture, whereas AAMs change both texture and shape during the optimization.

Also based on a prototype – and a finite element framework using Galerkin interpolants – is the Modal Matching technique proposed by Sclaroff & Pentland [59]. Objects are matched using the strain energy of the FEM. A major advantage is that the objects can have an unequal number of landmarks and it easily copes with large rotations.

For further information on deformable template models, the reader is referred to the surveys given in [26, 4, 44, 51].

Part I

Statistical Models of Shape and Texture

Chapter 3

Introduction

This part provides an in-depth treatment and discussion of how Active Appearance Models build its statistical models of shape and texture and how these are combined into one unified model.

The notation, treatment and even some parts of the algorithms is occasionally somewhat different from the treatment by the inventors of AAMs [10, 14]. However, the overall ideas are the same.

Figure 3.1: The three steps of handling shape and texture in AAMs.

The handling of shape and texture can be viewed as dual processes.[1] The setup of these processes is quite similar to other data handling processes though the composition of techniques is quite unique.

The first step is the data acquisition. Hereafter follows a suitable normalization after which the data are ready to be analyzed and described in terms of statistical models. The process setup is given as a flow chart on figure 3.1.

[1] Though the texture mode in reality is defined in terms of the shape model.

To stress the coherence between shape and texture handling the steps are specified below.

Capture

Shape Captured by defining a finite number of points on the contour of the object in question.

Texture Captured by sampling in a suitable image warping function (e.g. a piece-wise affine, thin-plate or another warp function).

Normalization

Shape Brought into a normalized frame by aligning shapes w.r.t. position, scale and orientation using a Procrustes analysis.

Texture Removing global linear illumination effects by standardization.

Statistical Analysis

Shape & Texture Principal Component Analysis is performed to achieve a constrained and compact description.

The level of detail in the following chapters is adjusted so that the current implementation can be understood and/or redone solely upon this description.

Chapter 4

Shape Model Formulation

4.1 Overview

The following chapter provides the fundamental concepts and techniques needed to understand the statistical models of shape used in AAMs. First the concept of a *shape* is defined, next – the basis of the mathematical framework – the concept of *landmarks* is treated. The chapter is concluded by demonstrating how shape variation can be efficiently modeled using *principal component analysis*.

Effort has been put into making the treatment rich on examples and references to further treatment of the topics.

4.2 Shapes and Landmarks

The first matter to clarify is: What do we actually understand by the term *shape*? A starting point could be the few definitions given below:

> *"A collection of corresponding border points."* [62]

> *"The characteristic surface configuration of a thing; an outline or a contour."* [1]

> *"Something distinguished from its surroundings by its outline."* [1]

Though the above captures the characteristics of the term shape fairly well; this thesis will adapt the definition by D.G. Kendall [20] and define shape as:

> **Definition 2**: **Shape** *is all the geometrical information that remains when location, scale and rotational effects are filtered out from an object.*

The term shape is – in other words – invariant to Euclidean transformations. This is reflected in figure 4.1.

The next question that naturally arises is: How should one describe a shape? In everyday conversation, unknown shapes are often described as references to known shapes – e.g. *"Italy has the shape of a boot"*. Such descriptions can obviously not easily be utilized in an algorithmic framework.

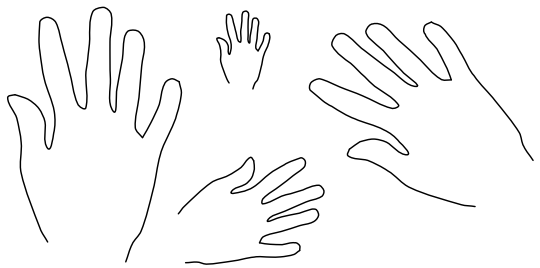


Figure 4.1: Four exact copies of the same shape, but under different euclidean transformations.

One way to describe a shape is by locating a finite number of points on the outline. Consequently, the concept of a *landmark* is adapted [20]:

> **Definition 3**: *A* **landmark** *is a point of correspondence on each object that matches between and within populations.*

Dryden & Mardia further more discriminates landmarks into three subgroups [20]:

- **Anatomical landmarks** Points assigned by an expert that corresponds between organisms in some biologically meaningful way.
- **Mathematical landmarks** Points located on an object according to some mathematical or geometrical property, i.e. high curvature or an extremum point.
- **Pseudo-landmarks** Constructed points on an object either around the outline or between landmarks.

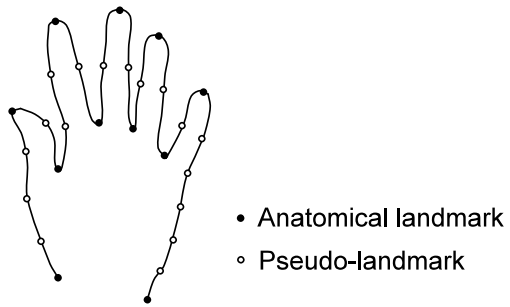


Figure 4.2: A hand annotated using 11 anatomical landmarks and 17 pseudo-landmarks.

Synonyms for landmarks include homologous points, nodes, vertices, anchor points, fiducial markers, model points, markers, key points etc.

A mathematical representation of an n-point shape in k dimensions could be to concatenate each dimension into a kn-vector.

In the following only 2D shapes are considered, all though most of the results in the remaining part of the thesis extend directly to 3D – and often even higher dimensionalities. Hence $k = 2$.

The vector representation for planar shapes would then be:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n]^{\mathrm{T}} \tag{4.1}$$

Notice that the above representation does not contain any explicit information about the point connectivity.

4.3 Obtaining Landmarks

Although the concept of landmarks conceptually is very useful – the acquisition of such can be very cumbersome. For 2D images the process could involve manually placing of hundreds of points including constantly comparing to other annotations to ensure correspondence.

It should be needless to mention that this approach becomes substantially more tedious and cumbersome in the 3D (x, y, z) and 4D $(x, y, z, time)$ case.

To ease the burden effort has been put into the development of automatic and semi-automatic placement of landmarks.

One could claim that solving the problem of automatic placement of landmarks equals solving *the general correspondence problem* in computer vision. Myriads of attempts have been done regarding that matter. If it successfully could be done one would only need to annotate one "gold" image of the object in question, and the solution to the correspondence problem could solve the object segmentation in this bottom-up fashion.

This is – in general – unfortunately not possible. For that reason we need to constrain the solution space somewhat. Defining these constraints – and handling outliers – constitutes the major part of all work in the field of computer vision.

One way to constrain the solution space, is to use a manual trained sparse model to initially place the landmark points. If necessary, the points can be corrected manually. Notice however – in the case of basic AAMs – if no adjustments of the points are done, then the training example only adds new texture variation to the model, since the shape itself is a superposition of known shapes.

Regarding semi-automatic placement of landmarks several successful attempts have been done. Most of these assume that a dense sampling of the object outline is given beforehand.

One example is that of Sclaroff & Pentland [59] where a *finite element model* (FEM) using Galerkin interpolants is built over the set of shape

points[1]. The correspondence of a single point to another set of points is determined by comparing the displacement vectors of the point as given by the finite element model. In this way the point set is described in terms of generalized symmetries (i.e. the objects FEM-eigenmodes). One major advantage hereof is that the two point sets can be of unequal sizes.

Another example include the work of Duta et al. [21] where *k-means* clustering of the training shapes is performed and followed by a Procrustes analysis of each cluster. Each shape is trimmed into a sparse representation and compared to a dense representation of the remaining shapes. Comparisons are collected into a *pair wise mean alignment matrix* which is used to determine the best point correspondences. Point connectivity is used to increase robustness.

Another example of using connectivity information while establishing point correspondences is the work by Andresen & Nielsen [2] where 3D registration solutions is constrained to a surface and an assumption of a non-folding displacement field. This method is called *Geometry-Constrained Diffusion*.

Efford [25] identifies landmarks from a dense object contour by estimating the curvature using a gaussian smoothing of the contour representation to obtain robustness from contour noise. Mathematical landmarks are consequently identified as extremums in the curvature function. Semi-landmarks are interpolated as uniformly spaced points between the mathematical landmarks.

Quite recently[2] Walker et al. [71] proposed an iterative algorithm for determine point correspondence. This was accomplished using feature vectors for each pixel inside a manually drawn region of interest (ROI) of each training image. Feature vectors were first and second order normalized Gaussian partial derivatives. It was shown that AAMs trained on the automatically generated training set could be of higher quality than AAMs built on hand annotated training sets.

However, since AAMs consider both shape and texture as object class descriptors we suggest that the point correspondence determination should not solely rely on changes in curvature or direction of FEM-eigenmode displacement vectors. Solutions should further be constrained by including

[1] Can be either sparse or dense.
[2] ECCV, Dublin, June 2000.

information of the textural variation around the points. This will lead to better models.

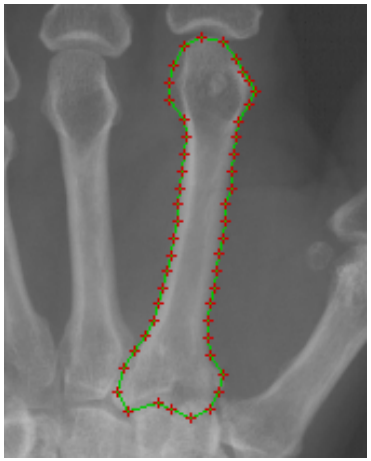


Figure 4.3: Metacarpal-2 annotated using 50 landmarks.

Another substantial problem in obtaining landmarks is that some object classes lack points, which can be classified as corresponding across examples. This is especially true for many biological shapes and is treated in depth by Bookstein [6]. Another source for this type of problems is occlusions in the 3D to 2D projections in perspective images. Annihilation of points can also be observed in malformation of organic shapes.

All examples in the remains of this part of the thesis are based on annotations of a bone in the human hand. The image modality is radiographs and the precise name of the bone is *metacarpal-2*. An example of such an annotation is given in fig. 4.3. For further information on AAMs on metacarpals, refer to the experimental part of this thesis.

As a concluding remark, one should remember that annotations by human experts itself contains errors. This is the core problem in obtaining the so-called *gold standards* to evaluate medical image analysis[3] techniques against. To evaluate this type of noise, annotations are often done several times by several graders to assess the *between* grader and *within* grader variation. This is also known as the *reproducibility* and *repeatability*.

[3] And all other learning-based image analysis techniques for that matter.

4.4 Shape Alignment

To obtain a true shape representation – according to our definition – location, scale and rotational effects need to be filtered out. This is carried out by establishing a *coordinate reference* – w.r.t. position, scale and rotation, commonly known as *pose* – to which all shapes are aligned.

Some literature also operates with the concept of *pre-shape* as introduced by Kendall [20]. Pre-shape is the last step toward true shape – rotational effects still need to be filtered out.

Below an alignment procedure for obtaining such a coordinate reference is described. This is commonly known as *Procrustes analysis*[4] [6, 14, 20, 35].

To aid the understanding and handling of a set of shapes from the same object class the term *shape space* is introduced. Adapted to our nomenclature from [20] this is defined as:

> **Definition 4**: *The* **Shape Space** *is the set of all possible shapes of the object in question. Formally, the shape space* Σ_k^n *is the orbit shape of the non-coincident* n *point set configurations in the* $\mathbb{R}^k$ *under the action of the Euclidean similarity transformations.*

If k denotes the Euclidean dimensions and n denotes the number of landmarks, the dimension of the shape space, follows from the above definition:

$$M = kn - k - 1 - \frac{k(k-1)}{2} \tag{4.2}$$

> **Proof** Initially we have kn dimensions. The translation removes k dimensions, the uniform scaling one dimension and the rotation $\frac{1}{2}k(k-1)$ dimensions.

[4] As a curiosity Procrustes was the nickname of a robber in Greek mythology called Damastes, who lived by the road from Eleusis to Athens. He offered travelers hospitality on a magical bed that would fit any guest. His humor was to stretch the ones who were too short to fit the bed – until they died – or, if they were too tall, to cut off as much of their limbs as would make them short enough. This rather unpleasant practice continued until Damastes was killed by Theseus, son of Æthra and the Athenian king Ægeus. Another nickname for Damastes was *The one who stretches*.
The term *Procrustes Analysis* was coined by Hurley & Cattell in 1962 [20].

If a relationship between the distance in shape space and Euclidean distance in the original plane can be established, the set of shapes actually forms a Riemannian manifold containing the object class in question. This is also denoted as the *Kendall shape space* [6]. This relationship is called a *shape metric*.

Often used shape metrics include the Hausdorff distance [42], the strain energy [59] and the Procrustes distance [21, 20, 6, 14]. Where the two former compare shapes with unequal amount of points, the latter requiring corresponding point sets. In the following, the Procrustes distance is used.

4.4.1 The Procrustes Shape Distance Metric

The Procrustes distance is a least-squares type shape metric that requires shapes with one-to-one point correspondence.

To determine the Procrustes distance between two shapes involves four steps:

1. Compute the centroid of each shape.
2. Re-scale each shape to have equal size.
3. Align w.r.t. position the two shapes at their centroids.
4. Align w.r.t. orientation by rotation.

The rotational step and the graphic interpretation of the Procrustes distance can be seen on fig. 4.4.

Mathematically the squared Procrustes distance between two shapes, $\mathbf{x}_1$ and $\mathbf{x}_2$, is the sum of the squared point distances after alignment:

$$P_d^2 = \sum_{j=1}^{n} [(x_{j_1} - x_{j_2})^2 + (y_{j_1} - y_{j_2})^2] \tag{4.3}$$

The *centroid* of a shape can be interpreted as center of mass of the physical system consisting of unit masses at each landmark. Thus to compute the centroid:

$$(\overline{x}, \overline{y}) = \left(\frac{1}{n} \sum_{j=1}^{n} x_j, \frac{1}{n} \sum_{j=1}^{n} y_j \right) \tag{4.4}$$

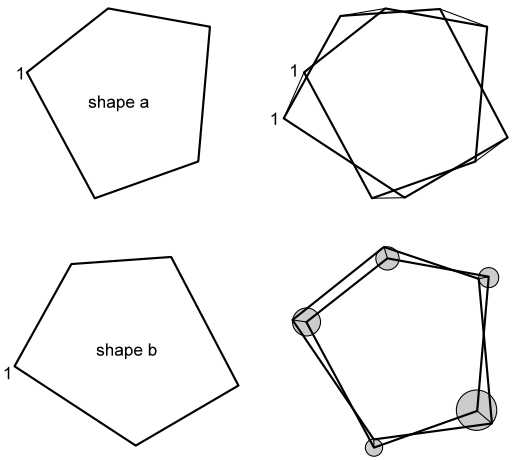


Figure 4.4: The Procrustes distance.

To perform step 2 we obviously need to establish a *size metric*:

> **Definition 5**: *A* **shape size metric** $\mathrm{S}(\mathbf{x})$ *is any positive real valued function of the shape vector that fulfils the following property:*
>
> $$\mathrm{S}(a\mathbf{x}) = a\mathrm{S}(\mathbf{x})$$

In the following the *Frobenius norm* is used as a shape size metric:

$$\mathrm{S}(\mathbf{x}) = \sqrt{\sum_{j=1}^{n}[(x_j - \overline{x})^2 + (y_j - \overline{y})^2]} \tag{4.5}$$

Another often used scale metric is the *centroid size*[5]:

$$\mathrm{S}(\mathbf{x}) = \sum_{j=1}^{n}\sqrt{(x_j - \overline{x})^2 + (y_j - \overline{y})^2} \tag{4.6}$$

[5]This metric also posses the interesting property that $2n\mathrm{S}(\mathbf{x})^2$ equals the sum of the inter-landmark distances [20].

To filter out the rotational effects the following *singular value decomposition* technique is used as suggested by Bookstein [6]:

1. Arrange the size and position aligned $\mathbf{x}_1$ and $\mathbf{x}_2$ as $n \times k$ matrices[6].
2. Calculate the SVD, $\mathbf{UDV}^{\mathrm{T}}$, of $\mathbf{x}_1^{\mathrm{T}}\mathbf{x}_2$
3. Then the rotation matrix needed to optimally superimpose $\mathbf{x}_1$ upon $\mathbf{x}_2$ is $\mathbf{VU}^{\mathrm{T}}$. In the planar case:

$$\mathbf{VU}^{\mathrm{T}} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{4.7}$$

As an alternative Cootes et al. suggest a variation on Procrustes distance-based alignment by minimizing the closed form of $|\mathrm{T}(\mathbf{x}_1) - \mathbf{x}_2|^2$ where T in the Euclidean case is:

$$T\begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{4.8}$$

The term $|\mathrm{T}(\mathbf{x}_1) - \mathbf{x}_2|^2$ is then simply differentiated w.r.t. (a, b, t_x, t_y). The solution to alignment using the affine transformation is also given. Notice however that this transformation changes the actual shape. Refer to [14] for the calculations.

This concludes the topic of how to provide a consistent metric in shape space and how to align two shapes.

4.4.2 Aligning a Set of Shapes

All though an analytic solution exists [41] to the alignment of a set of shapes the following simple iterative approach suggested by Bookstein et al. [6, 14] will suffice.

1. Choose the first shape as an estimate of the mean shape.
2. Align all the remaining shapes to the mean shape.
3. Re-calculate the estimate of the mean from the aligned shapes
4. If the mean estimate has changed return to step 2.

[6]In the planar case $k = 2$.

Convergence if thus declared when the mean shape does not change significantly within an iteration. Bookstein notes that two iterations of the above should be sufficient in most cases.

The remaining question is how to obtain an estimate of the mean shape?[7] The most frequently used is the *Procrustes mean shape* or just the *Procrustes mean*: If N denotes the number of shapes:

$$\overline{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \tag{4.9}$$

This is also referred to as the Frechét mean.

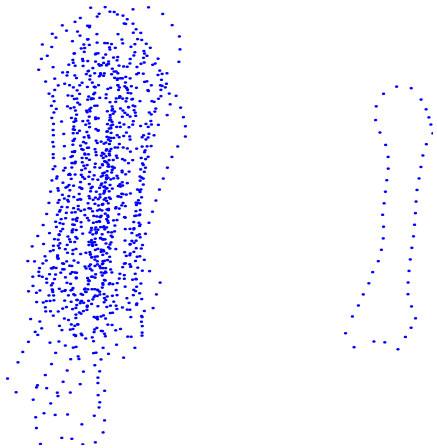


Figure 4.5: A set of 24 unaligned shapes. Notice the position-outlier to the right.

As an example figure 4.5 shows the landmarks of a set of 24 unaligned shapes. The result of the shape alignment can be seen as a scatter plot on figure 4.6 (a) where the mean shape is superimposed as a fully drawn shape. This is called the *point distribution model* (PDM) of our shapes. How to model the variation within the PDM is the topic of the forthcoming section.

[7] Also called the shape prototype.

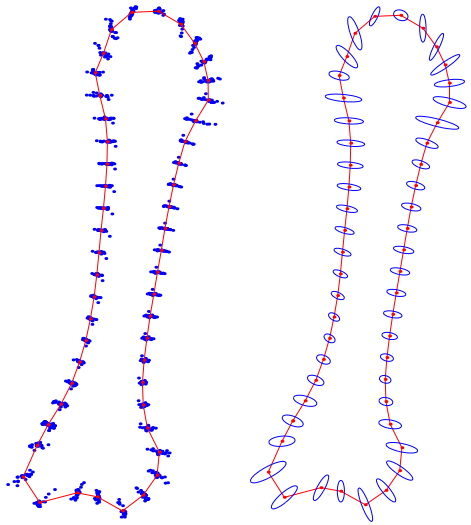
To give a more clear impression of the point variation over the set of shapes, an ellipsis has been fitted to each mean model point in figure 4.6 (b). [8]

4.5 Modelling Shape Variation

As the previous sections have considered the definition and handling of shapes, this section will demonstrate how intra-class shape variation can be described consistently and efficiently.

The fact alone that equivalence classes of shapes can be established – e.g. *"We have a collection of shapes formed as leaves."* – hint us in the direction

[8] Where the major and minor axes are the eigenvectors of the point covariance matrix (scaled to 3 std.dev.). More about this technique used on the complete set of points in the following chapter.



(a) (b)

Figure 4.6: (a) The PDM of 24 aligned shapes. (b) Ellipsis fitted to the single point distribution of figure (a).

that there must be some sort of inter-point correlation present. Naturally, as this actually is the only degrees of freedom left to constitute the perception of a shape, since – according to the definition of shape – all position, scale and rotational effects are filtered out.

A classical statistical method of dealing with such redundancy in multivariate data is the linear orthogonal transformation; *principal component analysis* (PCA). Based on work by Karl Pearson the principal component analysis method was introduced by Harold Hotelling in 1933 [54]. The principal component analysis is also known as the *Karhunen-Loeve transform*.

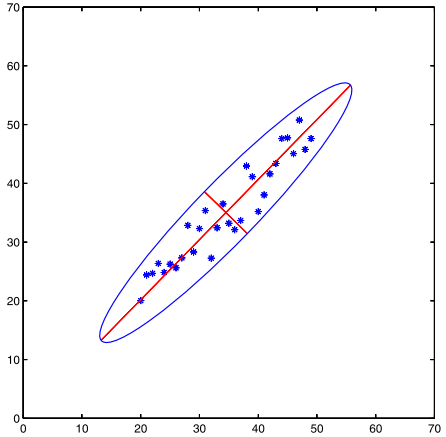


Figure 4.7: Principal axis. 2D example.

Conceptually the PCA performs a *a variance maximizing rotation* of the original variable space. Furthermore, it delivers the new axes ordered according to their variance. This is most easily understood graphically. In figure 4.7 the two principal axes of a two dimensional data set is plotted and scaled according to the amount of variation that each axis explains.

Hence, the PCA can be used as a dimensionality reduction method by producing a projection of a set of multivariate samples into a subspace constrained to explain a certain amount of the variation in the original samples. One application of this is visualization of multidimensional data.[9] In connection to the example in figure 4.7 one could choose to discard the second

[9]However – one should also consider the *multidimensional scaling – MDS* technique for this special purpose.

principal axis, and visualize the samples by the orthogonal projection of the point upon the first (and largest) axis.

Another application of PCA is to determine any underlying variables or to identify intra-class clustering or outliers.

In our application of describing shape variation by using PCA a shape of n points is considered a data point in a $2n^{\text{th}}$ dimensional space. But as stated above it is assumed that this space is populated more sparsely than the original $2n$ dimensions. It has been seen in eq. (4.2) that the reduction should be at least $k - 1 - \frac{1}{2}k(k-1)$ due to the alignment process.

In practice the PCA is performed as an eigenanalysis of the covariance matrix of the aligned shapes. The latter is also denoted the *dispersion matrix*.

It is assumed that the set of shapes constitute some ellipsoid structure of which the centroid can be estimated[10]:

$$\overline{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \tag{4.10}$$

The maximum likelihood (ML) estimate of the covariance matrix can thus be given as:

$$\mathbf{\Sigma}_s = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^{\mathrm{T}} \tag{4.11}$$

To prove the assumption of point correlation right, the correlation matrix of the training set of 24 metacarpal-2 bones is shown in figure 4.8. In the case of completely uncorrelated variables, the matrix would be uniformly gray except along its diagonal. Clearly, this is not the case.

The point correlation effect can be emphasized by normalizing the covariance matrix by the variance. Hence the *correlation matrix*, Γ, is obtained.

$$\mathbf{V} = diag(\frac{1}{\sqrt{diag(\Sigma)}}) = \begin{bmatrix} \frac{1}{\sigma_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\sigma_n} \end{bmatrix} \tag{4.12}$$

[10]Notice that this estimate naturally equals the mean shape.

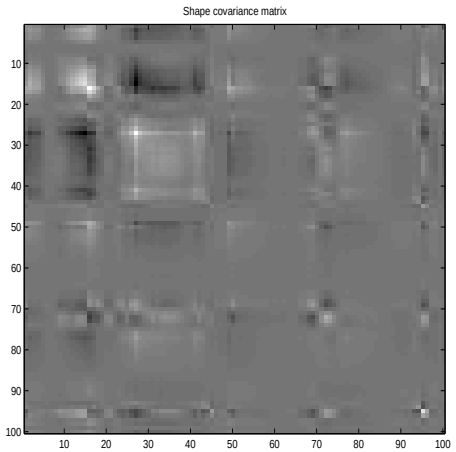


Figure 4.8: Shape covariance matrix. Black, grey & white maps to negative, none & positive covariance.

$$\Gamma = \mathbf{V}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}} \tag{4.13}$$

Recalling the shape vector structure; $xxyy$; it is from figure 4.9 – not surprisingly – seen that the x- and y-component of each point is somewhat correlated.

The principal axes of the $2n^{th}$ dimensional shape ellipsoid are now given as the eigenvectors, $\mathbf{\Phi}_s$, of the covariance matrix.

$$\mathbf{\Sigma}_s\mathbf{\Phi}_s = \mathbf{\Phi}_s\mathbf{\Lambda}_s \tag{4.14}$$

Where $\mathbf{\Lambda}_s$ denotes a diagonal matrix of eigenvalues

$$\mathbf{\Lambda}_s = \begin{bmatrix} \lambda_i & & \\ & \ddots & \\ & & \lambda_{2n} \end{bmatrix} \tag{4.15}$$

corresponding to the eigenvectors in the columns of $\mathbf{\Phi}_s$.

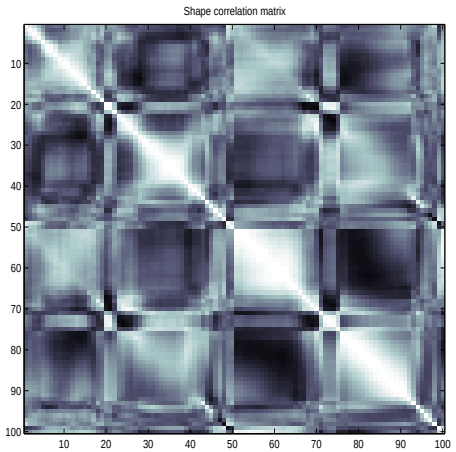


Figure 4.9: Shape correlation matrix. Black, white maps to low, high correlation.

$$\mathbf{\Phi}_s = \begin{bmatrix} & & \\ \phi_1 & \cdots & \phi_{2n} \\ & & \end{bmatrix} \tag{4.16}$$

A shape instance can then be generated by deforming the mean shape by a linear combination of eigenvectors:

$$\mathbf{x} = \overline{\mathbf{x}} + \mathbf{\Phi}_s\mathbf{b}_s \tag{4.17}$$

where $\mathbf{b}_s$ is shape model parameters. Essentially the point or *nodal representation* of shape has now been transformed into a *modal representation* where modes are ordered according to their *deformation energy* – i.e. the percentage of variation that they explains.

Notice that an eigenvector is a set of displacement vectors, along which the mean shape is deformed. To stress this point, the first eigenvector has been plotted on the mean shape in figure 4.10 (a). The resulting deformation of the mean shape can be seen in figure 4.10 (b).

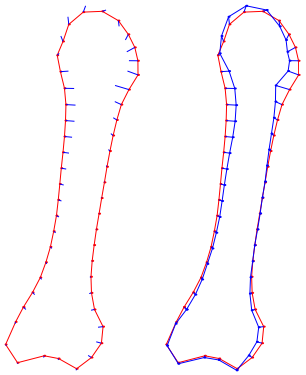
As a further example of such modal deformations, the first three – most significant – eigenvectors are used to deform the mean metacarpal shape in figure 4.11.

What remains is to determine how many modes to retain. This leads to a trade-off between the accuracy and the compactness of the model. However, it is safe to consider small-scale variation as noise. It can be shown that the variance along the axis corresponding to the i^{th} eigenvalue equals the eigenvalue itself, λ_i. Thus to retain p percent of the variation in the training set, t modes can be chosen satisfying:

$$\sum_{i=1}^{t} \lambda_i \geq \frac{p}{100} \sum_{i=1}^{2n} \lambda_i \tag{4.18}$$

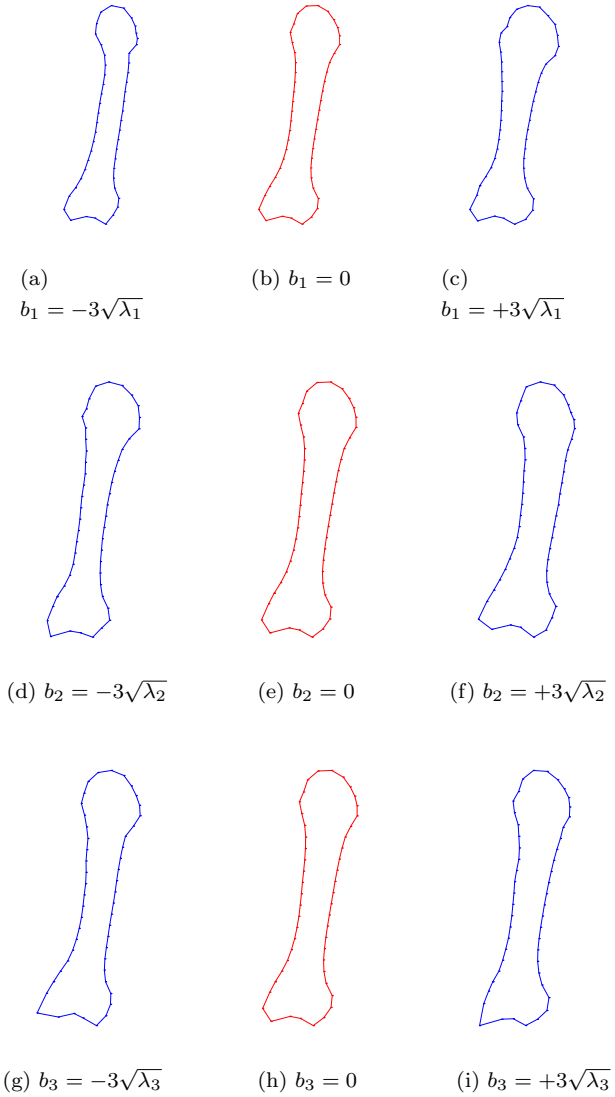
Notice that this step basically is a regularization of the solution space.

In the metacarpal case 95% of the shape variation can be modeled using 12 parameters. A rather substantial reduction since the shape space originally had a dimensionality of $2n = 2 \times 50 = 100$. To give an idea of the decay



(a) (b)

Figure 4.10: (a) Mean shape and deformation vectors of the 1st eigenvector. (b) Mean shape, deformation vectors of the 1st eigenvector and deformed shape.



(a) $b_1 = -3\sqrt{\lambda_1}$ (b) $b_1 = 0$ (c) $b_1 = +3\sqrt{\lambda_1}$

(d) $b_2 = -3\sqrt{\lambda_2}$ (e) $b_2 = 0$ (f) $b_2 = +3\sqrt{\lambda_2}$

(g) $b_3 = -3\sqrt{\lambda_3}$ (h) $b_3 = 0$ (i) $b_3 = +3\sqrt{\lambda_3}$

Figure 4.11: Mean shape deformation using 1st, 2nd and 3rd principal mode. $b_i = -3\sqrt{\lambda_i}$, $b_i = 0$, $b_i = 3\sqrt{\lambda_i}$.

rate of the eigenvalues a percentage plot is shown in figure 4.12.

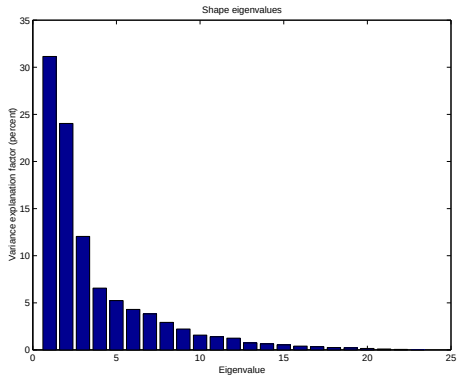


Figure 4.12: Shape eigenvalues in descending order.

To further investigate the distribution of the $\mathbf{b}_s$-parameters in the metacarpal training set $b_{s,2}$ is plotted as a function of $b_{s,1}$ in figure 4.13. These are easily obtained due to the linear structure of (4.17) and since the columns of Φ_s are inherently orthogonal.

$$\mathbf{b}_s = \mathbf{\Phi}_s^{-1}(\mathbf{x} - \overline{\mathbf{x}}) = \mathbf{\Phi}_s^{\mathrm{T}}(\mathbf{x} - \overline{\mathbf{x}}) \tag{4.19}$$

No clear structure is observed in figure 4.13, thus concluding that the variation of the metacarpal shapes can be meaningfully described by the linear PCA transform. This however is not a general result for organic shapes due to the highly non-linear relationships observed in nature.

An inherently problem with PCA is that it is linear, and can thus only handle data with linear behavior. An often seen problem with data given to a PCA is the so-called *horse-shoe effect*, where pc1 and pc2 is distributed as a horse-shoe pointing either upwards or downwards[11]. This simple non-linearity in data – which can be interpreted as a *parabola bending* of the hyper ellipsoid – causes the PCA to fail in describing the data in a compact

[11]Since the PCA chooses its signs on the axes arbitrary.

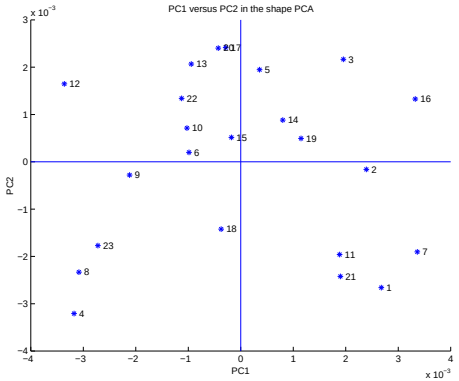


Figure 4.13: PC1 ($b_{s,1}$) vs. PC2 ($b_{s,2}$) in the shape PCA.

and consistent way, since the data structure can not be recovered using linear transformations only. This topic is treated in depth later on.

This section is concluded by remarking that the use of the PCA as a statistical reparametrisation of the shape space provides a compact and convenient way to deform a mean shape in a controlled manner similar to what is observed in a set of training shapes. Hence the shape variation has been modeled by obtaining a compact shape representation.

Furthermore the PCA provides a simple way to compare a new shape to the training set by performing the orthogonal transformation into **b**-parameter space and evaluating the probability of such a shape deformation. This topic is treated in depth in section 12.2 – Performance Assessment.

4.5.1 Reducing Non-linearity

One source of non-linearity in the shape model is the alignment procedure. In the alignment procedure described earlier the shapes were size-normalized by scaling to unit scale using $1/\mathrm{S}(\mathbf{x})$. In this way, the corners of a set of aligned rectangles with varying aspect ratio forms a unit circle (see fig. 4.15, the unaligned shapes are shown on fig. 4.14). Due to this non-linearity the PCA on the shapes must use two parameters to span the shape space: $\lambda_1 = 99.6\%$, $\lambda_2 = 0.4\%$ even though variation only exists on

one parameter (the aspect ratio). A closer look at figure 4.15 also shows that the overlaid mean shape does not correspond to an actual shape in the training set.

To avoid this non-linearity in the aligned training set the shape can be projected into *tangent space* by scaling by $1/(\mathbf{x} \cdot \overline{\mathbf{x}})$ [12, 14].

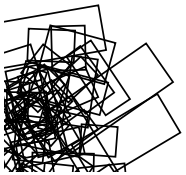


Figure 4.14: Training set of 100 unaligned artificially generated rectangles containing 16 points each.

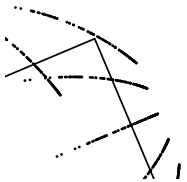


Figure 4.15: Point cloud from aligned rectangles sized to unit scale, $|\mathbf{x}| = 1$. The mean shape is fully shown.

The projection into tangent space align all rectangles with corners on

straight lines (see fig. 4.16) thus enabling modeling of the training set using only linear displacements.

Notice how the mean shape is contained in the training set since the PCA now only uses one parameter, $\lambda_1 = 100\%$, to model the change in aspect ratio.

In this way, the distribution of PCA-parameters can be kept more compact and non-linearities can be reduced. This leads to better and simpler models.

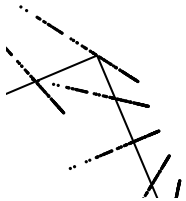


Figure 4.16: Point-cloud from aligned rectangles sized to unit scale, $|\mathbf{x}| = 1$, and transformed into tangent space. The mean shape is fully shown.

4.5.2 Improving Specificity in the PDM

Aside the alignment procedure, several factors can contribute to the breakdown of the PCA, due to non-linearites.

- **Articulated shapes** Shapes with pivotal rotations around one or more points are inherently non-linear.
- **Bad landmarks** Manually placed landmarks can easily cause non-linearies.

- **Bending** Can also be interpreted as a piece-wise rotation.

Examples of the breakdown includes the tadpoles, watch model and chromosomes of Sozou et al. [63, 64]. Chromosomes also constituted the original example of a PDM breakdown in [15]. Examples of the tadpole model are given in figure 4.17. Here a clear non-linear dependency between b_1 and b_3 (lower right) is seen, which also is clearly reflected in the deformations given by the principal modes (upper left). This behavior has been coined the *horse-shoe effect*, and serves as an example on structure that can't be decomposed by the linear PCA, namely one of the most simple non-linear dependencies one can think one; the quadratic.

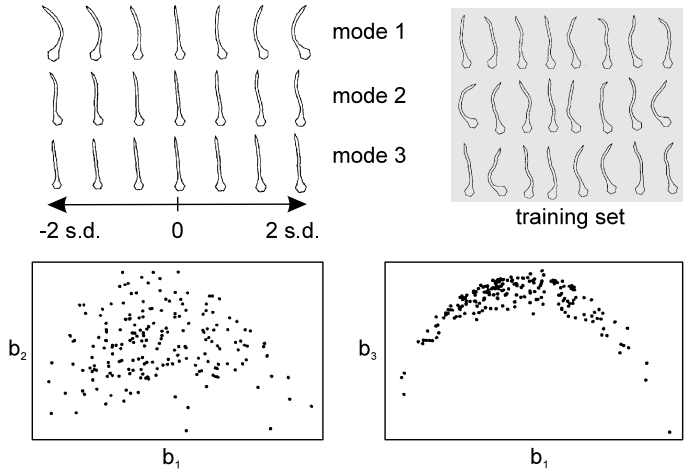


Figure 4.17: Tadpole example of a PCA breakdown. Notice in mode 1, how the head size and length is correlated with the bending. This is easily seen in the scatter plot of PCA parameter 1 vs. 3 (lower right), where b_3 has a simple non-linear dependency of b_1. Adapted from [64].

Another way to view the problem, is that the PCA-approach is based on the assumption, that all shapes of the class ends on the same manifold. More precisely as a hyper ellipsoid cluster in the new basis spanned by the PCA. However when dealing with non-linearity the ellipsoid changes into

a more structured form. Dealing with objects with discreetized behavior[12] for example, also changes the ellipsoid, here into a clustered distribution of PCA parameters. To accommodate this, it is proposed to approximate the distribution with a mixture of gaussian blobs in shape parameter space [12] thus avoiding illegal shapes in the PDM. This is accomplished using expectation maximization to fit the blobs to the shape parameters of the training set. The problem of implausible shapes in non-linear shapes has also been addressed by Heap & Hogg [38] using polar coordinates in the PDM without loss of computational performance. The algorithm automatically classifies landmarks into either the Cartesian or polar domain.

One of the early attempts include Sozou et al. where polynomial regression (PRPDM) was used to fit higher order polynomials to the non-linear axis of the training set – see figure 4.17. Later Sozou et al. out-performed the PRPDM by using a back propagation neural network[13] to perform non-linear principal component analysis (MLPPDM). The downside to this approach is a substantial increase in the – off-line – computation of the PDM.

The bottom line of all this is – if your shape parameters lie in more than one cluster or if dependencies between shape parameters exist – then the standard PDM is not specific enough. This can lead to more or less serious artifacts.

4.6 Summary

Throughout this chapter, a complete mathematical framework and the necessary set of definitions and concepts have been introduced to the extent that an efficient treatment of shape and shape modeling can be done. Further more selected topics have been discussed to increase the understanding of the AAM framework.

Emphasis has been put on the application in the Active Appearance Models though the methods presented are applicable in a wide range of situations.

[12] For example if certain parts of an object, only can reside in certain positions. Think of the second hand on a quarts watch.

[13] A multi-layer perceptron to be precise.

Chapter 5

Texture Model Formulation

5.1 Overview

To form a complete model of *appearance* one must not only consider *shape*. To stress this point observe that shape is only well defined by inferring from knowledge of the pixel neighborhood. One must also consider the information constituted by pixels themselves.

In the following a complete scheme for capturing pixel information, using image warping, and modeling pixel variation, using principal component analysis, is described.

In the shape case, the data acquisition is straightforward because the landmarks in the shape vector constitute the data itself. In the texture case one needs a consistent method for collecting the texture information between the landmarks, i.e. an image warping function needs to be established. This can be done in several ways. Here, a piece-wise affine warp based on the Delaunay triangulation of the mean shape is used. Thus to obtain texture information from the training set, each shape is warped to a *reference shape*[1] and sampled. Hereafter a photometric normalization of the

[1] Here the mean shape is used as reference shape.

obtained textures is done to remove influence from global linear changes in pixel intensities. Hereafter, the analysis is identical to that of the shapes. Hence, a compact PCA representation is derived to deform the texture in a manner similar to what is observed in the training set.

5.2 Object Texture

Contrary to the prevalent understanding of the term *texture* in the computer vision community, this concept shall be used somewhat differently below. The main reason for this is that most literature on AAMs uses this definition of texture, probably due to the close resemblance of some of the AAM-techniques to techniques in computer graphics.

In computer graphics the term texture relates directly to the pixels mapped onto virtual 2D and 3D surfaces. Thus, we derive at the following definition:

> **Definition 3:** **Texture** *is the pixel intensities across the object in question (if necessary after a suitable normalization).*

In the shape case, the actual data capture was straightforward because the landmarks in the shape vector constituted the data itself. In the texture case one needs a consistent method of collecting the texture information between the landmarks.

This method is called *image warping* and is described in detail in the forthcoming section.

As mathematical representation of the texture of an object, a vector is chosen:

$$\mathbf{g} = [g_1, g_2, \ldots, g_m]^{\mathrm{T}} \tag{5.1}$$

Here m denotes the number of pixel samples over the object surface.

5.3 Image Warping

Image warping is a simple matter of transforming one spatial configuration of an image into another. Hence, a simple translation of an image can be

considered an image warp. Formally: $\mathbf{I} \in \mathbb{R}^k \mapsto \mathbf{I}' \in \mathbb{R}^k$ and pictorial in the planar case, $k = 2$, in figure 5.1.

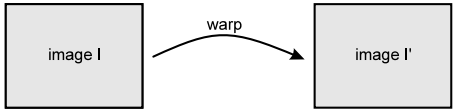


Figure 5.1: Image warping.

For a survey on warping techniques refer to Glasbey & Mardia [33]. Since AAMs are landmark-based we shall use the class of image warping methods that considers the mapping of one arbitrary point set $\{\mathbf{x}_1 \ \dots \ \mathbf{x}_n\}$ into another $\{\mathbf{x}'_1 \ \dots \ \mathbf{x}'_n\}$ where each point is represented as $\mathbf{x} = [x, y]^{\mathrm{T}}$. Formally written as a continuous vector valued mapping function such that:

$$\mathbf{f}(\mathbf{x}_i) = \mathbf{x}'_i \quad \forall \quad i = 1 \dots n \tag{5.2}$$

5.3.1 Piece-wise Affine

The most simple construction of an n-point based warp is to assume that $\mathbf{f}$ is locally linear. To utilize this in a planar framework such as 2D AAMs one need to define the term *locally* more tightly. One approach is to partition the convex hull of the points, using a suitable triangulation such as the Delaunay triangulation.

The Delaunay triangulation connects an irregular point set by a mesh of triangle's each satisfying the Delaunay property. This means that no triangle has any points inside its circumcircle, which is the unique circle that contains all three points (vertices) of the triangle – see figure 5.2. The Delaunay triangulation of the mean metacarpal from previous is given in figure 5.3. Notice that due to the concave nature of the metacarpal the Delaunay triangulation produce triangles outside the actual shape.

For a thorough treatment of planar triangulation and mesh representation, refer to [60].

Whereas dimensionality is of concern, the Delaunay triangulation extents to 3D[2], though its complexity increases drastically [69].

[2]Here the circumcircle becomes a "circumsphere".

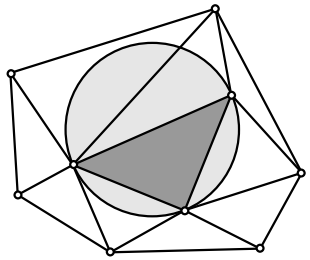


Figure 5.2: Circumcircle of a triangle satisfying the Delaunay property.

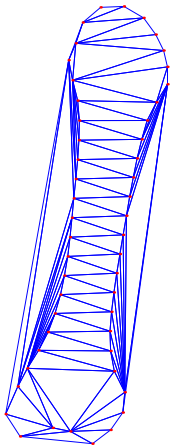


Figure 5.3: Delaunay triangulation of the mean shape.

The warp is now realized by applying the triangular mesh of the first point set in $\mathbf{I}$ to the second point set in $\mathbf{I}'$. In that each point on each triangle can be uniquely mapped upon the corresponding triangle of the second point set by an affine transformation, which basically consist of scaling, translation and skewing. If $\mathbf{x}_1, \mathbf{x}_2$ and $\mathbf{x}_3$ denotes the vertices of a triangle in $\mathbf{I}$ any internal point can be written as superposition:

$$\begin{aligned} \mathbf{x} &= \mathbf{x}_1 + \beta(\mathbf{x}_2 - \mathbf{x}_1) + \gamma(\mathbf{x}_3 - \mathbf{x}_1) \\ &= \alpha\mathbf{x}_1 + \beta\mathbf{x}_2 + \gamma\mathbf{x}_3 \end{aligned} \tag{5.3}$$

Thus $\alpha = 1 - (\beta + \gamma)$ giving $\alpha + \beta + \gamma = 1$. To constrain $\mathbf{x}$ inside the triangle we must have $0 \leq \alpha, \beta, \gamma \leq 1$. Warping is now given by using the relative position within the triangle given by α, β and γ on the triangle in $\mathbf{I}'$:

$$\mathbf{x}' = \mathbf{f}(\mathbf{x}) = \alpha \mathbf{x}'_1 + \beta \mathbf{x}'_2 + \gamma \mathbf{x}'_3 \tag{5.4}$$

Given the three points of a triangle it is trivial to determine α, β and γ by solving the system of the two linear equations given by (5.3) for a known point, $\mathbf{x} = [x, y]^{\mathrm{T}}$:

$$\begin{aligned} \alpha &= 1 - (\beta + \gamma) \\ \beta &= \frac{yx_3 - x_1 y - x_3 y_1 - y_3 x + x_1 y_3 + x y_1}{-x_2 y_3 + x_2 y_1 + x_1 y_3 + x_3 y_2 - x_3 y_1 - x_1 y_2} \\ \gamma &= \frac{x y_2 - x y_1 - x_1 y_2 - x_2 y + x_2 y_1 + x_1 y}{-x_2 y_3 + x_2 y_1 + x_1 y_3 + x_3 y_2 - x_3 y_1 - x_1 y_2} \end{aligned} \tag{5.5}$$

With implementation in mind, one should notice the common denominator. In pseudo-code the piece-wise affine warp can then be written as:

1. For each pixel $\mathbf{x} = [x, y]^{\mathrm{T}}$ inside the convex hull of $\{\mathbf{x}'_1 \ \ldots \ \mathbf{x}'_n\}$
2. Determine the triangle, t, that $\mathbf{x}$ belongs to
3. Find the reative postion of $\mathbf{x}$ given by (5.5) inside t
4. Use (5.4) to obtain the position inside t'
5. Set $\mathbf{I}'(\mathbf{x}) = \mathbf{I}(\mathbf{f}(\mathbf{x}))$
6. End

The naive solution to step 2 is to run through all triangles until $0 \leq \alpha, \beta, \gamma \leq 1$. A small enhancement to this is to perform a relative quick bounding box test beforehand. An ad hoc speed-up of this step would be to test the triangle from the previous point first. However, one needs not to rely on ad hoc methods since the problems of searching efficiently in spatial data has been of great concern with the community of computational geometry. The binary search tree exist for 1D data, and for planar data *quadtrees* [32] can be used. As an alternative the generalization of binary trees, the so-called *k-d trees* [55] can be used. Other tree structure

for spatially searching / distance measuring includes *octrees* and *binary space-partitioning trees* (BSP-trees), both 3D [32].

To conclude this section on piece-wise affine warping, notice that even $\mathbf{f}$ produces a continuous deformation, the deformation field is not smooth. This is reflected in figure 5.4. To overcome this limitation *thin plate splines* [5] could be used instead, since they guarantee a smooth deformation field. The drawback to this is an increase in the number of calculations per warp.

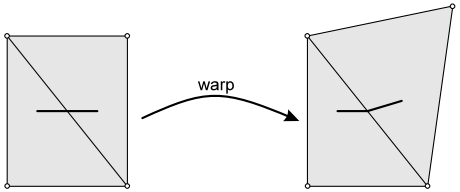


Figure 5.4: Problem of the piece-wise affine warping. Straight lines will usually be kinked across triangle boundaries.

Another flaw of piece-wise affine warping is that it will not detect singularities in the deformation field in the form of folding triangles. This can however be detected by a simple test on the triangle face normals.

5.3.2 Pixel Interpolation

Since equation (5.4) will inherently not produce positions on the integer pixel lattice of $\mathbf{I}$ some sort of pixel interpolation scheme is needed.

The traditional solution to this is to use bilinear interpolation, which consists of two consecutive linear interpolations using the four neighboring pixels. The graphical interpretation of this is given in figure in 5.5. Using this notation, the bilinear interpolation can be written as:

$$\varepsilon = a(b\gamma + (1-b)\varphi) + (1-a)(b\beta + (1-b)\alpha) \tag{5.6}$$

As alternative a second order approximation using bicubic interpolation could be used, involving 16 neighboring pixels. The current implementation uses the computational more feasible first order approximation.

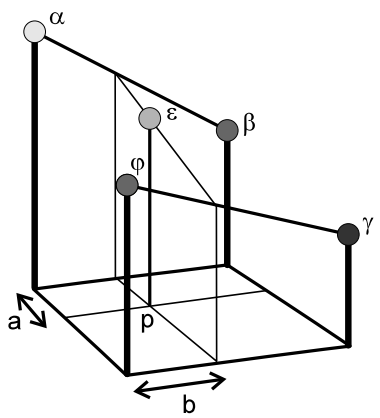


Figure 5.5: Bilinear interpolation. The intensity at ε is interpolated from the four neighboring pixels, α, β, γ and φ.

5.4 Acquiring Texture in Practice

Though the previous sections provided a firm basis for acquiring texture, some important observations still needs to be underlined. These should increase understanding in general and in specific ease the implementation and execution substantially.

All texture handling in AAMs happens in the normalized frame of the reference shape.[3] As mentioned earlier the mean shape is a good choice of such. This means that all warps in the shape modeling part (and in the later optimization) have a common source shape to sample in. Only the destination (the actual training shape) shape changes. Hence, substantially parts of the warp-calculations can be avoided using dynamic programming. This means that step 2 and 3 of the piece-wise warp pseudo code can be cached. This is done in the current implementation.

The second observation is useful in the optimization part. Tests has shown that in the current implementation approx. 30% of the total optimization time was spent doing bilinear interpolation. However, since the image being interpolated never changes during the optimization, these calculations can

[3] An exception to this, is when one wants to back-project the AAM texture onto the image, which is a pretty tedious task requiring two warps (and great care).

be cached if a certain discreet representation of the interpolation is allowed. If a resolution of $n \times n$ inside each pixel is chosen, this is accomplished by allocating a new image, $\mathbf{I}_{bip}$, of n times the size of the original image, $\mathbf{I}$. Then $\mathbf{I}$ is interpolated into $\mathbf{I}_{bip}$. Here forth bilinear interpolation in $\mathbf{I}$ is performed by lookup's in $\mathbf{I}_{bip}$ – i.e.:

$$\mathbf{I}(x, y) = \mathbf{I}_{bip}(\text{Round}(xn), \text{Round}(yn)) \tag{5.7}$$

5.5 Photometric Normalization

As we previous filtered out the pose from the object to obtain the true shape, one would like the texture model to be invariant to global changes in illumination. Effects that cause such changes include usage of different film media, different exposure times, external lightning or shadows etc.

Below we will compensate for linear changes by applying a scaling of α and an offset of β. If $\mathbf{g}_{image}$ denotes the actual pixel values sampled in the image:

$$\mathbf{g}_{norm} = \frac{\mathbf{g}_{image} - \beta \mathbf{1}}{\alpha} \tag{5.8}$$

In practice each texture vector of m pixels is aligned to the standardized mean texture, $\overline{\mathbf{g}}$, by offsetting it to zero mean:

$$\overline{\mathbf{g}}_{\mathbf{zm}} = \overline{\mathbf{g}} - \overline{g}\mathbf{1} \quad , \quad \overline{g} = \frac{1}{m}\sum_{i=1}^{m} g_i = \frac{1}{m}\,\overline{\mathbf{g}} \cdot \mathbf{1} \tag{5.9}$$

and scale it to unit variance:

$$\overline{\mathbf{g}} = \frac{1}{\sigma}\overline{\mathbf{g}}_{\mathbf{zm}} \quad , \quad \sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(g_i - \overline{g})^2 \tag{5.10}$$

Notice that the variance estimate simplifies to $\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} g_i^2$ since $\overline{\mathbf{g}}_{\mathbf{zm}}$ has a zero mean. α and β can thus be written as:

$$\alpha = \mathbf{g}_{image} \cdot \overline{\mathbf{g}} \tag{5.11}$$

$$\beta = \frac{\mathbf{g}_{image} \cdot \mathbf{1}}{m} \tag{5.12}$$

Since α is defined in terms of the mean, an iterative approach must be taken. In pseudo code this is:

1. Do
2. Estimate mean of all texture vectors, $\hat{\overline{\mathbf{g}}}$
3. Standardize $\hat{\overline{\mathbf{g}}}$
4. For each texture vector, $\mathbf{g}_{image}$
5. $\alpha = \mathbf{g}_{image} \cdot \hat{\overline{\mathbf{g}}}$
6. $\beta = (\mathbf{g}_{image} \cdot \mathbf{1})/m$
7. Normalize $\mathbf{g}_{image}$ using (5.8)
8. End
9. Until $\overline{\mathbf{g}}$ is stable

Alternative to the above is the approach used in Active Blobs [58], where two bilinear functions of x and y were used to obtain α and β thus providing a locally photometric compensation.

5.6 Modelling Texture Variation

As a starting point for the texture variation modeling the term *digital image* needs a discussion. The core of digital images is a set of spatially samples. In the thesis – and almost everywhere else – raster images is considered. This merely means that the samples are arranged in a uniformly spaced spatially grid. In the term spatially lies that the ordering of the samples becomes crucial. This suggests some sort of correlation between samples leading to data redundancy as in the case of shapes. Hence, it is natural to adapt the PCA approach for the texture variation also. Refer to the shape section for a more detailed description of the PCA.

The maximum-likelihood (ML) estimate of the mean texture of N normalized texture vectors is given as:

$$\overline{\mathbf{g}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{g}_i \tag{5.13}$$

The maximum-likelihood (ML) estimate of the covariance matrix can then be written as:

$$\mathbf{\Sigma}_g = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{g}_i - \overline{\mathbf{g}})(\mathbf{g}_i - \overline{\mathbf{g}})^{\mathrm{T}} \tag{5.14}$$

The principal axes of the m^{th} dimensional point cloud of textures are now given as the eigenvectors, $\mathbf{\Phi}_g$, of the covariance matrix.

$$\mathbf{\Sigma}_g\mathbf{\Phi}_g = \mathbf{\Phi}_g\mathbf{\Lambda}_g \tag{5.15}$$

Where $\mathbf{\Lambda}_g$ is a diagonal matrix of eigenvalues. A texture instance can then be generated by deforming the mean texture by a linear combination of eigenvectors:

$$\mathbf{g} = \overline{\mathbf{g}} + \mathbf{\Phi}_g\mathbf{b}_g \tag{5.16}$$

Where $\mathbf{b}_g$ are the texture model parameters. To further investigate the distribution of the $\mathbf{b}_g$-parameters in the metacarpal training set $b_{s,2}$ is plotted as a function of $b_{s,1}$ in figure 5.6.

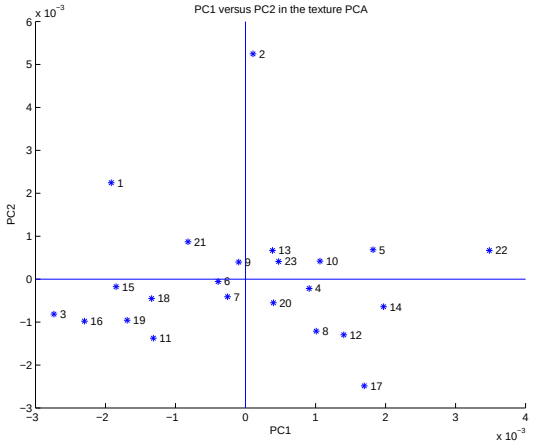


Figure 5.6: PC1 ($b_{g,1}$) versus PC2 ($b_{g,2}$) in the texture PCA.

These are easily obtained due to the linear structure of (5.16) and since the columns of $\mathbf{\Phi}_g$ are inherently orthogonal.

$$\mathbf{b}_g = \mathbf{\Phi}_g^{-1}(\mathbf{g} - \overline{\mathbf{g}}) = \mathbf{\Phi}_g^{\mathrm{T}}(\mathbf{g} - \overline{\mathbf{g}}) \tag{5.17}$$

To give an idea of the decay rate of the eigenvalues a percentage plot is shown in figure 5.7.

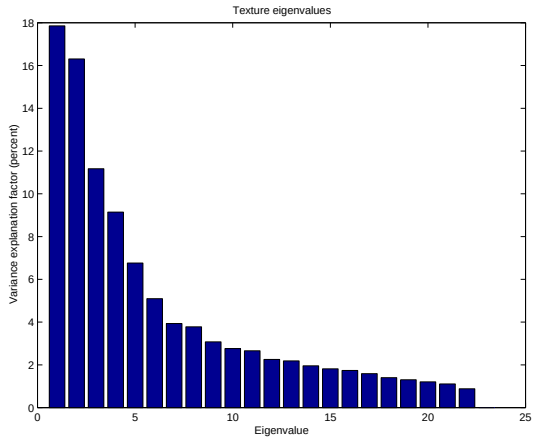


Figure 5.7: Texture eigenvalues in descending order.

To conclude this section remember the point regarding *spatially* correlation in digital images. PCA treats observations as vectors and thus misses this important feature. It is therefore suggested using orthogonal transformations taking the spatial nature of the data into account, such as the *Min/Max Autocorrelation Factors* (MAFs) [54] etc. This topic is somewhat outside the scope of this master thesis, but pose an interesting topic for further improvements to AAMs.

5.6.1 Reduction of Dimensions in the PCA

Due to size of the texture vectors – and the workload in manual annotation – it is safe to assume that there always is more dimensions (pixels) in the

samples than observations (training objects) in the texture case.[4]

This leads to rank deficiency of the covariance matrix $\mathbf{\Sigma}_g$ thus enabling use of the following efficient computation of the eigenanalysis.

First, remove the mean from the N texture vectors:

$$\mathbf{G} = \begin{bmatrix} & & \\ (\mathbf{g}_1 - \overline{\mathbf{g}}) & \cdots & (\mathbf{g}_N - \overline{\mathbf{g}}) \\ & & \end{bmatrix} \tag{5.18}$$

In matrix notation the (rather large) covariance matrix of $\mathbf{G}$ can thus written as:

$$\Sigma_l = \frac{1}{s}\mathbf{G}\mathbf{G}^{\mathrm{T}} \tag{5.19}$$

Consider instead the smaller matrix:

$$\Sigma_s = \frac{1}{s}\mathbf{G}^{\mathrm{T}}\mathbf{G} \tag{5.20}$$

It can then be shown that the non-zero eigenvalues of the matrices are equal:

$$\mathbf{\Lambda}_l = \mathbf{\Lambda}_\mathbf{s} \tag{5.21}$$

Let $\mathbf{\Phi}_l$ be given as:

$$\mathbf{\Phi}_l = \mathbf{G}\mathbf{\Phi}_s \tag{5.22}$$

Where $\mathbf{\Phi}_s$ is the eigenvectors of Σ_s. Finally normalize the columns of $\mathbf{\Phi}_l$ by letting $\varphi_{l,i}$ denote the i^{th} column of $\mathbf{\Phi}_l$:

[4] This is usually also the case for shape models, but not always. Consider the training set of Cootes et al. [14] using 400 faces each annotated using 122 landmarks which equals a dimensionality of 244 and 400 observations.

$$\varphi_{l,i} = \frac{1}{\sqrt{\lambda_{l,i}}}\varphi_{l,i} \tag{5.23}$$

Now $\mathbf{\Phi}_l$ hold the eigenvectors of the texture samples. This is proved using the Eckart-Young Theorem. Consult a textbook in statistics for the proof.

Notice that this gives a substantial speed up since the eigenvector decomposition goes as the cube of the size of covariance matrix [14]. Further more it results in a considerable reduction of the memory requirements during the eigenanalysis.

We stress that without this method only very small texture models would be feasible.

5.7 Summary

Throughout this chapter, a consistent method for sampling the intensities across an object (texture) has been given using image warping. To model the texture variation the use of a PCA has been described while giving usage examples from the metacarpal case. Furthermore, it has been demonstrated how the computational load of the texture PCA can be eased substantially.

Chapter 6

Combined Model Formulation

6.1 Overview

How to unify the presented shape and texture models into one complete compact *appearance model* is the topic of this chapter. It is also shown how this model representation can be regularized and further compressed by truncation of the model parameters (eigenmodes).

6.2 Combining Models of Shape and Texture

From the previous chapter it was seen that an object instance can be constructed using the two set of model parameters of shape, $\mathbf{b}_s$, and texture, $\mathbf{b}_g$.

To remove correlation between shape and texture model parameters – and to make the model representation more compact – a 3rd PCA is performed on the concatenated shape and texture parameters, $\mathbf{b}$, of the training set to obtain the combined model parameters, $\mathbf{c}$:

$$\mathbf{b} = \mathbf{\Phi}_c \mathbf{c} \tag{6.1}$$

where $\mathbf{\Phi}_c$ denotes a set of eigenvectors. The concatenated shape and texture parameters are easily obtained due to the linear nature of the model:

$$\mathbf{b} = \begin{pmatrix} \mathbf{W}_s \mathbf{b}_s \\ \mathbf{b}_g \end{pmatrix} = \begin{pmatrix} \mathbf{W}_s \mathbf{\Phi}_s^{\mathrm{T}} (\mathbf{x} - \overline{\mathbf{x}}) \\ \mathbf{\Phi}_g^{\mathrm{T}} (\mathbf{g} - \overline{\mathbf{g}}) \end{pmatrix} \tag{6.2}$$

Notice that a suitable weighting between pixel distances and pixel intensities is done through the diagonal matrix $\mathbf{W}_s$. How to obtain $\mathbf{W}_s$ is the topic of the next section.

Now – using simple linear algebra – a complete model instance including shape, $\mathbf{x}$ and texture, $\mathbf{g}$, can be generated using the model parameters, $\mathbf{c}$.

$$\mathbf{x} = \overline{\mathbf{x}} + \mathbf{\Phi}_s \mathbf{W}_s^{-1} \mathbf{\Phi}_{c,s} \mathbf{c} \tag{6.3}$$

$$\mathbf{g} = \overline{\mathbf{g}} + \mathbf{\Phi}_g \mathbf{\Phi}_{c,g} \mathbf{c} \tag{6.4}$$

Where

$$\mathbf{\Phi}_c = \begin{pmatrix} \mathbf{\Phi}_{c,s} \\ \mathbf{\Phi}_{c,g} \end{pmatrix} \tag{6.5}$$

Combined modal deformations of the metacarpal from, the first three – most significant – eigenvectors can be seen in figure 6.1.

Regarding the compression of the model parameters one should notice that the rank of $\mathbf{\Phi}_c$ will never exceed the number of examples in the training set.

Observe that another feasible method to obtain the combined model is to concatenate both shape points and texture samples into one observation vector from the start and then perform PCA on the correlation matrix of these observations.

We regard the reason for the two separate PCAs as being partly historical. Active Appearance Models is the direct continuation of the work with Active Shape Models (ASM) by Cootes et al. Basic ASMs did not include an explicit texture model but modeled the shape variation only – using an identical approach to that of AAMs – i.e. PCA. Another motivation is the need to discriminate object instances purely based upon the shape or texture characteristics alone. An example of this is the work of Edwards et al. [24] where an AAM model was utilized to identify faces.

Figure 6.1: Three largest combined metacarpal modes from top to bottom; $c_i = -3\sqrt{\lambda_i}$, $c_i = 0$, $c_i = 3\sqrt{\lambda_i}$.

6.2.1 Comparing Pixel-distances and Intensity

Since the shape parameters, $\mathbf{b}_s$, has units of pixel *distance* and the texture parameters has units of pixel *intensity* they will obviously not commensurate without the weighting of $\mathbf{W}_s$.

A simple method to estimate $\mathbf{W}_s$ devised in [14] is to weight uniformly with the ratio, r, of the total variance in shape and texture as seen in the training set. Remembering that the variance of parameter b_i equals λ_i we have:

$$\mathbf{W}_s = r\mathbf{I} = \begin{bmatrix} r & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & r \end{bmatrix} \tag{6.6}$$

$$r = \frac{\lambda_g}{\lambda_s} \quad , \quad \lambda_g = \sum \lambda_{g_i} \quad , \quad \lambda_s = \sum \lambda_{s_i} \tag{6.7}$$

An alternative, is to do the shape and texture PCAs based on the correlation matrix as opposed to the covariance matrix.

6.3 Choosing Modes of Variation

As in the shape PCA case, we can compress the combined model representation further by removing the smallest eigenmodes. Again, it is safe to consider small-scale variation as noise. Thus to retain p percent of the combined variation in the training set, t modes can be chosen satisfying:

$$\sum_{i=1}^{t} \lambda_i \geq \frac{p}{100} \sum_{i=1}^{2n} \lambda_i \tag{6.8}$$

In the metacarpal case 95% of the combined variation can be modeled using 18 parameters. This is a rather substantial reduction since the original space had a dimensionality of 50 points $\times$ 2+ $\sim$ 9400 pixels $\approx$ 9500. The decay rate of the combined eigenvalues is given in figure 6.2.

Again, we stress that this step basically is a regularization of the solution space.

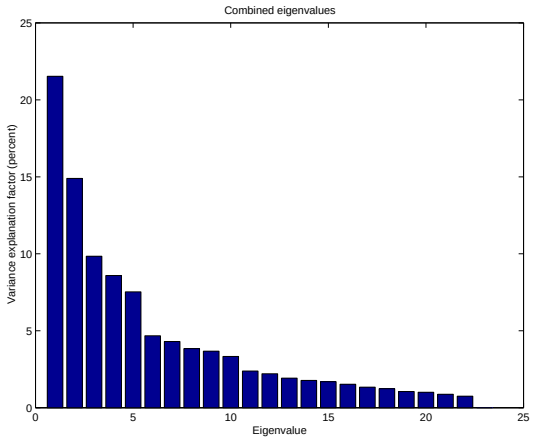


Figure 6.2: Combined eigenvalues.

6.4 Summary

In this chapter, a unified appearance model of shape and texture has been described.

Examples have been given that this model can generate near photo-realistic images of the object class that it represents. Using only a few parameters this model can be deformed w.r.t. shape and texture in a manner similar to what is observed in a training set.

Part II

Active Appearance Models

Chapter 7

Basic Active Appearance Models

One of the pleasant properties of the statistical models of shape and texture – as presented in the previous chapters – is that it is possible to use these to search images for new instances of the class of objects that they represent. Further more this can be done in a fast and robust manner. This is called the Active Appearance Models.

In the following, we will emphasize on the customized search algorithm of AAMs in such detail that an implementation can be understood and/or made solely upon this description.

The foundation of the AAM search is to treat the search as an optimization problem in which the difference between the synthesized object delivered by the AAM and an actual image is to be minimized. Formally this can be written as the difference vector $\delta\mathbf{I}$:

$$\delta\mathbf{I} = \mathbf{I}_{image} - \mathbf{I}_{model} \tag{7.1}$$

In this way the fit can be enhanced by adjusting the model and pose parameters to fit the image in the best possible way. Since the following will be based on normalized texture vectors, $\delta\mathbf{I}$ will be denoted as $\delta\mathbf{g}$.

Though we have seen that the parameterization of the object class in question can be compacted tremendously by the principal component analysis it is far from an easy task to optimize a system of the dimensionalities that we have seen in the earlier chapters. This is not only computationally cumbersome but also theoretically challenging – optimization theory-wise – since we are in no way guaranteed that the hyperspace sought in is smooth.

Methods of solving the optimization problem of deformable models include general optimization techniques such as gradient based methods like steepest descent and Marquardt-Levenberg [58], random sampling methods like simulated annealing [29] and genetic algorithms [39, 37].

AAMs circumvent these potential problems in a rather untraditional fashion. The key observation is that each model search constitutes what we call a *prototype search* – the search path and the optimal model parameters will be unique in each search where the initial and final model configuration matches this configuration. Or as expressed by Cootes et al. ([14] pp. 43, l. 6):

> *"We note, however, that each attempt to match the model to a new image is actually a similar optimization problem. We propose to learn something about how to solve this class of problems in advance."*

All this is of course within certain limitations – for example background handling – of which we will get back to.

These prototype searches can then be made at model building time; thus saving computationally expensive high-dimensional optimization. Below is described how to collect these prototype searches and how to utilize them into a run-time efficient model search of an image.

7.1 Solving Parameter Optimization Off-line

It is proposed that the spatial pattern in $\delta\mathbf{g}$ can predict the needed adjustments in the model and pose parameters to minimize $\delta\mathbf{g}$. The simplest model we can arrive at constitutes a linear relationship:

$$\delta\mathbf{c} = \mathbf{R}\delta\mathbf{g} \tag{7.2}$$

Cootes et al. show that this crude approximation suffices to produce good results in their work with AAMs [10, 14, 17, 22, 9]. Sclaroff & Isidoro have also had success in using the very similar *difference decomposition* approach in their work with *Active Blobs* and *Active Voodoo Dolls* [43]. The difference decomposition was originally proposed by Gleicher [34].

To determine a suitable $\mathbf{R}$ in equation (7.2) a set of experiments are conducted which is fed into a multivariate linear regression framework. The multivariate linear regression is described in detail in the next section.

Each experiment consists of displacing a set of ground truth parameters by a known amount and measuring the difference between the model and the part of the image below the model.

The parameters considered here is the model parameters $\mathbf{c}$ and the pose parameters $\mathbf{t}$. To ensure linearity and equilibrium at zero we represent pose as:

$$t = (s_x, s_y, t_x, t_x)^{\mathrm{T}} \tag{7.3}$$

where the mapping from the usual pose parameters (t_x, t_y, s, θ) are:

s_x : combined scaling and rotation: $s_x = s\cos(\theta) - 1$
s_y : combined scaling and rotation: $s_y = s\sin(\theta)$
t_x : translation in the x direction
t_y : translation in the y direction

In pseudo-code the j^{th} experiment can then be expressed as:

1. Displace a model by a known amount in (either) the pose parameters $\mathbf{t}$ and(or) the model parameters $\mathbf{c}$.
2. Update the current model parameters: $\mathbf{c} = \delta\mathbf{c} + \mathbf{c}_0$
3. Update the current pose to $\mathbf{t}$, that is: first by $\delta\mathbf{t}$ then by $\mathbf{t}_0$
4. Generate a new model instance by generation of a new texture $\mathbf{g}_m$ and a new shape $\mathbf{x}$
5. Obtain the shape in image coordinates, $\mathbf{x}_{image}$, by aligning the shape $\mathbf{x}$ to fit the pose given in $\mathbf{t}$
6. Sample the image below $\mathbf{x}_{image}$ into the texture vector $\mathbf{g}_{image}$.
7. Normalize the texture samples in $\mathbf{g}_{image}$ to obtain $\mathbf{g}_i$.
8. Form the normalized texture difference $\delta\mathbf{g} = \mathbf{g}_i - \mathbf{g}_m$
9. Write $\delta\mathbf{t}$ and $\delta\mathbf{c}$ into the experiment matrices $\mathbf{T}$ and $\mathbf{C}$ – each in the j^{th} column.

10. Write $\delta\mathbf{g}$ into the j^{th} column of $\mathbf{G}$

Based on the matrices $\mathbf{T}$, $\mathbf{C}$, $\mathbf{G}$ and the relations:

$$\mathbf{C} = \mathbf{R}_c\mathbf{G} \quad , \quad \mathbf{T} = \mathbf{R}_t\mathbf{G} \tag{7.4}$$

multivariate regression will then build the relationships between pixel differences and pose/model parameter displacements:

$$\delta\mathbf{c} = \mathbf{R}_c\delta\mathbf{g} \tag{7.5}$$

$$\delta\mathbf{t} = \mathbf{R}_t\delta\mathbf{g} \tag{7.6}$$

Though the above description seems beautiful in all its simplicity – several crucial questions remain unanswered – such as:

- How many displacements should we use?
- How large should the displacements be?
- Should all parameters be displaced at once or separately?
- Should the displacements be done in a deterministic or random fashion?
- Do we tie the model to the background seen in the training set?

Regarding the first and second matter, a vague guideline would be that there should be enough displacements to span the parameter space in which the linearity assumption holds true. This assures that the sensitivity towards good initialization is minimized since this is also maximizing the ranges from which the model can safely predict its deformations[1].

Cootes et al. [14] suggest that the optimum perturbation of the model parameters is within 0.5 std. dev. over the training set. Pose parameters were about 10% in scale, ± 3 pixel in x and y. Nothing is noted about the perturbation in the rotation parameter.

In the current work, the training scheme below has been used with great success.

[1] This is shape/texture changes in the model parameter case and position, scale and orientation in the pose case.

Displacement Training Scheme

- Pose parameter displacements

  t_x: $\pm 6, \pm 3, \pm 1$ (pixels)

  t_y: $\pm 6, \pm 3, \pm 1$ (pixels)

  s: 95%, 97%, 99%, 101%, 103%, 105%

  θ: $\pm 5, \pm 3, \pm 1$ (degrees)

- Model parameter displacements

  $\pm 0.5\sigma_i$, $\pm 0.25\sigma_i$ for each parameter over the training set

Notice the concentration of experiments around zero and 100% for the pose parameters. This is done to give the linear regression more weight towards equilibrium, thus potentially leading to a more accurate fit in the optimization.

Each parameter was displaced separately and the corresponding texture difference was measured. This leads to a total number of displacements, n, of:

$$n = m(4k + 24) \tag{7.7}$$

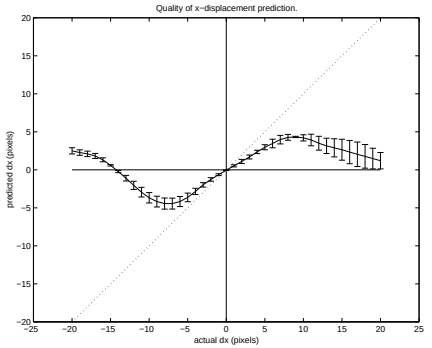
Where m denotes the number of examples in the training set and k the number of parameters in the model.

•

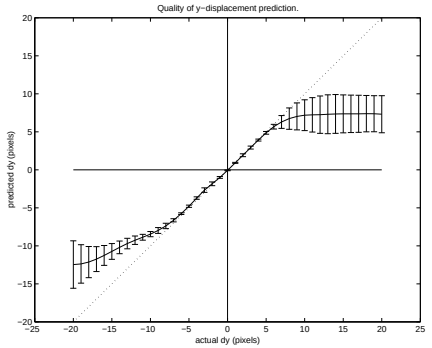
As an evaluation of the assumption of a linear relationship between the model and pose parameters and the observed texture differences fig. 7.1 shows the actual and the predicted displacement from a number of displacements. The error bars correspond to one standard deviation. From these it is seen that an iterative scheme using this regression matrix should converge within a limited range, even when the prediction diverges from the line $y = x$. As long as it preserves the right sign it will converge.

The nature of this figure points in the direction that the linear assumption can be used with success – though this type of performance assessment

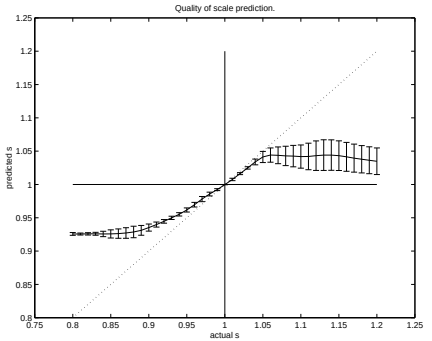
is limited in its use. Partly because this type of figures only evaluates the ability of the regression to predict displacement in one parameter at a time – implicitly assuming that all other parameters are at their equilibrium/optimum. Almost a utopia in real-life examples. Similar plots have been made for **c**-parameters displacements which also show a linear relationship within a limited range.



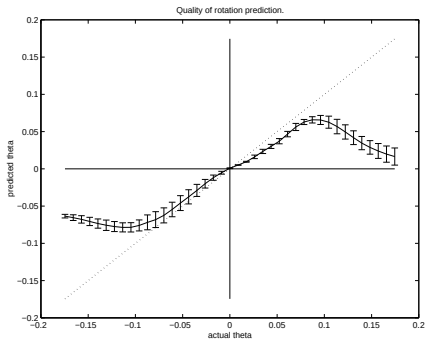
(a) X-parameter displacement.



(b) Y-parameter displacement.



(c) Scale-parameter displacement.



(d) Theta-parameter displacement.

Figure 7.1: Displacement plots for a series of model predictions versus the actual displacement. Error bars are equal to 1 std.dev.

7.1.1 Details on Multivariate Linear Regression

This section focus on the details of the multivariate linear regression used in AAMs. The method described below is by no means unique to the AAM approach and can thus be read by any with interest in prediction using experiments based on assumptions of linear behavior.

By necessity, this treatment is somewhat technical. The hasty reader may – without loss of continuity – proceed to section 7.2. The notation below roughly follows [14].

The basic idea behind multivariate linear regression is to assume linear dependency between two sets of variables. In the AAM case we assume that there is a linear relation between the measured normalized texture differences, $\delta\mathbf{g}$, stored in the matrix $\mathbf{G}$, and a set of model displacement parameters[2] stored in the matrix $\mathbf{V}$.

To determine the assumed linear relationship a series of s experiments are conducted. In each experiment the t model parameters are displaced by known amounts and the corresponding m pixel differences[3] is measured as mentioned in the previous section.

Expressed as matrix algebra this looks like:

$$\begin{bmatrix} \begin{bmatrix} v_{1_1} \\ \vdots \\ v_{1_t} \end{bmatrix} & \cdots & \begin{bmatrix} v_{s_1} \\ \vdots \\ v_{s_t} \end{bmatrix} \end{bmatrix} = \mathbf{R} \begin{bmatrix} \begin{bmatrix} g_{1_1} \\ \\ \vdots \\ \\ g_{1_m} \end{bmatrix} & \cdots & \begin{bmatrix} g_{s_1} \\ \\ \vdots \\ \\ g_{s_m} \end{bmatrix} \end{bmatrix} \tag{7.8}$$

Or in the short form:

$$\mathbf{V} = \mathbf{RG} \tag{7.9}$$

Hence $\mathbf{V}$ is a $t \times s$ matrix, $\mathbf{R}$ is a $t \times m$ matrix and finally $\mathbf{G}$: an $m \times s$ matrix.

[2] In this case it's the displacement of the $\mathbf{c}$-model parameters, $\delta\mathbf{c}$, and the pose displacement parameters, $\delta\mathbf{t}$.

[3] Thus m equals the number of texture samples in the model.

In the case of AAMs it should be safe to assume that $m \gg s > t$ which makes (7.9) with $\mathbf{R}$ as the unknown; a hopelessly underdetermined system.

A feasible path towards solving (7.9) can luckily be obtained by projecting the rather large $\mathbf{G}$-matrix into a k-dimensional subspace[4] which captures the major part of the variation in $\mathbf{G}$. This is called *principal component regression* or *reduced rank multivariate linear regression*. For more details on this topic refer to [57].

Let the k eigenvectors of the texture difference product $\mathbf{G}^{\mathrm{T}}\mathbf{G}$ corresponding to the k largest eigenvalues $\{\lambda_1 \dots \lambda_k\}$ be denoted Φ_k and let Λ_k be equal to $diag(\lambda_1 \dots \lambda_k)$. Thus the following identity holds true:

$$\mathbf{G}^{\mathrm{T}}\mathbf{G}\Phi_k = \Phi_k\Lambda_k \tag{7.10}$$

It can now be seen that the transpose of the eigenvectors is a projection of the columns of $\mathbf{G}$ into a k-dimensional subspace:

$$\mathbf{B}_k = \Phi_k^{\mathrm{T}} = (\Lambda_k^{-1}\Phi_k^{\mathrm{T}}\mathbf{G}^{\mathrm{T}})\mathbf{G} \tag{7.11}$$

Since the rows of $\mathbf{B}_k$ are orthogonal another property is fulfilled:

$$\mathbf{B}_k\mathbf{B}_k^{\mathrm{T}} = \mathbf{I} \tag{7.12}$$

Consider then an equation similar to (7.9) using an identical $\mathbf{V}$:

$$\begin{aligned} \mathbf{V} &= \mathbf{R}'_k\mathbf{B}_k \quad \Rightarrow \\ \mathbf{R}_k\mathbf{G} &= \mathbf{R}'_k\mathbf{B}_k \end{aligned} \tag{7.13}$$

If $\mathbf{R}'_k$ is a solution to (7.13), then using (7.11) yields:

$$\mathbf{R}_k = \mathbf{R}'_k(\Lambda_k^{-1}\Phi_k^{\mathrm{T}}\mathbf{G}^{\mathrm{T}}) \tag{7.14}$$

Using (7.12) the equation (7.13) can be rearranged to:

[4] Where $k \geq t$ since choosing a $k < t$ would not make $\mathbf{R}$ span the whole space of $\mathbf{V}$.

$$\mathbf{R}'_k = \mathbf{V}\mathbf{B}_k^{\mathrm{T}} \tag{7.15}$$

Then by applying (7.14) $\mathbf{R}_k$ is given as:

$$\mathbf{R}_k = \mathbf{V}\mathbf{B}_k^{\mathrm{T}}(\Lambda_k^{-1}\Phi_k^{\mathrm{T}}\mathbf{G}^{\mathrm{T}}) \tag{7.16}$$

What remains now is to determine the value of k. Since $\mathbf{G}^{\mathrm{T}}\mathbf{G}$ is an $s \times s$ matrix, and $\mathbf{R}$ must span the whole space of $\mathbf{V}$ the limits on k is $t \leq k \leq s$.

Basically the k-estimation can be viewed upon as a matter of determine when the linear regression is fitting to noise. Hence, we will use a leave-one-out approach.

Optimal choice of k

We shall now devise a way to determine k based on [14] with further comments and calculations.

The *brute force* way is to leave out one column of $\mathbf{V}$ – and the corresponding column in $\mathbf{G}$ – estimate $\mathbf{R}_k$ and use this to measure the error in predicting the error in estimating $\mathbf{v}_j$ from $\mathbf{x}_j$. However – due to the often large amount of experiments conducted[5] – this can be a rather infeasible approach computationally-wise.

An efficient way to circumvent this, is instead to leave out each column of $\mathbf{B}_k$ in turn. If $\mathbf{B}_k$ with its j^{th} column missing is denoted $\mathbf{B}_{k/j}$ – size $k \times (s-1)$ – the following is given (partly due to (7.12)):

$$\begin{aligned} \mathbf{B}_{k/j}\mathbf{B}_{k/j}^{\mathrm{T}} &= \mathbf{B}_k\mathbf{B}_k^{\mathrm{T}} - \mathbf{b}_{kj}\mathbf{b}_{kj}^{\mathrm{T}} \\ &= \mathbf{I} - \mathbf{b}_{kj}\mathbf{b}_{kj}^{\mathrm{T}} \end{aligned} \tag{7.17}$$

It can then be shown that [14]:

$$(\mathbf{I} - \mathbf{b}_{kj}^{\mathrm{T}}\mathbf{b}_{kj})^{-1} = (\mathbf{I} + f_{kj}\mathbf{b}_{kj}^{\mathrm{T}}\mathbf{b}_{kj}) \tag{7.18}$$

[5] Obviously since a large number of experiments, constitutes a better prediction model.

where f_{kj} is given as:

$$f_{kj} = \begin{cases} \frac{1}{1-\alpha_{kj}} & |1-\alpha_{kj}| \geq \epsilon \\ 0 & |1-\alpha_{kj}| < \epsilon \end{cases} \tag{7.19}$$

and $\alpha_{kj} = |\mathbf{b}_{kj}|^2$. The constant ϵ is a precision dependent value to avoid singularities (i.e. numerical unstability).

Similar to $\mathbf{B}_{k/j}$ we denote $\mathbf{V}$ with its j^{th} column missing; $\mathbf{V}_{/j}$. If then $\mathbf{R}'_{k/j}$ is a solution to:

$$\mathbf{V}_{/j} = \mathbf{R}'_{k/j}\mathbf{B}_{k/j} \tag{7.20}$$

Then – by using (7.17) and (7.18) – (7.20) yields:

$$\begin{aligned} \mathbf{R}'_{k/j}\mathbf{B}_{k/j} &= \mathbf{V}_{/j} \\ \mathbf{R}'_{k/j}\mathbf{B}_{k/j}\mathbf{B}_{k/j}^{\mathrm{T}} &= \mathbf{V}_{/j}\mathbf{B}_{k/j}^{\mathrm{T}} \\ \mathbf{R}'_{k/j} &= \mathbf{V}_{/j}\mathbf{B}_{k/j}^{\mathrm{T}}(\mathbf{B}_{k/j}\mathbf{B}_{k/j}^{\mathrm{T}})^{-1} \\ \mathbf{R}'_{k/j} &= \mathbf{V}_{/j}\mathbf{B}_{k/j}^{\mathrm{T}}(\mathbf{I} - \mathbf{b}_{kj}\mathbf{b}_{kj}^{\mathrm{T}})^{-1} \\ \mathbf{R}'_{k/j} &= \mathbf{V}_{/j}\mathbf{B}_{k/j}^{\mathrm{T}}(\mathbf{I} + f_{kj}\mathbf{b}_{kj}\mathbf{b}_{kj}^{\mathrm{T}}) \end{aligned} \tag{7.21}$$

Then to estimate $\mathbf{v}_j$ from $\mathbf{b}_{kj}$ the error is given by

$$\delta\mathbf{v}_{kj} = \mathbf{R}'_{k/j}\mathbf{b}_{kj} - \mathbf{v}_j \tag{7.22}$$

By insertion of (7.21) the error is given by:

$$\begin{aligned} \delta\mathbf{v}_{kj} &= \mathbf{V}_{/j}\mathbf{B}_{k/j}^{\mathrm{T}}(\mathbf{I} + f_{kj}\mathbf{b}_{kj}\mathbf{b}_{kj}^{\mathrm{T}})\mathbf{b}_{kj} - \mathbf{v}_j \\ &= \mathbf{V}_{/j}\mathbf{B}_{k/j}^{\mathrm{T}}(\mathbf{b}_{kj} + f_{kj}\mathbf{b}_{kj}\alpha_{kj}) - \mathbf{v}_j \\ &= f_{kj}\mathbf{V}_{/j}\mathbf{B}_{k/j}^{\mathrm{T}}\mathbf{b}_{kj} - \mathbf{v}_j \end{aligned} \tag{7.23}$$

Since $\mathbf{B}_{k/j}^{\mathrm{T}}\mathbf{b}_{kj}$ is equal to $\mathbf{B}_k^{\mathrm{T}}\mathbf{b}_{kj}$ with the j^{th} element, α_{kj} missing the above can be rearranged further:

$$\begin{aligned}\delta\mathbf{v}_{kj} &= f_{kj}(\mathbf{V}\mathbf{B}_k^{\mathrm{T}}\mathbf{b}_{kj} - \alpha_{kj}\mathbf{v}_j) - \mathbf{v}_j \\ &= \begin{cases} \frac{1}{1-\alpha_{kj}}(\mathbf{V}\mathbf{B}_k^{\mathrm{T}}\mathbf{b}_{kj} - \mathbf{v}_j) & |1-\alpha_{kj}| \geq \epsilon \\ -\mathbf{v}_j & |1-\alpha_{kj}| < \epsilon \end{cases}\end{aligned} \tag{7.24}$$

Thus to estimate the total leave-one-out error for k modes, one must do the following summation:

$$E_k = \sum_{j=1}^{s} |\delta\mathbf{v}_{kj}|^2 \tag{7.25}$$

In this way it is possible to pose the k-estimation as a standard minimization of the above summation.

A way to calculate E_k is to compute:

$$\delta\mathbf{V} = \mathbf{V}\mathbf{B}_k^{\mathrm{T}}\mathbf{B}_k - \mathbf{V} \tag{7.26}$$

If $\delta\mathbf{v}_j$ then denotes the j^{th} column of $\delta\mathbf{V}$ – the error measure E_k is given as:

$$E_k = \sum_{j=1}^{s} \begin{cases} f_{kj}^2|\delta\mathbf{v}_j|^2 & |1-\alpha_{kj}| \geq \epsilon \\ |\mathbf{v}_j|^2 & |1-\alpha_{kj}| < \epsilon \end{cases} \tag{7.27}$$

7.2 Iterative Model Optimization

The previous sections of this chapter have established a firm foundation for the optimization process in AAMs. This section will focus on the details in using the linear regression predictor in what is coined the *AAM search*.

The outline is to place the model in an initial state w.r.t. pose and model parameters, $\mathbf{c}_0$ – given by some prior initialization step – over the image, and obtain a normalized image sample below the model. Based on the

image sample and the model instance the difference vector, $\delta\mathbf{g}$, can be calculated and thus used to predict a set of parameters to make the model fit better to the image. This is done iteratively until no improvement is observed in the model fit.

A small refinement to the above is that if a prediction fails to improve the fit, the prediction is amplified/damped over some runs to see if that should improve the fit. This will compensate somewhat for overshooting and the opposite as experienced in the section on linear regression.

The procedure given in pseudo-code is:

1. Intialize the damping vector $\mathbf{k} = [1.5, 0.5, 0.25, 0.125, 0.0625]^{\mathrm{T}}$
2. Intialize the model within the reach of the parameter prediction range.
3. Generate the normalized texture vector from the model, $\mathbf{g}_m$
4. Sample the image below the model shape, $\mathbf{x}_{image}$, into $\mathbf{g}_{image}$
5. Normalized $\mathbf{g}_{image}$ into $\mathbf{g}_i$.
6. Evaluate the error vector $\delta\mathbf{g}_0 = \mathbf{g}_i - \mathbf{g}_m$
7. Evaluate the error $E_o = |\delta\mathbf{g}_0|$
8. Predict the displacements in pose $\delta\mathbf{t} = \mathbf{R}_t\delta\mathbf{g}_0$
9. Predict the displacements in model parameters $\delta c = \mathbf{R}_t\delta\mathbf{g}_0$
10. Set i=1
11. Update model parameters $\mathbf{c} = \mathbf{c} - k_i\delta\mathbf{c}$
12. Transform the shape to invert the $\delta\mathbf{t}$ transformation.
13. Repeat step 3-6 to form a new error E_i
14. If $E_i > E_0$ set $i = i + 1$ and go to step 10
15. Save the resulting error $E = E_i$
16. If no improvement is done in E from the last iteration, declare convergence. If not go to step 3.

Notice that when the algorithm terminates it is by no means certain that a valid model fit is obtained but merely an indication that the linear regression has given up in improving further on the fit. How to validate the final fit will be treated in depth in the section on *Performance assessment*.

An example of an AAM optimization is given in figure 7.2. In this example, the model converged after 7 iterations. In comparison with the original image, the AAM is somewhat blurrier because of the texture approximation.

Notice that the search could be sped up substantially by searching in a multi-resolution (pyramidal) framework.

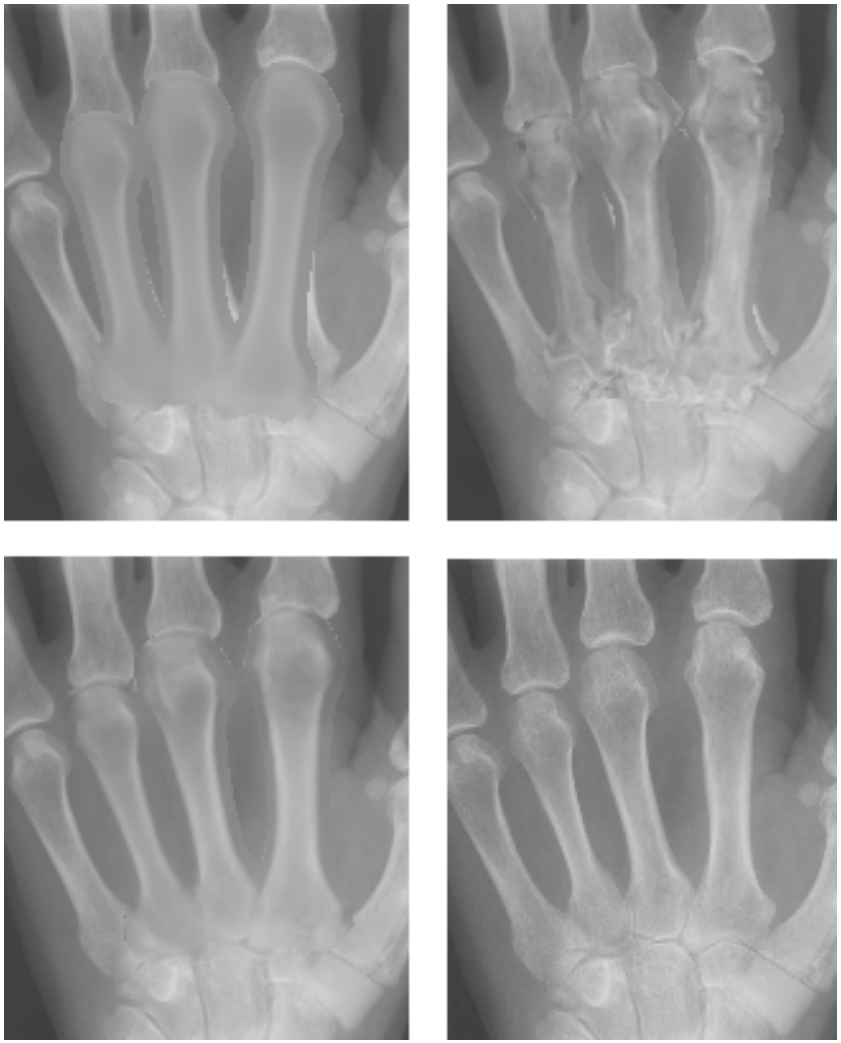


Figure 7.2: AAM Optimization. Upper left: The initial model. Upper right: The AAM after 2 iterations. Lower left: The converged AAM (7 iterations). Lower right: The original image.

7.3 Summary

This chapter concludes the description of the basic AAM. In this and the previous, it has been shown how to derive compact statistical models of shape and texture using principal component analysis. Furthermore, have these been combined into one unified model and an efficient optimization method based on multivariate linear regression has been described. This constitutes what is known as an Active Appearance Model.

Chapter 8

Discussion of Basic AAMs

8.1 Overview

This chapter provides a brief discussion of the Active Appearance Models approach as presented in the previous chapters. The forces and weaknesses of this approach are summarized and a number of secondary properties of AAMs are emphasized. Finally, for those comfortable with the Bayes paradigm, AAMs are posed in a Bayesian setting.

8.2 Forces

The AAM approach possess several pleasant properties:

General AAMs are general in the sense that any distinct shape and texture class can be modeled. Consequently, AAMs have been applied to a wide range of segmentation problems within computer vision:

- Cardiac MRIs [52] (chapter 14).
- Cross-sections of pork carcasses (chapter 15).
- Human faces [9, 10, 23, 24, 16, 17, 49, 14, 48, 18].
- MRIs of the human brain: [8, 10, 17, 14, 72].
- MRIs of the human knee: [10, 14].
- Radiographs of metacarpals (chapter 13).

Specific By capturing both shape and texture from a training set the AAM approach is highly specific to the segmentation task given.

Fast After a suitable initialization, the AAM search can be performed in the millisecond range using a standard PC.

Non-parametric Contrary to many other deformable models [26] the AAM has no parameters, which needs to be selected by an expert prior to each new segmentation problem. This makes the AAM approach truly data-driven.

Complete AAMs can synthesize near photo-realistic images of the object class learned from the training set.

Robust AAM segmentation usually succeeds in even cluttered images.

Captures domain knowledge A well-implemented AAM framework can be used by domain experts (medical doctors, radiologists etc.) with no need for assistance from image analysis experts.

8.3 Drawbacks

Below is listed a set of potential problems of basic AAMs with references to further descriptions and possible solutions (in no particular order):

Distinct features AAMs are designed to handle distinct shapes with recognizable features. Amorphous objects[1] such as clouds, trees, blood vessels (fundus images), cities & roads (remote sensing) etc. are not suitable. In other words; it must be possible to obtain a training set of representative examples.

Training set The cumbersome manual annotation of training sets is the Achilles' heel of AAMs. Several attempts have been done to address this drawback. Refer to section 4.3 for further treatment.

[1] Objects lacking a definite form.

Representation Shape variation may be over-constrained by a small training set (or a larger with insufficient variation) due to the PCA compression. However, this problem has been addressed for ASMs [11].

Point correspondence A fundamental assumption in AAMs is that of point correspondence. If training examples are available with a varying amount of model points (e.g. due annihilation and such) preprocessing is needed prior to AAM usage. Refer to section 4.3.

Initialization As mentioned earlier, AAMs are inherently dependent on a good initialization. This problem has later been addressed in [22]. Refer also to section 9.6.

Occlusions Due to the design of the AAM search – which performs parameter predictions based on pixel differences – outliers stemming from occlusions may cause the predictions to fail leading to optimization failure. This problem has been addressed in [22]. Refer also to section 9.8.

Homogeneous convex objects Convex objects with a low amount of texture variation may also pose a problem to basic AAMs. Refer to section 9.3.

Large-scale texture noise Heterogeneous objects with large-scale texture noise inside may pose a problem to basic AAMs, due to the PCA modeling of texture variation. Refer to section 9.4.

8.4 Hidden Benefits

Several techniques used in AAMs have previously been used in other applications. A far from exhaustive selection is given below. This should inspire to a creative use of AAMs contrary to a blackbox use, by using the secondary properties of AAMs:

Registration One of the purposes of using segmentation algorithms such as AAMs is to perform a subsequent image warping into some reference space of the object in question. This is especially true in many

medical applications. Fortunately – as seen in chapter 5 – registration is an inherent property of AAMs.

Analyses of Variance In group studies or longitudinal studies as for example biological studies, the goal is often to depict or describe the morphological changes [20]. For this, the shape PCA is very useful. Furthermore, the user is provided with a texture analysis and an analysis of the correlation between shape and texture through the combined PCA.

Classification/Interpretation The model parameters (also called the PCA scores) of either/or the shape-, texture- and combined-PCA can be used for classification and interpretation purposes. Edwards et al. [24] use this property to separate identity from variation in expression, pose and lightning in an AAM built upon faces.

8.5 AAMs Posed in a Bayesian Setting

For those comfortable with the Bayes paradigm of computer vision, we will pose AAMs in a Bayesian setting. This section is largely based on the general treatment of deformable template models in [26].

A Bayesian formulation of a vision problem is in general attractive, since it gives a natural separation of the model and image contribution to a given solution. In a probabilistic framework these are given as a *prior* and a *likelihood* probability distribution, respectively.

Prior – $P(\mathbf{v}|\theta)$ where $\mathbf{v} \in \Omega_{\mathbf{v}}$ denotes the model parameters, $\mathbf{c}$, and the pose parameters, and θ denotes the mean shape, mean texture and their corresponding covariance structures. θ is thus obtained from the training set.

Likelihood – $P(\mathbf{I}|\mathbf{v},\theta)$ where $\mathbf{I} \in \Omega_{\mathbf{I}}$ denotes the image being searched in, and again θ denotes the mean shape, mean texture and their corresponding covariance structures. θ is thus obtained from the training set.

The prior and likelihood distribution are then combined to form the *posterior distribution*, $P(\mathbf{v}|\mathbf{I},\theta)$, by use of Bayes theorem:

$$\begin{aligned} P(\mathbf{v}|\mathbf{I},\theta) &= \frac{P(\mathbf{v}|\theta)P(\mathbf{I}|\mathbf{v},\theta)}{P(\mathbf{I}|\theta)} \\ &\propto P(\mathbf{v}|\theta)P(\mathbf{I}|\mathbf{v},\theta) \end{aligned} \tag{8.1}$$

The Bayes formulation is then utilized (in e.g. segmentation) for making inference[2] about $\mathbf{v}$. The maximum a posteriori (MAP) of $\mathbf{v}$ is:

$$\hat{\mathbf{v}} = \max_{\mathbf{v}} P(\mathbf{v}|\mathbf{I},\theta) \tag{8.2}$$

The first thing to observe is that Active Appearance Models do not have an explicit prior model – i.e. the prior distribution is uniform. Implicitly however, the prior is an inherent property of an AAM since the training set only allows shape and texture to be within a certain subspace of the shape and texture space, due the (truncated) change of basis caused by the PCA.

The second thing to observe is that our optimization method in section 7.2 is formulated as an energy minimization rather than a probability maximization as in (8.2). Fortunately – under the weak assumption that the posterior distribution is Gibbs distributed – equation (8.1) can be expanded to [26]:

$$P(\mathbf{v}|\mathbf{I},\theta) = \frac{1}{z}e^{-U(\mathbf{v},\mathbf{I},\theta)} \tag{8.3}$$

where $U(\mathbf{v},\mathbf{I},\theta) : \Omega_{\mathbf{v}} \rightarrow \mathbb{R}$ is the energy function – i.e. the similarity measure function used in AAMs. And z is a normalizing constant which ensure a proper distribution.[3] Due to dimensionality of Ω_v and especially Ω_I it is highly infeasible to determine the value of z. However, since z is a constant this is of no importance to energy optimization.

By this (8.2) can be formulated as an energy minimization in the AAM case:

[2]The act or process of deriving logical conclusions from premises known or assumed to be true.

[3]I.e. $\int_{\Omega_{\mathbf{v}},\Omega_{\mathbf{I}}} P(\mathbf{v}|\mathbf{I},\theta)d\mathbf{v}d\mathbf{I} = 1$.

$$\begin{aligned} \hat{\mathbf{v}} &= \max_{\mathbf{v}} P(\mathbf{v}|\mathbf{I},\theta) \\ &= \max_{\mathbf{v}} -U(\mathbf{v},\mathbf{I},\theta) - \log(z) \\ &= \min_{\mathbf{v}} U(\mathbf{v},\mathbf{I},\theta) \end{aligned} \tag{8.4}$$

This concludes the topic of how AAMs can be interpreted in a Bayesian setting.

Chapter 9

Extensions of the Basic AAM

9.1 Overview

Though the previous chapters have provided a description and discussion of AAMs, extensions can still be done of the basic AAM.

The major contribution of this thesis is – aside from the discussion and documentation of AAMs – represented by the ideas and work described in this and the following chapter.

Other extensions to the basic AAM are the work of Cootes et al. [22, 13, 14], Walker et al. [71], Wolstenholme et al. [72], Mitchell et al. [52]. Some of these extensions will be surveyed in chapter 17.

9.2 Enhanced Shape Representation

Basic AAMs using the piece-wise affine warping, relies on the Delaunay triangulation of the shape points. This results in a triangular mesh covering the convex hull of the point set.

For concave shapes this might not be the optimal solution. For example there could be substantial texture noise in the area outside the shape – but still inside the convex hull – thus leading to a less compact model.

To avoid this we suggest removing the triangles outside the shape. This is trivial to accomplish. Simply traverse the triangles; test if the triangle centroid should be outside the shape polygon. If so, remove the triangle.

See figure 9.1 for a pictorial of the problem. To test if a point is inside the shape polygon several approaches could be taken. The current implementation utilizes the geometrical fact that, if a line from the point, $\mathbf{p}$, to infinity crosses the polygon an odd number of times, then the point $\mathbf{p}$ is inside the polygon.

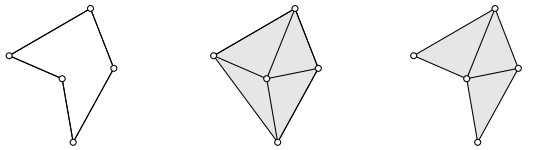


Figure 9.1: Removal of unwanted triangles resulting from the Delaunay triangulation of concave shapes.

(a) (b)

Figure 9.2: (a) Concave shape with convex triangles. (b) Concave shape with convex triangles removed.

Though the above method adds flexibility to the choice of the AAM shape structure, one can still think of problems where even greater shape flexibility is required. Objects can contain areas where the texture variation

might be considered noise. One might want to exclude such areas due to arguments similar to the above given. Another situation is that of having several structured objects, but in between those, the texture is highly unstructured.

Features to accommodate such situations are implemented in the current AAM framework. Shapes are defined in terms of *paths*, which is a subset of the shape points with a given point connectivity. Each path can have a combination the following properties:

- **Open/closed path** – Open path: a path where all points are connected to two points each.
- **Inner/outer path** – Outer path: a path where only the inside is included into the shape surface.
- **Original/artificial path** – Artificial path: a path added after the original annotation.
- **Hole/non-hole** – Hole: a path where the inside is excluded from the shape surface.

This provides the builder of an AAM with a high degree of freedom, resulting in a more optimal AAM representation of the given problem. For further details of the representation, refer to appendix G.

9.3 Increasing Texture Specificity

While the enhanced shape representation of the previous section gave great flexibility it also increased the risk of what we coin the *shrinking problem*.

During matching of objects with a relative homogeneous surface – such as the metacarpals – matches sometimes tends to lie inside the real object as illustrated on figure 9.3. This is due to the fact that the AAM evaluates the fit on the object texture only.

To avoid this shortcoming we suggest including some neighboring region of the object. This will usually lead to more texture contrast in the model,

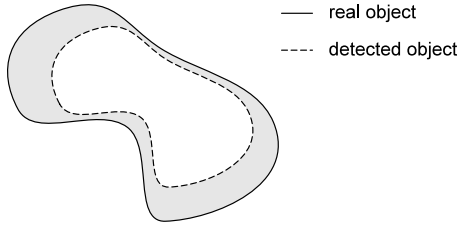


Figure 9.3: The shrinking problem.

since objects (often) are identified as something that stands out from its surroundings.

This neighborhood adding must be done carefully to preserve compactness of the AAM. Clearly, the texture PCA will suffer since the goal is to add new information, namely background pixels, of which one could expect a substantially higher degree of variation than across the object. However, regarding the shape PCA, points can be added without adding new shape information, thus retaining the eigenvalue distribution of the shape PCA. This is accomplished by generating new shape points correlated with the old ones, by adding new points on the normals of the original points in a distance proportional to the relative size of the shape. See figure 9.4.

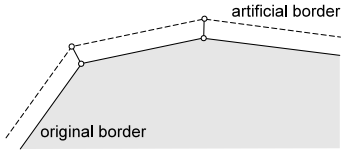


Figure 9.4: Shape neighborhood added using an artificial border placed along the normals.

In pseudo-code this is:

1. Choose an artificial point distance, $d_{mean} = n$ pixels, relative to the mean shape.
2. Calculate the mean size of all n shapes, $s_{mean} = \frac{1}{n}\sum_{i=1}^{n} S(\mathbf{x}_i)$.
3. For each shape $[1 \ldots n]$
4. For each outer path
5. For each model point
6. Calculate the normal of the point
7. $d = d_{mean}\frac{S(\mathbf{x}_i)}{s_{mean}}$
8. Add an artificial point d pixels along the point normal
9. End
10. End
11. End

(a) (b)

Figure 9.5: (a) Shape annotated using 150 landmarks. (b) Shape with a neighborhood region added resulting in $2 \times 150 = 300$ landmarks.

Refer to the experimental section for results on using this rationale. The metacarpals in figure 9.5 serves a good example of a shape with a relative

homogeneous surface. By adding neighborhood the texture in 9.5 (b) is substantially more specific than the shape without, figure 9.5 (a).

9.4 Border AAMs

As earlier stated the AAM approach is a direct extension of the Active Shape Models [15]. A major force of ASMs is the handling of objects with heterogeneity inside shapes, in the meaning of inside areas with high texture variation that is not possible to capture by annotation.

Since ASMs only model the texture variation in a local neighborhood around the landmarks, the heterogeneity issue never arises.

To incorporate this force of ASMs into AAMs, the enhanced shape representation has been used to provide the current implementation with the feature of *Border AAMs* – i.e. AAMs that only capture texture information on the border of the object in question. Border AAMs can be thought of as ASMs with very closely spaced landmarks.

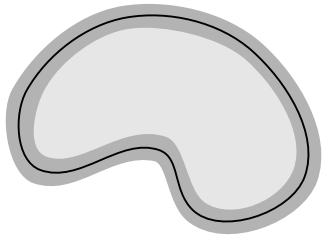


Figure 9.6: ASM-like AAM generated by adding shape neighborhood and a hole.

A Border AAM is achieved by adding an interior path, which defines a hole and by adding an outer path as described in the previous section. A schematic example on a Border AAM is given figure 9.6.

By using this rationale AAMs can be made insensitive to large-scale texture noise inside the shapes, which otherwise would lead to a poor texture fit and a low landmark accuracy.

(a) (b)

Figure 9.7: (a) Shape annotated using 83 landmarks. (b) Border shape with $3 \times 83 = 249$ landmarks.

The pork carcasses of figure 9.7 constitutes a good example of this situation, due to the heterogeneity of the structure of fat and meat from one training example to another.

To conclude this section, we stress that Border AAMs also should be substantially faster than basic AAMs, since only a fraction of the original pixels is considered.

9.5 Constrained AAM Search

This section is merely a documentation of considerations for the AAM search, rather than an actual extension. However, this is included here because no accurate description exists to our knowledge.

As described in section 7.2 the optimization is accomplished using a combination of pose and model parameter predictions. As seen earlier these predictions are only valid within some range. Consequently, it would make good sense to constrain the predictions within these ranges. For pose parameters we define the bounds as being a function of the training range $[-\alpha; \alpha]$:

$$\text{pose parameter bound}: \quad [-\kappa\alpha; \kappa\alpha] \tag{9.1}$$

For the model parameters we assume that these are independently normal distributed over the training set, thus making the bound proportional to

the standard deviation over the training set – i.e. $\sqrt{\lambda_i}$. The bounds for the i^{th} model parameter is then:

$$\text{model parameter bound}: \quad [-\nu\sqrt{\lambda_i}; \nu\sqrt{\lambda_i}] \tag{9.2}$$

The current implementation uses $\kappa = 2$ and $\nu = 3$.

Another – and theoretically better – approach could be to calculate the Mahalanobis distance of the model parameters and do a test in the χ^2-distribution. If this would fail, the model parameters are then projected onto the closest point of the hyper ellipsoid constituted by the Mahalanobis distance. Refer to the later section on Performance Assessment for a further treatment.

9.6 Initialization

The basic AAM optimization scheme is inherently dependent on good initialization. To accommodate this, we devise the following search-based scheme thus making the use of AAMs fully automated. The technique is inspired by the work of Cootes et al. [22] who uses a pixel difference evaluation criteria and a threshold estimation for detecting multiple object instances.

The fact that the AAMs are self-contained is exploited in the initialization – i.e. they can fully synthesize (near) photo-realistic objects of the class that they represent concerning shape and textural appearance. Hence, the model, without any additional data, can be used to perform the initialization.

The idea is to exploit an inherent property of the AAM-optimization – i.e. convergence within some range from the optimum. This is utilized to narrow down an exhaustive search from a dense to a sparse population of the hyperspace spanned by pose- and $\mathbf{c}$-parameters. In other words, normal AAM-optimizations are performed sparsely over the image using perturbations of the pose and model parameters.

This has proven to be both feasible and robust. A set of relevant search configuration ranges is established and the sampling within this set is done as sparsely as possible.

Consider the graph given in figure 7.1, which demonstrates that it should be safe to sample the y-parameter with a frequency of at least 10 pixels. One could also claim that as long as the prediction preserves the right sign it is only a matter of a sufficient number of iterations.

To evaluate the fit, the normal similarity measure of AAMs is used:

$$E_{fit} = |\delta \mathbf{g}|^2 = \sum (g_{i,model} - g_{i,image})^2 \tag{9.3}$$

As in the normal optimization, this could easily be improved by using more elaborate error measures.

The crucial add-on to this algorithm is somewhat inspired from the class of Genetic Algorithms[1] (GA) which is based on the natural selection of the Darwinian theory. The total set of search configurations constitutes the initial population. From this we let the n fittest survive. These are then reproduced into more evolved guesses. From these the best is drawn and deemed the initial configuration. This is sometimes referred to as a *multiple hypotheses* technique.

In pseudo-code, the initialization scheme for detecting one object per image is:

1. Set m to a suitable low number (we use $m = 3$)
2. Allocate room for a candidate set, $\{\mathbf{c}\}$, containing n result entries
3. Obtain application specific search ranges within each parameter (e.g. $-\sigma_1 \leq c_1 \leq \sigma_1,\ x_{min} \leq x \leq x_{max}$, etc.)
4. Populate the space spanned by the ranges – as sparsely as the linear regression allows – by a set of search configurations $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$.
5. For each vector in $\mathbf{V}$
6. Do AAM optimization (max m iterations)
7. Calculate the fit, $E_{fit} = |\delta \mathbf{g}|^2$
8. If $E_{fit} < \max_{E_{fit}} \{\mathbf{c}\}$ add $(\mathbf{V}_i, E_{fit})$ and remove the element $\max_{E_{fit}} \{\mathbf{c}\}$
9. End
10. For each element in $\{\mathbf{c}\}$
11. Do AAM optimization (max k iterations, $k > m$)
12. Calculate the fit, $E_{fit} = |\delta \mathbf{g}|^2$
13. Update E_{fit} of the current entry
14. End

[1] Notice however, while GAs are probabilistic, our technique is deterministic. Further more are the aspects of mutation and crossover in GAs not utilized here.

The element in $\{\mathbf{c}\}$ with the minimum E_{fit} will now hold the initial configuration.

Notice that the application specific search ranges in step 3 is merely a help to increase initialization speed and robustness than it is a requirement. If nothing is known beforehand, step 3 is eliminated and an exhaustive search is performed.

This scheme is readily extended into more than one object per image by a clustering of the candidate set using overlap tests. The approach in general can be accelerated substantially by searching in a multi-resolution (pyramidal) representation of the image.

Another example on search-based initialization of deformable models is [45]. Also [27, 29, 28] featured search-based initialization where multiple rigid template matching using the convolution theorem[2] for a FFT formulation was utilized.

9.7 Fine-tuning the Model Fit

Though the AAM search provides a fast way of optimizing the AAM using prior knowledge this might not lead to the optimal solution, primarily due to weakness in the assumption that the optimization problems in an AAM search is strictly similar to those observed in a training set.

Therefore, it is suggested to fine-tune the model fit by using a general-purpose optimization method. This approach is feasible since it is reasonable to assume that we are close to the optimum after the traditional AAM search. Hence the optimization fine-tuning should be possible in an reasonable amount of time.

Though one is not guaranteed that the hyperspace is totally smooth at the position where the AAM search has converged, it is still more probable that we are near a well-behaved manifold in the hyperspace.

[2] I.e. $\mathcal{F}(f * g) = \mathcal{F}(f)\mathcal{F}(g)$, where f, g are two signals, $\mathcal{F}$ denotes the Fourier transform and $*$ the convolution operator.

The considered optimization methods are:

- Gradient based methods:
  - Steepest descent (SD)
  - Conjugate gradient with Fletcher-Reeves update (CG) [31]
  - Quasi-Newton (BFGS)
- Non-gradient based methods:
  - Pattern search (PS) [40]
- Random-sampling based methods:
  - Simulated annealing (SA) [7, 47]

Another noteworthy random-sampling technique is the *Genetic algorithms* which has proven successful and efficient in applications of other deformable models (e.g. [39]). However, this not considered in the following. Other popular optimization methods used in deformable models include Marquardt-Levenberg as used in the Active Blobs framework [58] as a supplemental to difference decomposition [34].

Conjugate Gradient and BFGS are well known optimization methods (see e.g. [19]), which should have better properties than Steepest Descent. Pattern Search is a non-gradient based optimization algorithm, which has received increasing interest in the optimization community. Finally Simulated Annealing borrows ideas from a physical procedure called annealing where a substance is melted and then slowly cooled down in search of a low energy configuration. In a similar manner, probabilistic optimization is performed with a decreasing temperature that determines how greedy the procedure is in the search for a global minimum. However, since a detailed discussion of optimization methods is somewhat outside the scope of this thesis, the reader is referred to [19, 31, 61].

To assess the effect of such fine-tuning an AAM has been built upon 20 metacarpal images (238×272×8bit) where metacarpals 2, 3 & 4 were annotated using 50 points each. A model neighborhood of four mean shape pixels was added. The texture model consisted of approx. 13.000 pixels. Here after the model has been used to search four unseen images with a ground truth annotation using the automatic initialization approach devised previously. Results reported in table 9.7 are the mean of the four

Type	**Pt.-Crv.** (pixels)	**Pixel error** (intensity)
None	0.92	5.7
BGFS	0.89	5.5
SD	0.88	5.5
CG	0.87	5.5
PS	0.87	5.3
SA	0.89	5.4

Table 9.1: Mean fit results using general-purpose optimization methods for fine-tuning.

experiments. For a definition of *pt. to crv.*, refer to the later section on *Performance Assessment*.

This investigation is not conclusive. Nevertheless, the presented results indicate clearly that a fine-tuning of the AAM fit can be accomplished successfully using general-purpose optimization techniques.

In the current case the simple method *Pattern Search* yielded the best results, reaching a 5 % improvement over the point accuracy and 7 % over the pixel error. However, since the performance of these methods is quite sensitive to configuration parameters – i.e. step sizes, standard deviations, stop criteria – we stress that this might not be generally true. In agreement with this, the random-sampling optimization technique *Simulated Annealing* is chosen in the experimental chapters over the better Pattern Search in the example. This decision is motivated by the observation that our objective function – i.e. $|\delta\mathbf{g}|^2$ – most likely is non-convex. Thus deterministic optimization techniques has a high risk of getting caught in spurious minimas, whereas random-sampling techniques are more likely to escape.

9.8 Robust Similarity Measures

As mentioned earlier the basic AAM optimization is driven by texture differences. More accurately the square length of the normalized texture difference vector, $|\delta\mathbf{g}|^2$ is used. The measure with which the optimization evaluates itself with, is here forth named the *similarity measure*. This constitutes the essence of what one want: evaluate how similar the model is to the image.

However, the term *similar* is inherently vague in a mathematical sense. The term $|\delta\mathbf{g}|^2$ is thus only one interpretation of what similar actually means.

In this section, we will dwell on other interpretations of the term similar. Namely, the one that mimics the human ability of compensating for small numbers of gross errors, and thus achieving robustness in recognition. These are called *robust* similarity measures and stems from an increasingly popular statistical discipline in vision named *robust statistics*, where the term robust refer to the insensitivity to outliers. The notation and treatment below is somewhat based on the work of Black and Rangarajan [3].

Cootes et al. [22] has previously extended the basic AAM with learning-based matching to obtain robustness. This is achieved using a threshold for each element in $\delta\mathbf{g}$ estimated from the training set.

We suggest using robust similarity measures. To formalize the model fitting problem, a set of parameters, $\mathbf{c} = [c_1, \ldots, c_p]^{\mathrm{T}}$, are adjusted to fit a set of measurements (e.g. an image), $\mathbf{g} = [g_1, \ldots, g_m]^{\mathrm{T}}$. This is done by a minimization of the residuals:

$$E = \sum_{i=1}^{m} \rho(g_i - u(i, \mathbf{c}), \sigma_s) = \sum_{i=1}^{m} \rho(e_i, \sigma_s) \tag{9.4}$$

where u is a function that returns the model reconstruction of the i^{th} measurement and σ_s is the scale parameter that determines what should be deemed outliers.

The ρ-function determines the weighting of the residuals, and is also called the *error norm*. The most common error norm is least squares or the *quadratic norm*:

$$\rho(e_i) = e_i^2 \tag{9.5}$$

This is often referred to as the L_2 norm. Basic AAMs uses the quadratic norm (or simply the 2-norm) without any normalization:

$$E = \sum_{i=1}^{m} (g_{model} - g_{image})^2 = \sum_{i=1}^{m} \delta(g_i)^2 = |\delta\mathbf{g}|^2 \tag{9.6}$$

It is however easily seen, that the quadratic norm is notoriously sensitive to outliers, since these will contribute highly to the overall solution due the rapid growth of the x^2 function. To quote [3] pp. 62:

> *"... an estimator must be more forgiving about outlying measurements ..."*

A straightforward approach is to put an upper bound on the accepted residual. This is called the *truncated quadratic norm*:

$$\rho(e_i, \sigma_s) = \begin{cases} e_i^2 & e_i \leq \sqrt{\sigma_s} \\ \sigma_s & e_i > \sqrt{\sigma_s} \end{cases} \tag{9.7}$$

Another approach is to make the residual growth linear above a certain threshold. This results in the so-called *Huber's minimax estimator*:

$$\rho(e_i, \sigma_s) = \begin{cases} \frac{e_i^2}{2\sigma_s} + \frac{\sigma_s}{2} & e_i \leq \sigma_s \\ |e_i| & e_i > \sigma_s \end{cases} \tag{9.8}$$

Another smooth norm – which falls off faster than the Huber norm – is the *Lorentzian estimator* as used by Sclaroff and Pentland in the Active Blob framework [58][3]:

$$\rho(e_i, \sigma_s) = \log(1 + \frac{e_i^2}{2\sigma_s^2}) \tag{9.9}$$

Finally we suggest to use a similarity measure with closed connection to the Mahalanobis distance, since it takes the variance of the training set into account and thus makes the similarity measure less sensitive to large residuals in areas of high variance (as observed in the training set). Instead of using the dependent Mahalanobis distance:

$$\rho(e_i) = (\mathbf{g} - \overline{\mathbf{g}})^{\mathrm{T}} \Sigma^{-1} (\mathbf{g} - \overline{\mathbf{g}}) \tag{9.10}$$

[3] To achieve performance the log-function was implemented using precomputed look-up tables.

where Σ is the texture difference covariance matrix, we are compelled to assume independence[4] between pixels, due to the length of the pixel vectors which makes the computation of (9.10) infeasible. This reaches a form similar to the Mahalanobis distance norm of:

$$\rho(e_i) = \frac{e_i^2}{\sigma_i^2} \tag{9.11}$$

where σ_i^2 is the maximum likelihood estimate of the variance of the i^{th} pixel. Notice however, that this is *not* the Mahalanobis distance since σ_i^2 should be the variance of the difference to be so.

Of the above similarity measures the Lorentzian and the "Mahalanobis" distance have been integrated into the basic AAM to supplement the quadratic norm of Cootes et al. Thereby robustness to outliers in the unseen image has been obtained. An example of this could be to detect the complete bone structure of human hands in radiographs. Since radiographs are 2D projections of density, people wearing finger rings will have high-white outliers on one or more phalanges. Other examples include highlights in perspective images and absence of interior parts (for example due to partial occlusion).

However, one should notice that even though the AAM search evaluate its predictions using a robust measure, the predictions themselves are done using the pixel differences. To address this problem Cootes et al. [22] performs a thresholding of the texture vector before the prediction. This could be viewed upon as a robust preprocessment. The threshold limit is estimated from the training set.

Consequently, to evaluate the proposed similarity measures the fine-tuning optimization was included in all experiments.

To show the effect of robust error norm usage, an AAM search with fine-tuning using Simulated Annealing has been done *with* and *without* a robust similarity measure. In the case given in figure 9.8 the Lorentzian error norm was used. To simulate outliers the radiograph searched in has been manipulated so that it appears as if the metacarpal is wearing a finger

[4]Unfortunately *dependence* is the fundamental characteristic of the concept of an image.

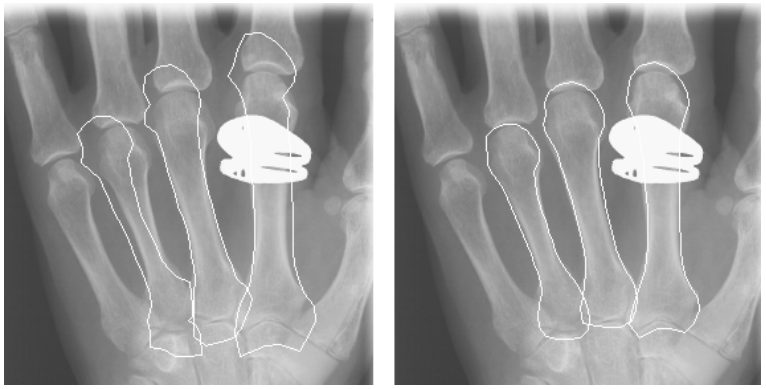


Figure 9.8: Example of AAM search and Simulated Annealing fine-tuning, without (left) and with (right) the use of a robust similarity measure (Lorentzian error norm). Landmark error decreased from 7.0 to 2.4 pixels (pt.-to-crv. error).

ring.[5] While not perfect, the robust AAM provides a substantially better fit compared to that of the basic AAM.

Through preliminary experiments, it was observed that the Lorentzian error norm performed best. Hence, this is the only norm used in the experimental chapter.

For a further treatment of robust error norms and line processes in vision refer to [3].

9.9 Summary

In this chapter several ideas for extending the basic AAM have been proposed and implemented. Each one addressing potential problems in certain cases of the basic AAM. Preliminary experiments have been conducted to 1) demonstrate the effect of these and 2) to form a basis for the selection of extensions in the experimental part.

[5]Though this is highly unlikely since the metacarpals is situated in the middle of the hand.

Chapter 10

Unification of AAMs and Finite Element Models

10.1 Overview

In conjunction with the previous chapter, this chapter constitutes the primary contribution of this thesis.

An idea for extending the AAM framework is presented conceptually and followed by an algorithmic formulation, which can be readily implemented in an AAM system. This has been done and preliminary results and observations hereof are presented.

The treatment of finite element models (FEM) is partly inspired from the work with extending shape flexibility in ASMs by Cootes et al. [11] and the Active Blob / Active Voodoo Doll models of Sclaroff and Isidoro [43, 58].

Quite recently[1] Cootes et al. proposed a combination of elastic and statistical models of appearance [13], which should not be confused with the contribution below. In [13] elasticity was obtained using so-called *local AAMs* – i.e. AAM-blobs around each landmark with a smoothness constraint, thus providing a free-form-like deformable model extension of AAMs.

[1] ECCV, Dublin, June 2000.

10.2 Motivation

The Basic AAM is inherently driven by texture differences. Thus AAMs will favor texture fit at the expense of landmark precision. This is quite acceptable since the texture fit is the only part that is explicitly known at the optimization time.

The predecessor to AAMs, Active Shape Models (ASM) model texture using a local neighborhood around each landmark[2]. This makes the ASM approach less sensitive to texture abnormalities in the middle of the shape, but also less specific and less complete. In agreement with this, a comparative study between ASMs and AAMs [17] concludes that ASMs achieves more accurate landmark location than AAMs, but AAMs gives a better texture match.

This chapter devises a method to enhance the texture fit without sacrificing landmark accuracy.

To fully understand the source of the problem look at the synthetic example in figure 10.1. Here an object, a, in an image with a high background intensity (light gray) is depicted. Inside a there is an area that is hard to place landmarks upon. In this example this is represented by the radial blob, b. Imagine that the blob, b is roughly on the same spot on a across the whole training set. If one then supplies a new image to search in, where b is displaced, the AAM will sacrifice the edge accuracy to accommodate the displacement of b. Which is due to the fact that there is far less contrast in the edge area than in the blob, b. This makes it "cheaper" to accept the erroneous edge fit, since the only way to modify the inside of the model is by moving the outside points.

To solve this situation one could include the example where the blob has moved, into the training set. This would however make the compactness of the texture model suffer. Another approach could be to annotate the blob in all training images, thus retaining the compactness of the texture model. Unfortunately, this approach is not always feasible since objects can contain structures within that are very hard to annotate, due to arbitrarily absence or structures with no well defined extends. Furthermore, the manual definition of landmarks is the Achilles' heel of AAMs, which

[2] Although any neighborhood could be used, a 1-pixel wide profile at each landmark normal is used.

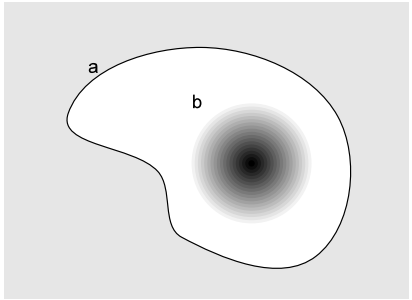


Figure 10.1: A shape, a, with a blob, b, inside that is hard to annotate.

makes such an approach undesirable in general. We need to minimize the need for additional landmarks.

To increase the quality of the texture model and the robustness [10], Cootes et al. annotated well defined – but varying – structures such as the nose and the eyebrows in their face model.

10.3 The Basic Idea

To accommodate the above-mentioned flaws it is proposed to make the inside of AAMs more flexible. In this way the inside could be deformed to fit the blob of the above example *without* sacrificing the precision at the border.

To obtain such flexibility we suggest enforcing a physical model of a rubber-like material over the inside of the AAM. The class of models called *finite element models* (FEM) is chosen to accomplish this task.

Using FEMs, a compact parameterization can be obtained to deform the inside of the model. These parameters can then be used as a part of the AAM optimization to provide greater flexibility.

10.4 Finite Element Models

A simple conceptual model for an FEM, is to model an elastic body as a set of point masses mutually interconnected by springs. The material properties are then given by the masses, the spring constant and the rest length. See figure 10.2.

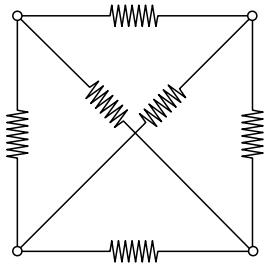


Figure 10.2: A finite element model interpreted as a set of point masses interconnected by springs.

To represent the position of the masses a vector similar to the shape vector is used. Thus having n masses:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n]^{\mathrm{T}} \tag{10.1}$$

Formally, a finite element model can be written as a generalized eigenproblem. If $\mathbf{M}$ denotes a mass matrix and $\mathbf{K}$ stiffness matrix (containing the spring constants and rest lengths) this is [11]:

$$\mathbf{K}\boldsymbol{\Phi}_{FEM} = \mathbf{M}\boldsymbol{\Phi}_{FEM}\boldsymbol{\Omega}^2 \tag{10.2}$$

where $\boldsymbol{\Phi}_{FEM}$ is a matrix of eigenvectors:

$$\boldsymbol{\Phi}_{FEM} = \begin{bmatrix} & & \\ \phi_1 & \cdots & \phi_{2n} \\ & & \end{bmatrix} \tag{10.3}$$

representing the modes of vibration and $\mathbf{\Omega}^2$ is a diagonal matrix of eigenvalues:

$$\mathbf{\Omega}^2 = \begin{bmatrix} \omega_i^2 & & \\ & \ddots & \\ & & \omega_{2n}^2 \end{bmatrix} \tag{10.4}$$

associated with each eigenvector.

If $\overline{\mathbf{x}}$ denotes the equilibrium configuration – i.e. the non-deformed material, then a deformed version, $\mathbf{x}$ can be obtained by a weighted sum of eigenvectors given by the vector, $\mathbf{u}$:

$$\mathbf{x} = \overline{\mathbf{x}} + \mathbf{\Phi}_{FEM}\mathbf{u} \tag{10.5}$$

Henceforth $\mathbf{u}$ is called the FEM deformation parameters. The energy of the deformation in the i^{th} parameter is proportional to ω_i^2.

A simple FEM can be obtained by assuming unit masses and rest length of the springs equal to the distance between the points. This makes $\mathbf{M}$ the identity matrix and the stiffness matrix becomes:

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{xx} & \mathbf{K}_{yx} \\ \mathbf{K}_{xy} & \mathbf{K}_{yy} \end{bmatrix} \tag{10.6}$$

where $\mathbf{K}_{xy} = \mathbf{K}_{yx}^{\mathrm{T}} = \mathbf{K}_{yx}$, all size $n \times n$ with off-diagonal elements given by:

$$k_{xxij} = \frac{(x_i - x_j)^2}{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{10.7}$$

$$k_{yyij} = \frac{(y_i - y_j)^2}{(x_i - x_j)^2 + (y_i - y_j)^2} \ , \tag{10.8}$$

$$k_{xyij} = \frac{(x_i - x_j)(y_i - y_j)}{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{10.9}$$

and diagonal elements:

$$k_{xxii} = k_{yyii} = 1 \ , \ k_{xyii} = 0 \tag{10.10}$$

$\mathbf{\Phi}_{FEM}$ is now easily obtained through an eigenvalue decomposition of $\mathbf{K}$.

One should notice the striking similarity with the PCA approach. Instead of *modes of variation* the FEM provide *modes of vibration* as controlled by the FEM deformation parameters, $\mathbf{u}$.

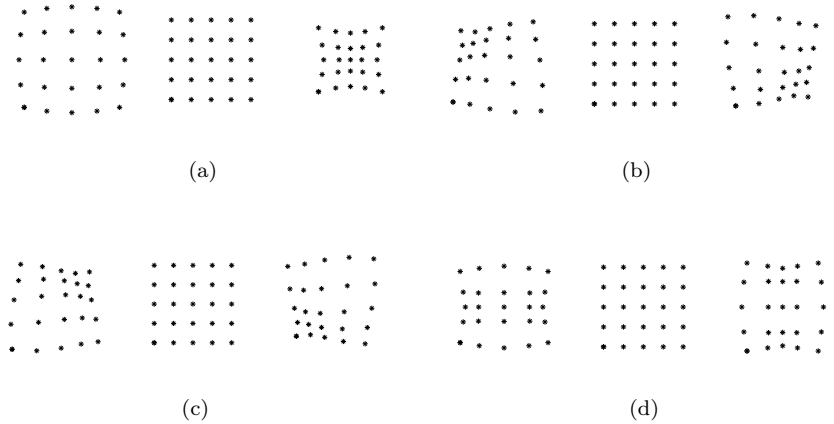


Figure 10.3: High frequency FEM-modes of a square surface modelled by 25 unit masses.

As mentioned above, FEM has an ordering of the eigenvectors corresponding to the amount of deformation and to the frequency of deformation. For simple shapes the first, low-frequency modes will correspond to rigid body changes and more intuitive deformations – i.e. bending etc. Whereas the high frequency modes contain more complex localized deformations [11, 59]. The effect of deforming a square surface modeled by 25 unit masses using high frequency modes can be seen in figure 10.3.

10.5 Integration into AAMs

As the above section introduced the basics of FEMs this section will demonstrate how to utilize an FEM in the AAM framework.

The first thing to notice is that the FEM deformations described in the previous chapter displaced all points on the surface. In AAMs we are interested in deforming the *interior* of a shape rather than the whole shape. Consequently the concept of *artificial interior points*, (AIP) is introduced. AIPs are generated using a given *point generating function*, $\mathbf{q}(\mathbf{x})$, and a shape, $\mathbf{x}$. The generated AIPs are then deformed using an FEM, to increase the AAM fit.

Formally this can be viewed upon as the normal shape vector, $\mathbf{x}_s$ being augmented with the AIPs, $\mathbf{x}_a$ into the combined shape vector $\mathbf{x}_c$. Thus, the deformations can now be written as:

$$\mathbf{x}_c = \begin{bmatrix} \mathbf{x}_s \\ \mathbf{x}_a \end{bmatrix} = \begin{bmatrix} \overline{\mathbf{x}} \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{\Phi}_s \\ \mathbf{0} \end{bmatrix} \mathbf{W}_s^{-1} \mathbf{\Phi}_{c,s} \mathbf{c} + \mathbf{h}\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{\Phi}_{FEM} \end{bmatrix} \mathbf{u}, \mathbf{c}, \mathbf{q}(\overline{\mathbf{x}}) \right) \tag{10.11}$$

where $\mathbf{0}$ denotes the zero vector and matrix respectively and $\mathbf{h}$ is a function that warps the deformed AIPs to match the current shape (as defined by $\mathbf{c}$).

As an alternative to use (10.11) directly, the technique can also be interpreted as a way of manipulating the warp function to act as if the interior of a shape has been deformed. In pseudo-code this would be:

1. Generate AIPs using the mean shape, $\overline{\mathbf{x}}$.
2. Concatenate the AIPs and the mean shape into a warp shape $\mathbf{x}_{warp,c}$
3. Deform the AIPs using a FEM into the point set $\mathbf{p}$.
4. Obtain the PCA-deformed shape $\mathbf{x}_s$.
5. Establish a warp, $\mathbf{f}()$, between $\mathbf{x}_s$ and $\overline{\mathbf{x}}$
6. Warp each deformed AIP in $\mathbf{p}$ using $\mathbf{f}()$ into the point set $\mathbf{p}_{warp}$
7. Concatenate the AIPs in $\mathbf{p}_{warp}$ and $\mathbf{x}_s$ into $\mathbf{x}_{s,c}$
8. Establish a warp, $\mathbf{g}()$, between $\mathbf{x}_{s,c}$ and $\mathbf{x}_{warp,c}$
9. Use $\mathbf{g}()$ to obtain the texture for the shape $\mathbf{x}_s$.

The pseudo-code is also expressed graphically in figure 10.4. The case of a triangular mean shape is used to keep it simple.

It now remains to define a suitable point generating function to obtain the AIPs. Since the current AAM implementation uses a piece-wise affine warp

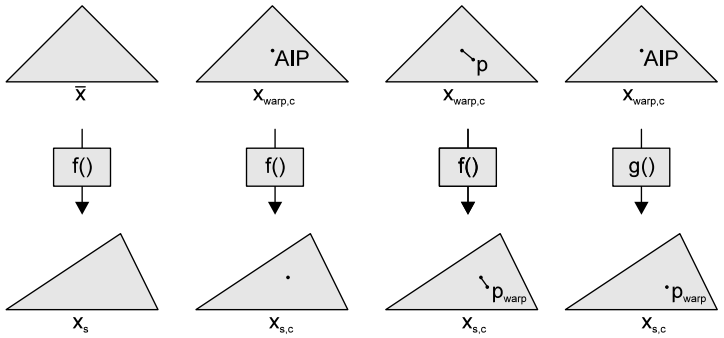


Figure 10.4: Warp modification by FEMs.

it seems natural to choose the AIPs as the centroid of each triangle. The FEM modification of this type of warp function is given in figure 10.5. An appealing alternative to this, is the constrained Delaunay triangulation of Shewchuk [60] which easily could be used to generate AIPs.

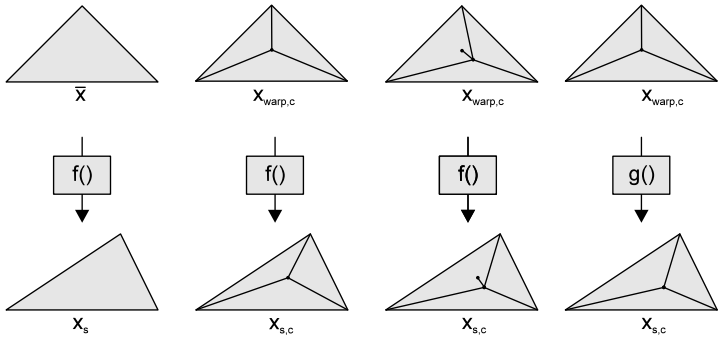


Figure 10.5: Warp modification by FEMs using piece-wise affine warps.

For small FEM deformations, the above description will work out-of-the-box, but for large deformations, one needs to consider the situation where the AIPs are deformed outside the shape (or just outside the corresponding triangle). Then the warp function will break down due to the folding of the surface. To constrain the movements of the AIPs several approaches can be

adapted. The theoretically most pleasing is to include the original shape points into the FEM and giving them a mass of infinity. In the current implementation however, the following approach was taken, the original shape points were included in the FEM with unit mass, the FEM deformation was done and the deformed AIPs were extracted. See figure 10.6. In practice, this method has proven successfully since large deformations never increased the overall fit of the AAM, and consequently were not performed.

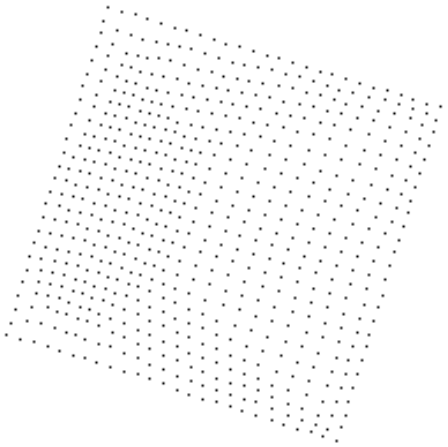


Figure 10.6: A square shape deformed by adding FEM-deformed AIPs and fixating the original outer shape points.

Regarding the integration into the optimization process, several approaches have been tested. First the FEM-parameter adjustment was trained using a multivariate linear regression (MLG) similar to that of the basic AAM. Unfortunately it was discovered that the three MLGs – model parameters, pose parameters and FEM parameters – could not co-exist since the introduction of the FEM parameter prediction made the pose and model parameter predictions erroneous. It is hard to explain why this happens. Its equally hard to explain why the pose and model parameter predictions *can* co-exist "peacefully", since they were trained in an exact similar manner. This could be caused by the *global* behavior of the pose predictions in contrast to the *local* behavior of the model parameter prediction and the FEM parameter prediction.

Consequently, the FEM deformation was integrated into the general-purpose optimization framework instead.

10.6 Results

To assess the effect of the described FEM-modification, an AAM has been built upon 20 metacarpal images (238×272×8bit) with metacarpal 2 annotated using 50 points. A model neighborhood of four mean shape pixels was added.

This model was subsequently used to search four unseen images with a ground truth annotation using automatic initialization. The overall performance was as expected. Using Simulated Annealing (SA) the landmark error decreased from 0.83 pixels to 0.78 pixels (both with SA, but without/with FEM) . The texture error decreased from 4.9 intensities to 4.7 intensities.

While further investigations are required to be conclusive, the results point towards that greater flexibility leads to a better fit.

10.7 Conclusion

The basic AAM has been augmented with a method providing deformation of the inside of a shape without compromising the landmark accuracy of the outer border. The method is based on a finite element framework and integrated into the general-purpose optimization of AAMs (as described in the previous chapter). More work could be done in finding the optimal integration into AAMs. Preliminary experiments showed that using multivariate linear regression prediction in conjunction with the standard AAM search yielded undesirable behavior. In conjunction with a general-purpose optimization method (Simulated Annealing) higher landmark accuracy *and* improved texture fit, was obtained as a result of increased model flexibility.

Part III

Implementation

Chapter 11

The AAM-API

11.1 Overview

This section emphasizes on the details of the AAM implementation. All experiments, illustrations etc. is done by developing an Active Appearance Models Application Programmers Interface (AAM-API) in the C++ language.

The primary motivation for developing a ready-to-use API in C++ contrary to a test bench done in some mathematical framework (i.e. Matlab) is partly due to an agreement with the partial sponsor[1] of this thesis work and partly due to:

Performance Due to the inherent design of AAMs feasible performance could be achieved using C++.

Structure Though the description of AAMs can be done rather brief, an actual implementation requires a substantial amount of machinery. This is proven by the more than 15.000 lines of code in the AAM-API[2]. Consequently, it was of the utmost importance to realize the AAM framework in an environment providing suitable tools for a clear structure of the code.

[1] The Danish company Pronosco A/S.
[2] This is without any 3rd party libraries or auto-generated code.

Reusability Since many of the techniques (warping, shape handling etc.) is of general interest, means of reusability was also set forth as a goal. One of the original design criteria of C++ was reusability.

It was further more the intention to bring the documentation objective to effect right down to source code level by providing the AAM-API as open source. This basically means that other freely can download, use and elaborate on the AAM-API with no additional cost as long as the original author is credited and no headers are removed from the source code. This also motivates the choice of a language which afford extension and modification by inheritance.

To provide a consistent and complete documentation all source code has been augmented with special tags. Within these tags structured code documentation has been written and subsequently extracted by a suitable program[3] into a printable PostScript version and a cross-referenced HTML version. Both of these and further information on the AAM-API can be downloaded in full from the AAM site[4]. Refer to appendix D for further information on the documentation including an exhaustive class listing.

Below the requirements for the AAM-API are given followed by a brief introduction to the most important classes.

11.2 Requirements

The AAM-API is targeted at the Windows NT/9x platform and developed in Microsoft Visual C++ 6.0. Additional functionality was provided by integration of the following 3rd party libraries:

- **BLAS** Fast matrix and vector multiplication.
- **LAPACK** Matrix inversion, eigenvalue decomposition.
- **Intel Math Kernel Library** SIMD implementation of BLAS.
- **VisionSDK** Image, vector and matrix handling etc.
- **DIVA** Image handling, image i/o, vector and matrix handling etc.
- **ImageMagick** Image i/o.

[3] Doc++ [http://www.linuxsupportline.com/~doc++/].
[4] At http://www.imm.dtu.dk/~aam/

These libraries are freely available and needed by the AAM-API[5]. To use AAMC, only VisionSDK DLLs are needed.

11.3 The API at a Glance

Below is enumerated the most important classes of the AAM-API together with a brief introduction. Consult the documentation for a complete and thorough treatment of all classes.

CAAMCore Core functionality of the AAM framework. Includes model building, principal component analysis, linear regression experiments, i/o etc.

CAAMShape Shape container. Path manipulation. Shape expansion and contraction, i/o. Pose transformations. Inside tests. AIP function etc.

CAAMShapeCollection Shape collection container and shape alignment using Procrustes analysis, i/o etc.

CAAMDelaunay Delaunay Triangulator. This is merely a conversion interface to the free Delaunay Triangulator by Steven Fortune. Copyright (c) 1994 by AT&T Bell Laboratories. AT&T, Bell Laboratories.

CAAMLinearReg General multi-variate linear regression on a set of experiments using principal component regression.

CAAMFEMDeform Finite element framework for deforming a set of points, using unit masses. Eigenanalysis is done using LAPACK.

CAAMOptimize General-purpose optimization of the AAM. Interprets the pose and model parameters such that optimization using Steepest Descent, Conjugate Gradient, BFGS, Pattern Search and Simulated Annealing can be accomplished.

CAAMMesh 2D triangular mesh container including hit test, i/o etc.

[5]At least in binary form with header files and stub libraries.

CAAMTriangle Triangle container with built-in hit test.

CAAMWarp Base class of 2D warp classes. Warps are defined in terms of the CAAMShape instances.

CAAMWarpLinear Piece-wise affine warping between two shapes using Delaunay triangulation. Provide optional speed-ups using dynamic programming (caching of warps).

11.4 API Extension by Inheritance

The AAM-API was designed to accommodate extension by inheritance. A natural choice since the API is implemented in the object-oriented language C++. Below is given a simple example of changing functionality of the API, while retaining the possibility of updating core functionality without breaking existing code.

Example: Changing the shape-to-pixel weights

One could desire to change the way that the shape-to-pixel weights are estimated (as mentioned in section 6.2).

```
class CAAMMyCore : public CAAMCore {

public:
    virtual void CalcPixel2ShapeWeights();
};

void CAAMMyCore::CalcPixel2ShapeWeights() {

    // your implementation
}
```

One should however pay attention to the common pitfalls of extension by inheritance such as name collisions in multiple inheritance etc. Consult a textbook in C++ programming for further details.

11.5 Console interface

As an example usage of the AAM-API, a console interface, AAMC, has been provided. Most of the experiments, illustrations etc. were produced using AAMC. This approach has also the advantage of being capable of scripting AAM-API operations using any high-level script-language. As an example has the leave-one-out testing been implemented using the script language provided by command.com in Windows NT and AAMC without any modifications. This has proven to be a highly flexible and fast interface for doing a large number of experiments.

However, it should be simple to provide a graphical user interface on top of the AAM-API, showing eigenmodes etc. with real-time interaction.

11.6 File I/O

As of now much interfacing to the AAM-API consist of files:

AMF – AAM Model File Binary model description containing eigenvalues, eigenvectors, mean shape, mean texture etc. All .amf files has a human-readable equivalent (.txt) providing information about the model.

ACF – AAM Config File ASCII configuration file for setting up the properties of an AAM. Given to the model building process.

ASF – AAM Shape File ACSII representation of a shape including path information etc. The format is tab-separated and can thus be readily converted into virtually any data processing program (Matlab, S-Plus, Excel etc.).

Refer to appendix E for examples on the file formats and appendix G for a detailed format specification of the shape format (.asf).

Part IV

Experimental Results

Chapter 12

Experimental Design

12.1 Methodology

As evaluation of the basic AAM and the proposed enhancements, several training sets were obtained. To prove the alleged generality of AAMs emphasis was put on gathering very different image modalities with objects of very different appearance. This resulted in the following training cases:

- Radiographs of Metacarpals – 24 images/annotations.
- Cardiovascular Magnetic Resonance Images – 45 images/annotations.
- Perspective images of Pork Carcass – 14 images/annotations.

To obtain the best possible evaluation leave-one-out[1] experiments was conducted on all training sets. The substantial increment in workload herein is justified by the removal of all uncertainties caused by dividing the cases into a training set and an evaluation set. Further more, this methodology yield better models due to the capture of almost all variation in each of the three cases; all with a limited amount of images.

[1]Model building where each training example is missed out in turn and evaluated upon.

12.2 Performance Assessment

To assess performance at least two approaches can be adapted. The first approach is to compare the optimized model to a known ground truth, as given by experts. For a survey of this topic, refer to [26]. Secondly, the result could be validated by using the distribution of the training set, which constitutes a very interesting property of AAMs which we will call *self-contained validation* in the following.

12.2.1 Comparison to Ground Truth

Using a ground truth given by a finite set of landmarks for each example performance can easily be assessed. In a leave-one-out setting this could be the same landmarks used for building the models. This calls out for a distance measure, $D(\mathbf{x}_{gt}, \mathbf{x})$, that gives a scalar interpretation of the fit between the two shapes, the ground truth, $\mathbf{x}_{gt}$, and the optimized shape, $\mathbf{x}$.

To assess the performance using landmarks two distance measures could be considered.

- **Point to point error** Defined as the Euclidean distance between each corresponding landmark. Mean pt.pt. error is thus:

$$D_{pt.pt.}(\mathbf{x}_{gt}, \mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(x_i - x_{gt,i})^2 + (y_i - y_{gt,i})^2} \tag{12.1}$$

- **Point to associated border error/Point to curve error** Defined as the Euclidean distance between a landmark of the fitted shape, $\mathbf{x}$, to the closest point on the border given as the linear spline, $\mathbf{r}(t) = (r_x(t), r_y(t))\ ,\ t \in [0;1]$, of the landmarks from the ground truth, $\mathbf{x}_{gt}$. Mean point to associated border error is thus:

$$D_{pt.crv.}(\mathbf{x}_{gt}, \mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\min_{t}\sqrt{(x_i - r_y(t))^2 + (y_i - r_x(t))^2} \tag{12.2}$$

This distance measure is here forth abbreviated to the *point to curve error* (pt.crv.).

The graphical interpretation of these two distance measures is given in figure 12.1.

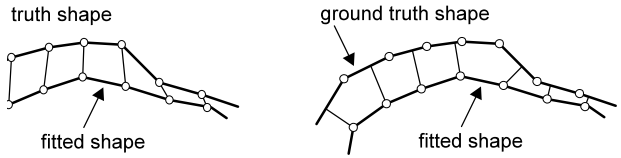


Figure 12.1: Left: Point to point (pt.pt.) error. Right: Point to associated border (pt.crv.) error.

Due to a (common) interest in knowing the outline of the objects rather than the exactly position of landmarks, these were allowed to slide somewhat along the contour of the ground truth in all of the cases. Consequently, only the latter (pt.crv.) distance measure is used in the following.

12.2.2 Self-contained Validation

The term self-contained validation covers the technique of assessing performance using prior knowledge only. In the following, we suggest using the texture error and perform tests in the model parameters.

Texture Error

As in [22] the texture of the optimized model is subtracted from the (normalized) image below and de-normalized. If $\mathbf{g}$ and $\mathbf{g}_d$ denotes normalized and de-normalized texture vectors respectively, the mean de-normalized texture error is:

$$\begin{aligned} t_e(\mathbf{g}, \mathbf{g}_d) &= \frac{1}{m}\sum_{i=1}^{m} |g_{d,model,i} - g_{d,image,i}| \\ &= \frac{1}{m}\sum_{i=1}^{m} |(\alpha g_{model,i} - \beta) - (\alpha g_{image,i} - \beta)| \\ &= \frac{1}{m}\sum_{i=1}^{m} \alpha |g_{model,i} - g_{image,i}| \end{aligned} \tag{12.3}$$

Units are of intensity. This measure is called the *mean intensity error* (mie) and is used extensively in the following to evaluate the texture fit.

Model Deformity

By using prior knowledge from the training set, a probability model for the model parameter configuration can be obtained.

One way to determine if a model parameter configuration is plausible is to impose hard limits on the model parameters, $\mathbf{c}$, under the model-assumption that the $\mathbf{c}$-parameters are independent gaussian distributed with zero mean. Since the variance, σ_i^2, of i^{th} principal component equals λ_i – and 99.73% of distribution of c_i is covered in the range $\pm 3\sigma_i$ – the limits can be chosen as:

$$-3\sqrt{\lambda_i} \leq c_i \leq 3\sqrt{\lambda_i} \tag{12.4}$$

Due to this simple hyper cube restriction every c-parameter is allowed simultaneously to take the value of $\pm 3\sqrt{\lambda_i}$ which is highly unlikely.

To avoid this the $\mathbf{c}$-parameters can be restricted to a hyper ellipsoid using the Mahalanobis distance.

$$D_m^2 = \sum_{i=1}^{t} \frac{c_i^2}{\lambda_i} \leq D_{max}^2 \tag{12.5}$$

such that a D_m is smaller than a suitable D_{max} corresponds to a plausible model instance. As a suitable value for D_{max}, 3.0 could be used.

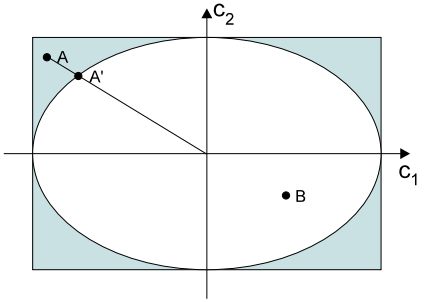


Figure 12.2: The effect of using the Mahalanobis distance in two dimensions. Model instance **B** is valid, while model instance **A** is classified illegal

An even better way to determine this, is to perform a test in the χ^2-distribution, since (12.5) is $\chi^2(t)$ distributed.

12.3 Summary

It was decided to assess AAM performance using leave-one-out evaluation on a set of training examples. For each test the point to curve error, a, the texture error, b and the number of failures c are to be recorded in a table similar to 12.1.

#	**Type** (pixels)	**Pt.-Crv.** (pixels)	**Texture** (mie)	**Failures**
		a	b	c

Table 12.1: Result tabular.

The point to curve error, a, is calculated as the mean of, N experiments each giving the mean point to curve error using shapes with n landmarks:

$$a = \frac{1}{N}\sum_{i=1}^{N} D_{pt.crv.}(\mathbf{x}_{gt,i}, \mathbf{x}_i) \tag{12.6}$$

Likewise the texture error is given as the *mean intensity error* (mie), b, of texture vectors of length m stemming from N experiments:

$$b = \frac{1}{N}\sum_{i=1}^{N} t_e(\mathbf{g}_i, \mathbf{g}_{d,i}) \tag{12.7}$$

Failure, c, was declared when the point to curve error exceeded 10 pixels.

$$c = \frac{1}{N}\sum_{i=1}^{N} [D_{pt.crv.}(\mathbf{x}_{gt,i}, \mathbf{x}_i) > 10.0 \ ? \ 1 \ : \ 0] \tag{12.8}$$

Here formalized using the '?' operator, $e \ ? \ a \ : \ b$, that evaluates the boolean expression, e, and returns a on true and b on false.

Due to the rather small training sets, the uncertainty on the distribution estimations were deemed too high to perform any deformity validation as described in section 12.2.2.

Chapter 13

Radiographs of Metacarpals

13.1 Overview

Radiographs (x-rays) constitute an important image modality in medical imaging. This section focuses on radiographs of hands, from which many medical analyses can be done in a fast and non-invasive manner. These include 1) assessment of skeletal maturity and 2) assessment of bone quality using the *bone mineral density* (BMD) estimate.[1]

Recently image analysis has been applied to estimate BMD and two corrective factors in the BMD calculation namely the *porosity* and *striation*. To accomplish this, segmentation of the metacarpals is required. Refer to figure 13.1 for an atlas of hand anatomy.

However, segmentation in radiographs (x-rays) pose a difficult problem due to large shape variability in human bones and the inherent ambiguity of radiographs. This forms a suitable challenge for an AAM. Other attempts to perform segmentation in radiographs include the work of Efford [25] and Stegmann et al. [66] where the ASM approach was used.

Twenty-four radiographs of different human hands were obtained and three metacarpal bones were annotated using 50 points on each. The annotation

[1] Which is used in the diagnosis of osteoporosis.

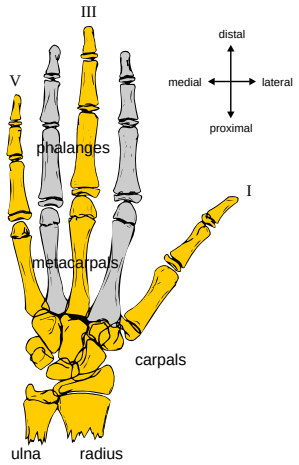


Figure 13.1: Hand anatomy. Metacarpals numbered at the fingertips.

of metacarpals 2, 3 and 4 was concatenated into a 150-point model. Landmarks were extracted from a dense outline representation of the metacarpals using a proprietary algorithm from Pronosco A/S.

13.2 Results

Using leave-one-out and automatic initialization AAM performance has been assessed on radiographs. The 24 images were subsampled to 240×275 pixel to obtain a manageable size, due to the large number of models required in the leave-one-out evaluation. To motivate this decision, each model used approx. 2000 experiments in the linear regression – i.e. 96.000 experiments were conducted to obtain the results below.

Mean results are given in table 13.1 using combinations of the developed enhancements.

The single failure in test one was caused by an erroneous match at metacarpal 3, 4, 5 instead of 2, 3, 4. This was removed before the calculation in table 13.1. One method to resolve this issue would be to include all four

#	**Type** (pixels)	**Pt.-Crv.** (pixels)	**Texture** (mie)	**Failures**
1	Basic AAM	0.88	4.9	1
2	1+Neighborhood	0.84	5.2	0
3	2+SA	0.82	5.0	0
4	3+Lorentzian	0.83	5.0	0

Table 13.1: Leave-one-out test results for the metacarpal AAMs.

Figure 13.2: The mismath at metacarpal 3, 4, 5 instead of 2, 3, 4. in test 1.

metacarpals in the model. Notice however that the fit is rather short in the distal end in figure 13.2. This is a typical example of the previous (in section 9.3) mentioned *shrinking problem*. Consequently, a neighborhood of 4 pixels around the outer border was added to increase texture specificity (test 2 in table 13.1). The result was an overall point accuracy increase of 5 % and elimination of the mismatch. As expected, the texture fit suffered.

The basic AAMs consisted of 150 shape points and approx. 11.000 pixels. Neighborhood AAMs consisted of 300 shape points and approx. 13.000 pixels. In both models, 18 parameters were used to explain 95% of the variation in the training set.

Since test three explicitly optimizes the texture fit, the error was decreased.

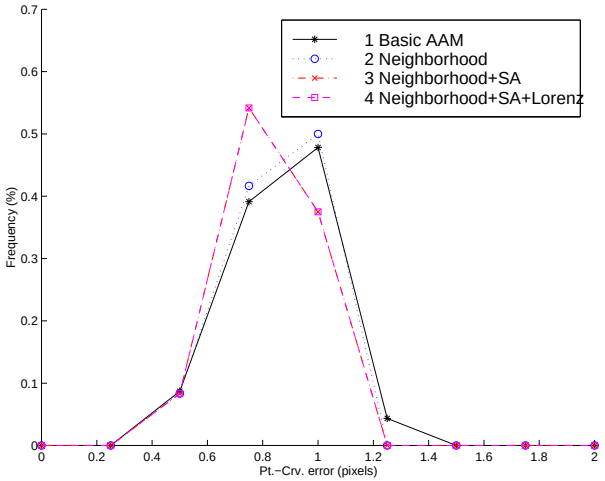


Figure 13.3: Point to curve histograms for radiograph AAMs. Bin size = .25 pixel.

While this was not guaranteed, the landmark accuracy also improved from 0.84 to 0.82. Finally, the Lorenztian error norm was applied in test 4, without any noteworthy improvement. As there were no explicit outliers incorporated into the training images, this result is quite acceptable.

To give a more comprehensive – yet compact – impression of the distribution of the point to curve error of each test, frequency histogram-plots are given in figure 13.3. The outcome of each experiment was subsequent separated into bins, where each covered an error range of .5 pixels. The histogram shows a rather good precision over all leave-one-out experiments without any noteworthy outliers.

To assess the performance *within* points, the mean point to point distance is plotted for a set of evaluations in figure 13.4. Not surprisingly, problems arise in the distal and proximal end of the metacarpals due the large shape variability and the ambiguous nature of radiographs in regions of overlap.

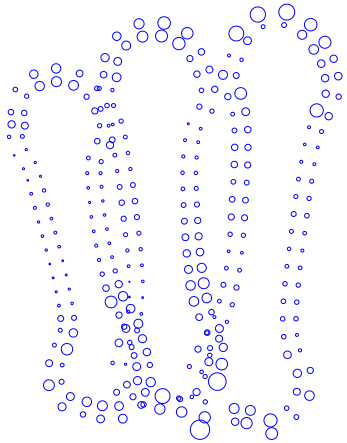
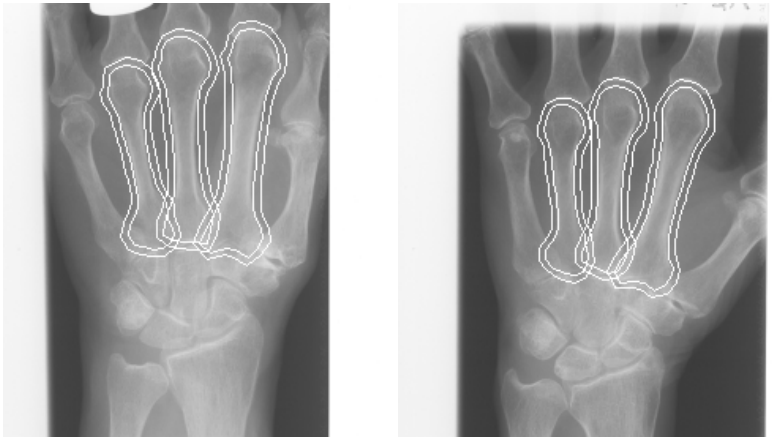


Figure 13.4: Mean point to point deviation from the ground truth annotation of each metacarpal. Low location accuracy is observed at the distal and proximal ends.



(a) (b)

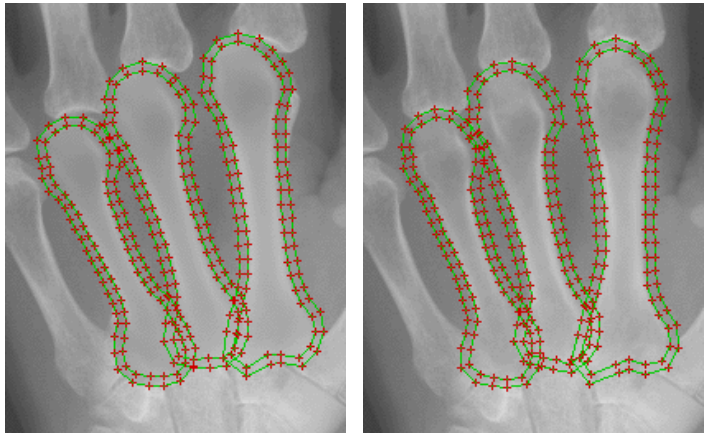
Figure 13.5: Test 3: (a) Worst model fit, 1.01 pixels (pt.crv.). (b) Best model fit, 0.53 pixels (pt.crv.).

Further more to give an impression of the error range, the worst and the best model fit – w.r.t. to landmark accuracy – are given in figure 13.5. Notice a fairly good fit even in the distal (upper) and proximal (lower) end of the metacarpals where radiographs are rather ambiguous. An example showing the full AAM model fit is given in figure 13.6.

For a detailed pictorial documentation of this case, refer to appendix A.

13.3 Summary

The AAM approach has successfully been used to segment metacarpals in radiographs of human hands. By increasing texture specificity using a model neighborhood and a fine-tuning the fit using simulated annealing, leave-one-out evaluation reached a landmark accuracy of 0.82 pixels and a texture error of 5.0 intensities. The automatic initialization method used



(a) (b)

Figure 13.6: (a) AAM after automatic initialization. (b) Optimized AAM. Both cropped to show details.

yielded one failure in the case of models without neighborhood added. No initialization failures were observed when using model neighborhood.

Chapter 14

Cardiac MRIs

Cardiovascular Magnetic Resonance scanning is a very flexible image modality to assess cardiac function in a non-invasive manner. Particularly, multi-slice multi-phase short-axis image views have shown highly useful to examine global and regional cardiac function [52]. To accomplish this, a segmentation of the left-ventricular endocardial and epicardial borders is required. Due to the massive amount of data produced from the 4D Cardiovascular MRIs, automated segmentation is highly desirable. Unfortunately, segmentation in MRIs has shown a very challenging task. This constitutes the primary motivation for applying AAMs to this problem.

Four training sets were obtained using 2D extracts from the original 4D data. Each slice had the resolution of 256×256. The pixel depth was 8 bits. To obtain temporal registration relative to the heart cycle, the image acquisition was triggered by ECG. The endocardial and epicardial contour of the left ventricle were annotated by experts and organized as follows:

Set 1 – Normal hearts

Contain two sets of corresponding slices, from the same heart but at two different spatial locations. The sets were annotated by M.D. Jens Christian Nilsson, H:S Hvidovre Hospital.

A-Slices 14 images, 66-points. Contain papillary muscles, which are small muscles inside the ventricle.

B-Slices 14 images, 66-points. No papillary muscles present.

Set 2 – Abnormal hearts

Contain two sets of corresponding slices, again from the same heart but at two different spatial locations. The sets were annotated by M.D. Bjørn A. Grønning, H:S Hvidovre Hospital.

A-Slices 10 images, 66-points. The slices contain papillary muscles.

B-Slices 7 images, 66-points. No papillary muscles present.

An example of the differences between A and B-slices is given in figure 14.1. Both were taken from Set 1. Due to the large variability and weak image evidence in Set 2 this poses a substantially more challenging task. A severe example showing two different hearts from Set 2 is given in figure 14.2.

Beforehand it was clear that the papillary muscles of the A-slices, posed a challenging problem since their positions seemed rather arbitrarily. This indicates that a free-form deformable template model might perform better given a good initialization. This could for example stem from an AAM.

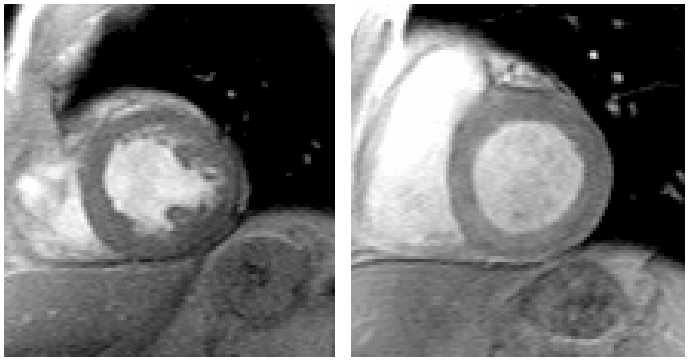


Figure 14.1: Left: Set 1 Cardiac A-slice with papillary muscles. Right: Set 1 Cardiac B-slice without papillary muscles. Both cropped and stretched to enhance features.

AAM segmentation of 2D cardiac MRIs has previously been done by Mitchell et al. [52]. A total of 102 images were used for the training set reaching a

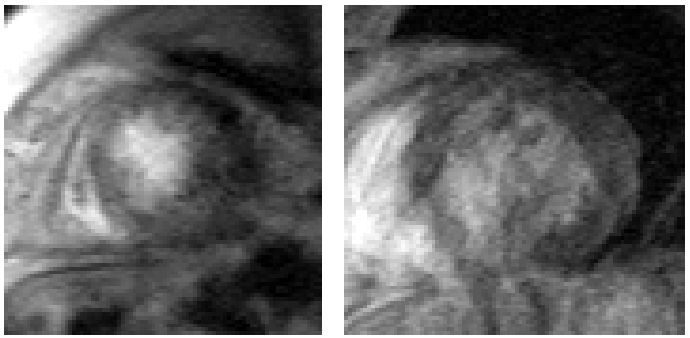


Figure 14.2: Left: Set 2 Cardiac A-slice with papillary muscles. Right: Set 2 Cardiac B-slice without papillary muscles. Both cropped and stretched to enhance features.

mean point accuracy of approx. 1 pixel on the endocardial and epicardial contour. Annotated structures were the right ventricle and endocardial and epicardial contours. Contrary to the following, the model was initialized manually.

14.1 Results

AAMs were built on each of the two set of slices in Set 1 and Set 2 and tested separately using leave-one-out evaluation and automatic initialization on the resulting four models.

The B-slice AAM from Set 1 (here forth 1B) consisted of approx. 2200 pixels. More than 95% of the combined variation was explained using 10 model parameters. As comparison consisted the 1B-model including 3 pixels of neighborhood of approx. 2800 pixels.

The optimization results in given in table 14.1 - 14.4. Neighborhood consisted of adding 3 pixels around the outer border as described in section 9.3.

Not surprisingly, the neighborhood only yielded better results in one of the models in Set 1. This is due to two circumstances. 1) substantial texture variation are already present inside the original shapes and 2) due to the

#	**Type** (pixels)	**Pt.-Crv.** (pixels)	**Texture** (mie)	**Failures**
1	Basic AAM	1.38	9.8	0
2	1+Neighborhood	1.21	10.4	0
3	1+SA	1.37	7.8	0
4	3+Lorentzian	1.32	7.5	0

Table 14.1: Leave-one-out test results for the 14 A-slices of Set 1.

#	**Type** (pixels)	**Pt.-Crv.** (pixels)	**Texture** (mie)	**Failures**
1	Basic AAM	1.18	7.1	0
2	1+Neighborhood	1.73	7.5	0
3	1+SA	1.06	5.9	0
4	3+Lorentzian	1.13	6.0	0

Table 14.2: Leave-one-out test results for the 14 B-slices of Set 1.

#	**Type** (pixels)	**Pt.-Crv.** (pixels)	**Texture** (mie)	**Failures**
1	Basic AAM	3.27	12.1	1

Table 14.3: Leave-one-out test results for the 10 A-slices of Set 2.

#	**Type** (pixels)	**Pt.-Crv.** (pixels)	**Texture** (mie)	**Failures**
1	Basic AAM	3.52	9.1	1

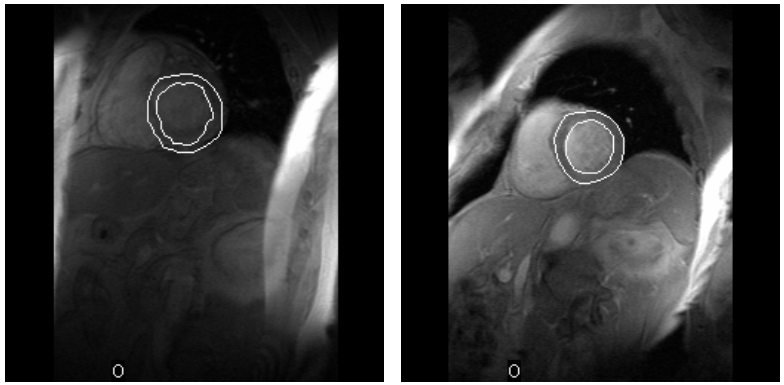
Table 14.4: Leave-one-out test results for the 7 B-slices of Set 2.

varying nature of the MRIs around the ventricle[1] neighborhood adding rather confuses the texture model than making it more specific.

As in the metacarpal case fine-tuning the model fit not only yielded a better texture fit, but also a higher landmark accuracy.

Using the Lorentzian error norm as similarity measure yielded higher landmark accuracy on the A-slices of Set 1, where papillary muscles were present. However lower landmark accuracy was obtained on the B-slices of Set 1, where no papillary muscles were present. Hence when no outliers are present the Lorentzian error norm, yielded lower performance w.r.t. landmark accuracy. This fact suggest that the scale parameter of the robust error norm needs adjustment.

To give an impression of the error range, the worst and the best model fit – w.r.t. to landmark accuracy – are given in figure 14.3. An example showing the full AAM model fit is given in figure 14.7.



(a) (b)

Figure 14.3: Test 1 on B-slices of Set 1: (a) Worst model fit, 2.43 pixels (pt.crv.). (b) Best model fit, 0.65 pixels (pt.crv.).

As in the metacarpal section, a more comprehensive impression of distribution of the point to curve error of each model, is given as histogram-plots in figure 14.4, 14.5 and 14.6. However, the rather crude resolution due to small number of experiments, it should still be possible to assess the relative performance. As expected the plots shows that the performance of Set 1 images is significantly higher than that of Set 2 images. The two outliers of Set 2 were discarded before assessment.

[1]This could for example be hearts where the epicardial boundary is embedded in fatty tissue.

For a detailed pictorial documentation of this case, refer to appendix A.

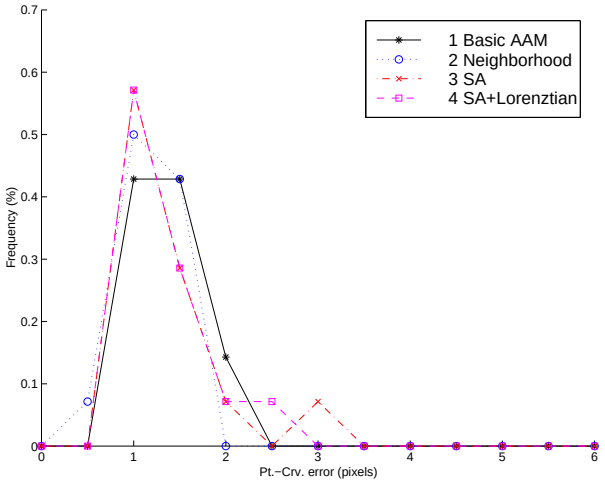


Figure 14.4: Point to curve histograms for the AAMs built on A-slices from Set 1. Bin size = .5 pixel.

14.2 Summary

AAMs have been applied successfully on Cardiac MRIs of normal hearts (Set 1) using automatic initialization. Fine-tuning of the model fit using simulated annealing increased both texture fit and landmark accuracy yielding a mean landmark accuracy of 1.37 pixels and a mean texture error of 7.8 for slices with papillary muscles. For slices without papillary muscles, a landmark accuracy of 1.02 pixels and a texture error of 5.9 were yielded.

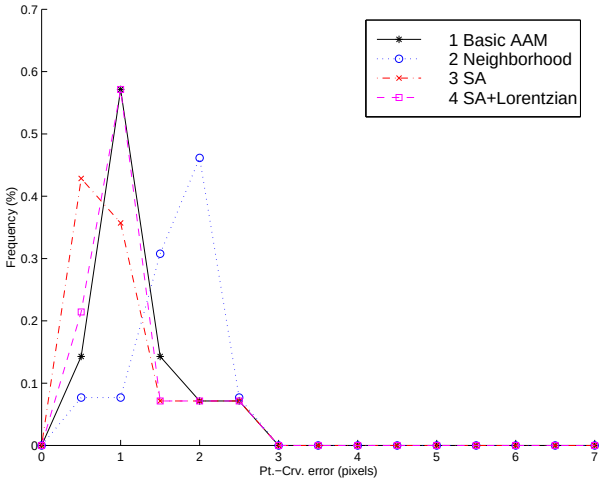


Figure 14.5: Point to curve histograms for the AAMs built on B-slices from Set 1. Bin size = .5 pixel.

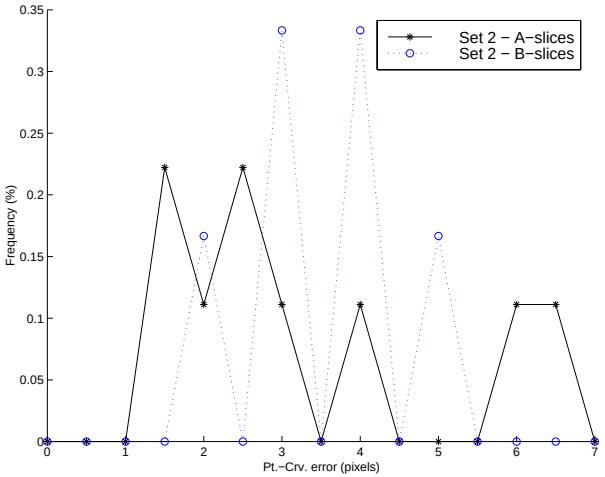
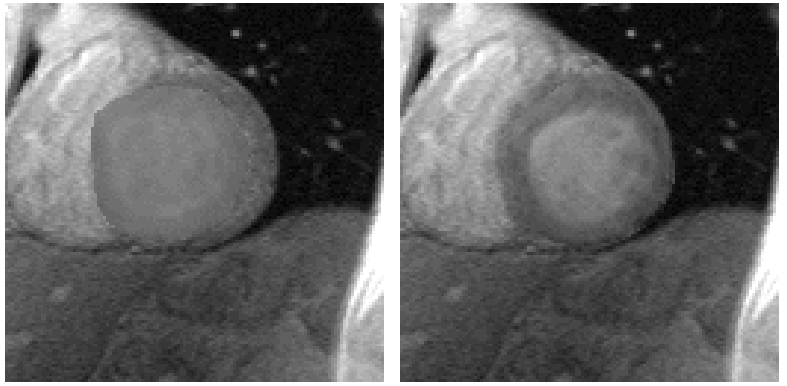


Figure 14.6: Point to curve histograms for the AAMs built on A- and B-slices from Set 2. Bin size = .5 pixel.



(a) (b)

Figure 14.7: A: AAM after automatic initialization. B: Optimized AAM. Both cropped to show details.

No initialization failures were observed on the 28 images.

In Cardiac MRIs of abnormal hearts, the basic AAMs yielded a landmark accuracy of 3.27/3.52 pixels and a texture error of 12.1/9.1, respectively on slices with and without papillary muscles. In this case, the initialization failed twice on the 17 images given.

Chapter 15

Cross-sections of Pork Carcass

As the final case study, perspective images of pork carcass cross-sections are presented. This case is chosen due to the difference in image modality and object behavior. Since this type of meat contains a complex structure of fat and pure meat, these images made a fine contrast to the previous cases. Furthermore, the very flexible nature of the meat-slices gave rise to a substantially higher degree of shape variation than seen in the previously presented radiographs and MRIs.

A training set of 14 images was annotated by a dense outline and 83 landmarks were extracted using the technique of Duta et al. [21]. The image size was 256×191 pixels.

Previous results using this training set have been reported by Fisker et al. [30]. Using the Grenander Model [36] on this set reached a landmark accuracy of 1.02 pixels (pt.crv.).

15.1 Results

Mean results are given in table 15.1 using combinations of the developed enhancements. All experiments were done using leave-one-out and automatic initialization.

#	**Type** (pixels)	**Pt.-Crv.** (pixels)	**Texture** (mie)	**Failures**
1	Basic AAM	1.12	13.2	0
2	1+Neighborhood	0.91	13.9	0
3	2+SA	0.89	13.6	0
4	3+Lorentzian	0.91	13.6	0
5	Border AAM	0.86	23.5	0

Table 15.1: Leave-one-out test results for the pork carcass AAM.

To reduce the penalty from the large-scale texture noise inside the shapes a Border AAM was applied in test 5. This increased the landmark accuracy by 23% over the accuracy of basic AAMs. Notice that the texture error of 23.5 intensities is not comparable to the other texture errors, since a completely different texture model was used.

The basic AAMs consisted of 83 shape points and approx. 13.000 pixels. Neighborhood AAMs consisted of 166 shape points and approx. 15.000 pixels. The Border AAM consisted of 249 points and approx. 3500 pixels. In all models 11 parameters explained more than 95% of the variation in the training set.

Point to curve frequency histogram-plots of all five tests are given in figure 15.1. From this, the increased landmark accuracy of the Border AAMs stands out.

Furthermore, to give an impression of the error range, the worst and the best model fit – w.r.t. to landmark accuracy – are given in figure 15.2.

For a detailed pictorial documentation of this case, refer to appendix A.

15.2 Summary

AAMs has successfully been used to segment pork carcass. By removing large-scale texture noise inside shapes using a Border AAM leave-one-out evaluation reached a landmark accuracy of 0.86 pixel. This was an increment of 23% over basic AAMs. The texture error for basic AAMs was 13.2 intensities. The automatic initialization method yielded no failures in any of the 14 images.

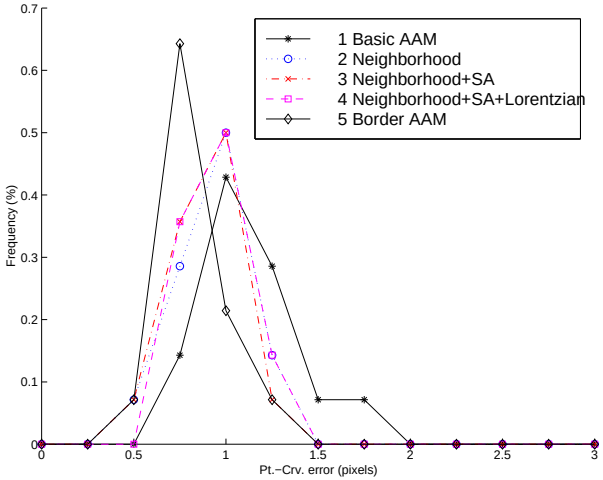
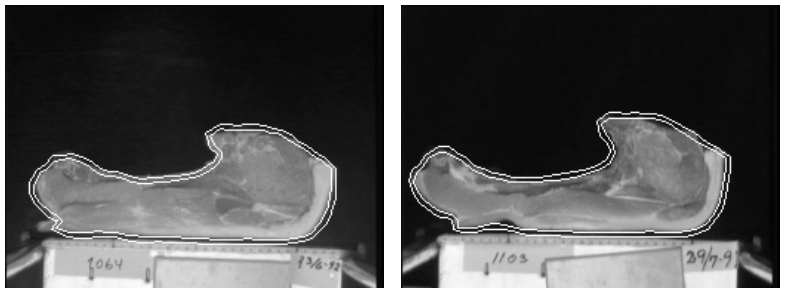


Figure 15.1: Point to curve histograms for different pork carcass AAMs. Bin size = .25 pixel.



(a) (b)

Figure 15.2: Test 3: (a) Worst model fit, 1.34 pixels (pt.crv.). (b) Best model fit, 0.60 pixels (pt.crv.).

Part V

Discussion

Chapter 16

Propositions for Further Work

16.1 Overview

The following chapter serves as an appetizer, discussing ideas developed during this six month master thesis work, that either were out of the scope or out of reach within the given time span.

16.2 Robust Model Building

Taking the ideas presented in section 9.8 further, one could think of many steps in the model building process, where robust statistics could be utilized.

Notice however that AAMs are built upon a selection of representative examples, which makes the use of robust methods less crucial in the model. Human outlier rejections have so to speak already been done. Robust methods probably will – nevertheless – still increase the quality of the model.

Alignment using the Procrustes Distance

Recall that minimizing the Procrustes Distance basically is a least-squares adjustment – i.e. a quadratic norm is used. Alignment using robust norms could be used here instead. This would enable handling of point annihilation if redescending estimators[1] [3] are used.

For a short survey of attempts using robust estimators in the shape alignment, refer to Dryden and Mardia [20].

Mean Shape Estimation

Estimation of the mean shape is currently accomplished by finding the shape that minimizes the summed square distance to all shapes. This could also be improved by robust statistics. On a per-point basis as mentioned above and on a per-shape basis.

16.3 Active Texture Weighting

Given a texture model used in AAMs one could desire means of controlling the importance of each pixel in the model. This could for example be due to prior or estimated knowledge about the reliability of each pixel – i.e. the pixel-quality used in a similarity driven setup such as AAMs. Therefore we propose to include a weighting vector, $\mathbf{w}_p$ into the texture model. In this way each incoming texture sample, $\mathbf{g}$, would be included into the model as $\mathbf{g}_w$. This is done both in the model building phase and in the optimization. If m denotes the number of texture samples this is:

$$\mathbf{g}_w = \begin{bmatrix} w_{p,1} & & 0 \\ & \ddots & \\ 0 & & w_{p,m} \end{bmatrix} \mathbf{g} \tag{16.1}$$

What weights to use are rather application specific. Example usage's include:

[1] Error norms where the first derivative – the so-called *influence function* [3] – goes to zero for gross errors.

- Enhanced shape flexibility using the weight vector $\mathbf{w}_p$ to partially or totally include exclude areas in the shape. Think of defining $\mathbf{w}_p$ as drawing an semi-transparent or opaque mask in a drawing program using an airbrush-like tool.

- Exclusion of regions with large variability which only lead to less compact texture models. This could be accomplished by down-weighting pixel with high variance. Any function could be designed to take care of this. For suitable flexibility we suggest using:

$$w_i = f(\sigma_i^2) = \begin{cases} 1 & , \ \sigma_i^2 < \beta \\ cos^{\alpha}(x-\beta) & , \ \beta \leq \sigma_i^2 \leq \frac{\pi}{2} + \beta \\ 0 & , \ \sigma_i^2 > \frac{\pi}{2} + \beta \end{cases} \tag{16.2}$$

  where σ_i^2 denotes the variance of the i^{th} pixel over the training set and β controls where the down weighting should begin and α controls the rate of decay. Hard thresholds at β could then be obtained using a high value for α.

We anticipate that active texture weighting would improve flexibility and lead to better texture models.

16.4 Relaxation of Shape Constraints

This section is merely a reminder to the fact, that a similar approach to the one of Cootes et al. [11] applied on ASMs, could be used if the shape variation in AAMs is over-constrained by a small or non-representative training set.

The shape covariance structure is simply augmented with a covariance structure obtained from a finite element model.[2] This provides a more flexible model – yet still globally constrained as opposed to Snakes [46].

Quite recently[3] however, a more Snake-like approach has been proposed as an extension to AAMs [13]. In [13] this was obtained using so-called

[2] Using a suitable weighting – e.g. a function of the number of examples in the training set.

[3] ECCV Dublin, June 2000.

local AAMs – i.e. AAM-blobs around each landmark which could deform semi-separately under a smoothness constraint w.r.t. position.

16.5 Scale-Space Extension

Cootes et al. [14] suggest to use a multi-resolution pyramidal framework for representing AAMs thus gaining speed and robustness in the AAM search.

This should however not be confused with a *scale-space* [50] implementation of AAMs. In scale-space each image is extended by the scale parameter, t, that controls the variance of a gaussian kernel which is convoluted with the image. Formally, for any k-dimensional signal, $f : \mathbb{R}^k \rightarrow \mathbb{R}$, its scale space representation $L : \mathbb{R}^k \times \mathbb{R}_+ \rightarrow \mathbb{R}$ is defined by:

$$L(x;t) = \int_{\xi \in \mathbb{R}^k} f(x-\xi) g(\xi, t) d\xi \tag{16.3}$$

where $g : \mathbb{R}^k \times \mathbb{R}_+ \rightarrow$ denotes the Gaussian kernel:

$$g(x;t) = \frac{1}{(2\pi t)^{\frac{1}{2}k}} e^{-\frac{x_1^2 + \cdots + x_k^2}{2t}} \tag{16.4}$$

This approach has several interesting mathematical properties. [50] stress that an inherent property of real-world objects, is that they only exists as meaningful entities over a certain range of scale. Further more has the use of scale-space a substantially damping effect on the problems stemming from the discreet nature of digital images – i.e. it justifies the use of many assumptions from continuous mathematics.

In the AAM case, scale-space would for example increase the smoothness of the search space, and thus substantially ease the optimizations of AAMs.

Consequently it would be interesting to build AAMs for varying t's and combine them into one scale-space AAM.

Chapter 17

Perspectives of AAMs

17.1 AAMs in 3D

As the techniques used in AAMs extends to 3D we would anticipate to see future work in this direction since acquisition of 3D images is an interesting and rapidly growing field. As pointed out by Jain et al. [44] 3D deformable models are becoming increasingly popular in medical applications due to the many 3D imaging modalities such as MRI, fMRI, CT and recently; 3D-ultrasound scanning.

The simple version of extending AAMs into 3D is to treat each 2D slice as an independent model, thereby concatenating the result of each 2D AAM into a 3D result. However, this basically disregards all coherent information across slices, which can be used to regularize the model substantially, which would lead to better models. The alignment and PCA shape framework is – if not readily, then through minor modifications – extendable to 3D. Regarding the texture model, has the Delaunay triangulation been generalized into n-dimensions, though the generation is somewhat more cumbersome [69]. Therefore, although the practical circumstances may make the implementation and acquisition of landmarks semi-infeasible, the theoretical framework should be present to extend AAMs into 3D.

Initial work on extending AAMs to 3D has been done by Wolstenholme et al. [72], where it was shown that wavelet compression successfully could

be used to reduce the memory requirements without noteworthy loss of quality. The texture PCA was simply performed on wavelet coefficients instead of the raw pixels. Reliable image interpretation was obtained at a compression ratio of 20:1.[1] We emphasize that a 3D AAM has not been built, but rather important preliminary work in making 3D AAMs feasible was done.

During two talks given by Professor Milan Sonka in spring and summer 2000[2], it was indicated that his group at University of Iowa are working toward 3D AAMs. Their work on 2D Cardiac AAMs by Mitchell et al. is described in [52].

When extending AAMs to 3D one should be aware of the close connection between deformable models and image registration. Especially many 3D-image registration techniques applied on medical problems could serve as inspiration.

For a survey on 3D-image segmentation using deformable models, refer to McInerney and Terzopoulos [51].

17.2 Multivariate Imagery

The trend in image acquisition as of now, is that both the spatial, temporal and radiometric[3] resolution increases. On top of that the spectral resolution also starts to increase – i.e. the number of spectral bands per image increases towards a full electromagnetic spectrum for each pixel. The most well known transition is that of the single-band gray scale to the three-band RGB.

In [22] it is shown how AAMs can accommodate such increments in the spectral structure of input data. Specificity is increased in the texture model by incorporating color. Each band is simply concatenated in the texture vector as:

$$\mathbf{g} = \begin{pmatrix} g_{r,1}, g_{r,2}, \ldots, g_{r,m}, \\ g_{g,1}, g_{g,2}, \ldots, g_{g,m}, \\ g_{b,1}, g_{b,2}, \ldots, g_{b,m} \end{pmatrix}^{\mathrm{T}} \tag{17.1}$$

[1] On top of this; the spatial filtering of wavelets is also quite desirable in multi-resolution frameworks such as the AAMs.

[2] At Herlev Hospital and IMM, DTU respectively.

[3] The amplitude accuracy – i.e. number of bits per pixel in digital imagery.

where $g_{r,i}$, $g_{g,i}$, $g_{b,i}$ denotes the intensities of red, green and blue respectively. Here forth the texture analysis proceeds unchanged.

The above technique can thus be used to incorporate any number of bands stemming from either multivariate imagery or the addition of artificial feature bands. Such feature bands could for example be the output of linear filters such as Sobel, Laplacian etc. or non-linear methods such as mathematical morphology operations, Canny edge detection etc.

Chapter 18

Discussion

18.1 Summary of Main Contributions

The main objectives set forth was:

- Discuss, document and explore the basic AAM.
- Design general extensions to the AAM approach.
- Evaluate AAMs through a set of relevant and varying cases.

In this thesis, the Active Appearance Models have been described in detail. It has been intended to make this treatment as rich as possible on discussions, illustrative examples and references for further investigation. Thus fulfilling the first of the three major objectives of this thesis.

Regarding the second objective – design of extensions of general use, several have been proposed. Among these are:

- Enhanced shape representation.
- Handling of homogeneous objects.
- Handling of heterogeneous objects.
- General and robust automated initialization.
- Fine-tuning of the model fit using Simulated Annealing etc.
- Applying robust statistics to the optimization.
- Unification of Finite Element Models and AAMs.

All of the proposed extensions have been shown to have a positive effect on both landmark accuracy and the texture fit. Though the unification of finite element models and AAMs yielded improved accuracy, more work is needed to find the optimal integration into AAMs.

The third objective concerned validation of Active Appearance Models in general and the designed extensions in specific. To fulfil this objective an evaluation methodology for AAMs has been designed and applied on three cases with largely varying segmentation problems and image modalities:

- Radiographs of Metacarpals.
- Cardiovascular Magnetic Resonance Images.
- Perspective images of Pork Carcasses.

In two of the three cases subpixel landmark accuracy was obtained using the designed extensions. In the pork carcass case, use of the designed extensions increased landmark accuracy by 23%.

Finally, a structured, high performance, open source implementation of AAMs (and the designed extensions) has been developed. This is named the AAM-API. The motivation for this work was to ensure further development on AAMs by providing an open and well-documented platform for education and research.

As concluding remark the need for "gold standard" implementations and "gold standard" training sets, can not be stressed too much. This is the utmost fastest way to make progress in the field of image segmentation. As in many other fields for that matter. This thesis represents introductory work towards this situation as all material produced have been made publicly available.

18.2 Conclusion

Computer vision spans a wide range of problems. This calls out for general solutions – i.e. techniques that span the largest possible subspace of all problems known.

In this thesis, a general model-based vision technique has been presented. In agreement with the constructivist theorists of cognitive psychology it learns through observation.

The technique has been thoroughly studied, documented and extended. Subsequently it has been presented with real-world observations in the form of training sets. This produced a set of models capturing the presented knowledge of shape and texture, which subsequently have been applied to unseen problems of the same class.

Using a developed initialization technique and a combination of the proposed extensions, subpixel accuracy was obtained in two of three cases w.r.t. object segmentation.

All proposed extensions yielded higher segmentation accuracy, when applied to the type of problems that they addressed. Both landmark accuracy and texture fit were increased using the proposed extensions.

The objects in the three cases were, human bones (metacarpals), human hearts (left ventricle) and slices of meat (pork carcass).

It has been shown that Active Appearance Models with the developed extensions – as a fully automated and data-driven model – can perform segmentation in challenging image modalities. A thorough evaluation has shown that this can be done with very high accuracy.

Bibliography

[1] *The American Heritage Dictionary of the English Language,* 3rd *Edition.*

[2] P. R. Andresen and M. Nielsen. Non-rigid registration by geometry-constrained diffusion. *Lecture Notes in Computer Science*, 1679:533–543, 1999.

[3] M. J. Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *Int. Journal of Computer Vision*, 19(1):57–92, 1996.

[4] A. Blake and M. Isard. *Active Contours.* Springer, 1998.

[5] F. L. Bookstein. Principal warps: thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):567–85, 1989.

[6] F. L. Bookstein. Landmark methods for forms without landmarks: localizing group differences in outline shape. *Medical Image Analysis*, 1(3):225–244, 1997.

[7] V. Cerny. Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *Jour. of Optimization Theory and Applications*, 45:41–51, 1985.

[8] T. F. Cootes, C. Beeston, G.J. Edwards, and C. J. Taylor. A unified framework for atlas matching using active appearance models. In *Information Processing in Medical Imaging. 16th Int. Conf., IPMI'99. Proc.*, pages 322–33. Springer-Verlag, 1999.

[9] T. F. Cootes, G. Edwards, and C. J. Taylor. A comparative evaluation of active appearance model algorithms. In *BMVC 98. Proc.of the Ninth British Machine Vision Conf.*, volume 2, pages 680–689. Univ. Southampton, 1998.

[10] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In *Proc. European Conf. on Computer Vision*, volume 2, pages 484–498. Springer, 1998.

[11] T. F. Cootes and C. J. Taylor. Combining point distribution models with shape models based on finite element analysis. *Image and Vision Computing*, 13(5):403–9, 1995.

[12] T. F. Cootes and C. J. Taylor. A mixture model for representing shape variation. *Image and Vision Computing*, 17(8):567–574, 1999.

[13] T. F. Cootes and C. J. Taylor. Combining elastic and statistical models of appearance variation. In *Proc. European Conf. on Computer Vision*, volume 1, pages 149–163, 2000.

[14] T. F. Cootes and C. J Taylor. *Statistical Models of Appearance for Computer Vision.* Tech. Report , University of Manchester, http://www.isbe.man.ac.uk/∼bim/, Feb. 2000.

[15] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models - their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, 1995.

[16] T. F. Cootes, K. Walker, and C. J. Taylor. View-based active appearance models. In *Proc. 4th IEEE Int. Conf. on Automatic Face and Gesture Recognition*, pages 227–32. IEEE Comput. Soc, 2000.

[17] T.F. Cootes, G. J. Edwards, and C. J. Taylor. Comparing active shape models with active appearance models. In *Proc. British Machine Vision Conf.*, pages 173–182, 1999.

[18] N. Costen, T. Cootes, G. Edwards, and C. Taylor. Simultaneous extraction of functional face subspaces. In *Proc. of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 492–497. IEEE, 1999.

[19] J. E. Dennis and R. B. Schnabel. *Numerical Methods For Unconstrained Optimization and Nonlinear equations.* Prentice-Hall, 1983.

[20] I. L. Dryden and K. V. Mardia. *Statistical Shape Analysis.* John Wiley & Sons, 1998.

[21] N. Duta, A. K. Jain, and M.-P. Dubuisson-Jolly. Learning 2D shape models. In *Proc. Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 8–14, 1999.

[22] G. J. Edwards, T.F. Cootes, and C. J. Taylor. Advances in active appearance models. In *Proc. Int. Conf. on Computer Vision*, pages 137–142, 1999.

[23] G.J. Edwards, C. J. Taylor, and T. F. Cootes. Interpreting face images using active appearance models. In *Proc. 3rd IEEE Int. Conf. on Automatic Face and Gesture Recognition*, pages 300–5. IEEE Comput.

Soc, 1998.

[24] G.J. Edwards, C. J. Taylor, and T. F. Cootes. Learning to identify and track faces in image sequences. In *6th Int. Conf. on Computer Vision*, pages 317–22. Narosa Publishing House, 1998.

[25] N. D. Efford. *Knowledge-Based Segmentation and Feature analysis of Hand Wrist Radiographs.* Tech. Report, University of Leeds, 1994.

[26] R. Fisker. *Making Deformable Template Models Operational.* PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, 2000.

[27] R. Fisker, J. M. Carstensen, M.F. Hansen, F. Bødker, and S. Mørup. Estimation of nanoparticle size distributions by image analysis. *Jour. of Nanoparticle Research.* To appear.

[28] R. Fisker, J. M. Carstensen, and K. Madsen. Initialization and optimization of deformable models. In *Proc. 11th. Scandinavian Conf. on Image Analysis*, pages 295–302, 1999.

[29] R. Fisker, N. Schultz, N. Duta, and J. M. Carstensen. A general scheme for training and optimization of the Grenander deformable template model. In *Proc. Conf. on Computer Vision and Pattern Recognition*, volume I, pages 698–705, 2000.

[30] R. Fisker, N. Schultz, N. Duta, and J. M. Carstensen. The grenander deformable template model: A general scheme. 2000. Submitted.

[31] R. Fletcher. *Practical Methods of Optimization.* John Wiley & Sons, 1987.

[32] J. D. Foley, A. Dam, S. K. Feiner, and J. F. Hughes, editors. *Computer Graphics: Principles and Practice, 2. Edition.* Addison-Wesley, 1992.

[33] C. A. Glasbey and K. V. Mardia. A review of image-warping methods. *Journal of Applied Statistics*, 25(2):155–172, 1998.

[34] M. Gleicher. Projective registration with difference decomposition. In *Proc. 1997 Conf. on Computer Vision and Pattern Recognition*, pages 331–337. IEEE Comput. Soc, 1997.

[35] C. Goodall. Procrustes methods in the statistical analysis of shape. *Jour. Royal Statistical Society, Series B*, 53:285–339, 1991.

[36] U. Grenander, Y. Chow, and D. M. Keenan. *Hands: A Pattern Theoretic Study of Biological Shapes.* Springer, 1991.

[37] T. Heap and Samaria F. Real-time hand tracking and gesture recognition using smart snakes. *Technical Report, Olivetti Research Limited, Cambridge CB2 1QA, UK*, 1995.

[38] T. Heap and D. Hogg. Extending the point distribution model using polar coordinates. pages 130–7. Springer-Verlag, 1995.

[39] A. Hill, T. F. Cootes, and C. J. Taylor. A generic system for image interpretation using flexible templates. In *BMVC92. Proceedings of the British Machine Vision Conference*, pages 276–85. Springer-Verlag, 1992.

[40] R. Hooke and T. A. Jeeves. Direct search: solution of numerical and statistical problems. *Jour. Assoc. Comput.*, 8(212-229), 1961.

[41] B.K.P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A (Optics and Image Science)*, 4(4):629–42, 1987.

[42] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(9):850–863, 1993.

[43] J. Isidoro and S. Sclaroff. Active voodoo dolls: a vision based input device for nonrigid control. In *Proc. Computer Animation '98*, pages 137–143. IEEE Comput. Soc, 1998.

[44] A. K. Jain, Y. Zhong, and M.-P. Dubuisson-Jolly. Deformable template models: A review. *Signal Processing*, 71(2):109–129, 1998.

[45] A. K. Jain, Y. Zhong, and S. Lakshmanan. Object matching using deformable templates. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(3):267–278, 1996.

[46] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Int. Jour. of Computer Vision*, 8(2):321–331, 1988.

[47] S. Kirkpatrick, C. D. Gellant, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[48] A. Lanitis, C. J. Taylor, and T. Cootes. Automatic interpretation and coding of face images using flexible models. *IEEE Trans. of Pattern recognition and Machine Intelligence*, 19(7):743–756, 1997.

[49] A. Lanitis, C. J. Taylor, and T. F. Cootes. Modeling the process of ageing in face images. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pages 131–6. IEEE Comput. Soc., 1999.

[50] T. Lindeberg. Scale-space: A framework for handling image structures at multiple scales. In *CERN School of Computing. Proceedings.* CERN, 1996.

[51] T. McInerney and D. Terzopoulos. Deformable models in medical image analysis: a survey. *Medical Image Analysis*, 2(1):91–108, 1996.

[52] S. Mitchell, B. Lelieveldt, R. Geest, J. Schaap, J. Reiber, and M. Sonka. Segmentation of cardiac mr images: An active appearance model approach. In *Medical Imaging 2000: Image Processing,*

San Diego CA, SPIE, volume 1. SPIE, 2000.

[53] J.L. Mundy. Object recognition based on geometry: Progress over three decades. *Philosophical Transactions of the Royal Society London, Series A (Mathematical, Physical and Engineering Sciences)*, 356(1740):1213–1231, 1998.

[54] A. A. Nielsen. *Analysis of Regularly and Irregularly Sampled Spatial, Multivariate, and Multi-temporal Data*. PhD thesis, Institute of Mathematical Modelling, Technical University of Denmark, Lyngby, 1994.

[55] F. Preparata and Shamos M. *Computational Geometry*. Springer, 1986.

[56] J. Price, Y. Rogers, H. Sharp, D. Benyon, S. Holland, and T. Carey, editors. *Human-Computer Interaction*. Addison-Wesley, 1994.

[57] J. O. Rawlings. *Applied Regression Analysis*. Wadsworth & Brooks/Cole, 1988.

[58] S. Sclaroff and J. Isidoro. Active blobs. *Proc. of the Int. Conf. on Comput. Vision*, pages 1146–1153, 1998.

[59] S. Sclaroff and A. P. Pentland. Modal matching for correspondence and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(7):545–61, 1995.

[60] J.R. Shewchuk. Triangle: engineering a 2D quality mesh generator and Delaunay triangulator. In *Applied Computational Geometry. FCRC'96 Workshop.*, pages 203–222. Springer-Verlag, 1996.

[61] M. Sonka, V. Hlavac, and R. Boyle. *Image processing, analysis and machine vision*. Chapman & Hall, 1993.

[62] Milan Sonka. Lecture given at Herlev Hospital May 30th, 2000.

[63] P. D. Sozou, T. F. Cootes, C. J. Taylor, E. C. Di Mauro, and A. Lanitis. Non-linear point distribution modelling using a multi-layer perceptron. *Image and Vision Computing*, 15(6):457–63, 1997.

[64] P.D. Sozou, T.F. Cootes, C.J. Taylor, and E.C. Di Mauro. Non-linear generalization of point distribution models using polynomial regression. *Image and Vision Computing*, 13(5):451–7, 1995.

[65] M. B. Stegmann. Active appearance models: Theory, extensions and cases. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, 2000. http://www.imm.dtu.dk/∼aam/.

[66] M. B. Stegmann, R. Fisker, and B. K. Ersbøll. *On Properties of Active Shape Models*. Informatics and Mathematical Modelling, Technical University of Denmark, 2000.

[67] M. B. Stegmann, R. Fisker, B. K. Ersbø ll, H. H. Thodberg, and L. Hyldstrup. Active appearance models: Theory and cases. In *Proc. 9th Danish Conference on Pattern Recognition and Image Analysis, Aalborg, Denmark*, volume 1, pages 49–57. AUC, 2000.

[68] C. Studholme, D. L. G. Hill, and D. J. Hawkes. An overlap invariant entropy measure of 3D medical image alignment. *Pattern Recognition*, 32(1):71–86, 1999.

[69] K Sugihara and H. Inagaki. Why is the 3d delaunay triangulation difficult to construct? *Information Processing Letters*, 54(5):275–280, 1995.

[70] P. Viola and W. M. Wells III. Alignment by maximization of mutual information. *International Journal of Computer Vision*, 24(2):137–154, 1997.

[71] K.N. Walker, T. F. Cootes, and C. J. Taylor. Determining correspondences for statistical models of facial appearance. In *Proc. Fourth IEEE Int. Conf. on Automatic Face and Gesture Recognition*, pages 271–6. IEEE Comput. Soc, 2000.

[72] C.B.H. Wolstenholme and C.J. Taylor. Wavelet compression of active appearance models. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI'99*, pages 544–554, 1999.

[73] A. L. Yuille, P. W. Hallinan, and D. S. Cohen. Feature extraction from faces using deformable templates. *Int. Jour. of Computer Vision*, 8(2):99–111, 1992.

Index

Appendix A

Detailed Model Information

In the following pages, one model per case is documented by plots of:

- Point cloud of the unaligned annotations.
- Point cloud of the aligned annotations.
- Delaunay triangulation of the mean shape.
- Independent principal component analysis of each model point.
- Mean shape deformation using 1st, 2nd and 3rd principal mode.
- Shape eigenvalues in descending order.
- PC1 ($b_{s,1}$) vs. PC2 ($b_{s,2}$) in the shape PCA.
- Texture eigenvalues in descending order.
- PC1 ($b_{g,1}$) versus PC2 ($b_{g,2}$) in the texture PCA.
- Correlation matrix of the annotations.
- Texture variance.
- Combined eigenvalues.

This is done to give a complete pictorial impression of the annotations and the subsequent shape and texture analysis. This appendix should be useful for both education as well as for further research.

A.1 Radiographs of Metacarpals

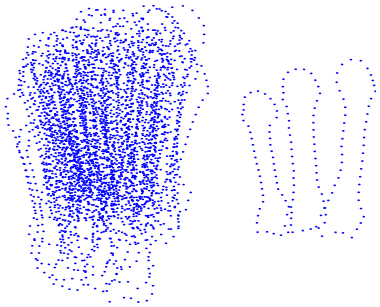


Figure A.1: Point cloud of the unaligned annotations.

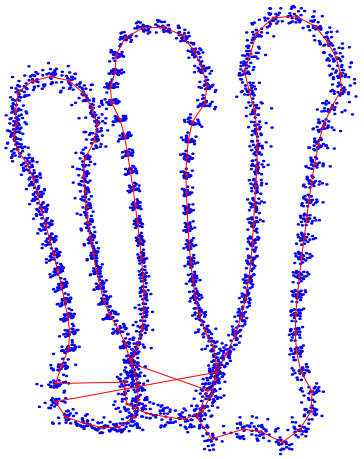


Figure A.2: Point cloud of the aligned annotations with mean shape fully drawn.

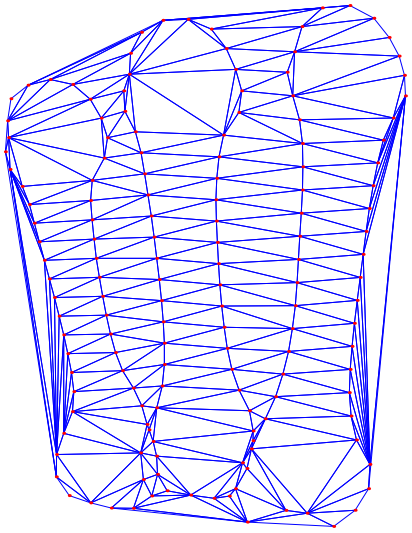


Figure A.3: Delaunay triangulation of the mean shape.

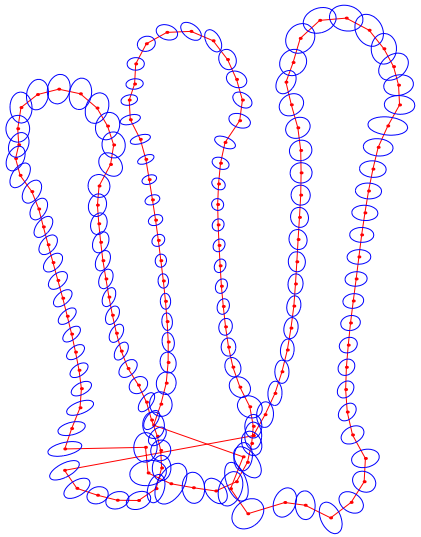
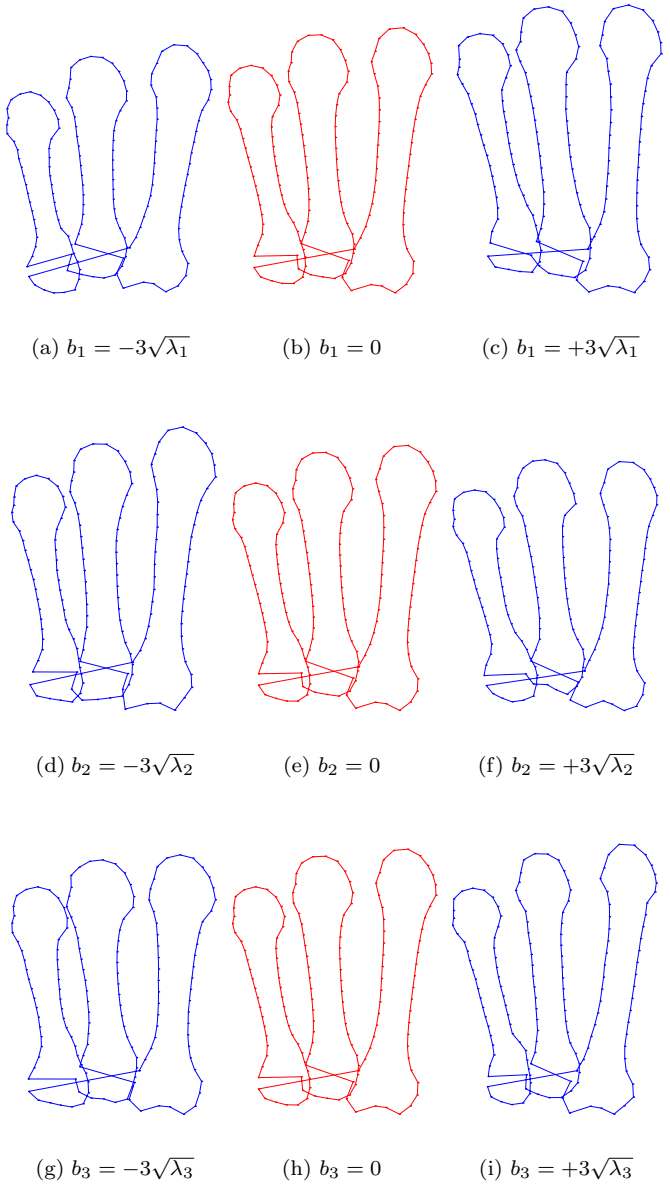


Figure A.4: Independent principal component analysis of each model point.



(a) $b_1 = -3\sqrt{\lambda_1}$ (b) $b_1 = 0$ (c) $b_1 = +3\sqrt{\lambda_1}$

(d) $b_2 = -3\sqrt{\lambda_2}$ (e) $b_2 = 0$ (f) $b_2 = +3\sqrt{\lambda_2}$

(g) $b_3 = -3\sqrt{\lambda_3}$ (h) $b_3 = 0$ (i) $b_3 = +3\sqrt{\lambda_3}$

Figure A.5: Mean shape deformation using 1st, 2nd and 3rd principal mode. $b_i = -3\sqrt{\lambda_i}$, $b_i = 0$, $b_i = 3\sqrt{\lambda_i}$.

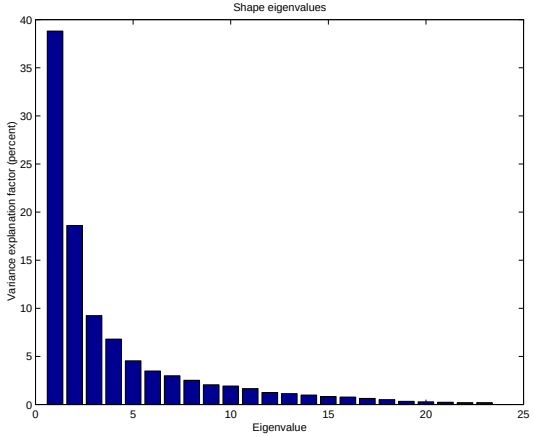


Figure A.6: Shape eigenvalues in descending order.

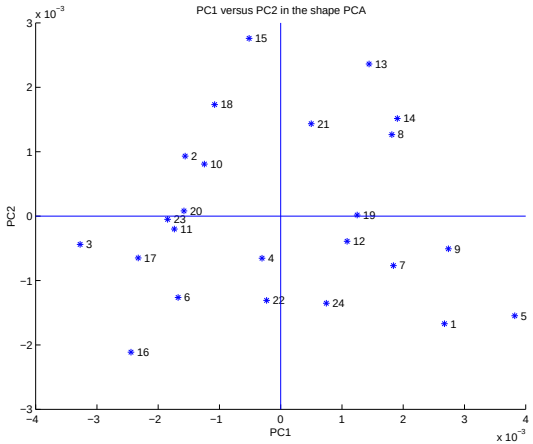


Figure A.7: PC1 ($b_{s,1}$) vs. PC2 ($b_{s,2}$) in the shape PCA.

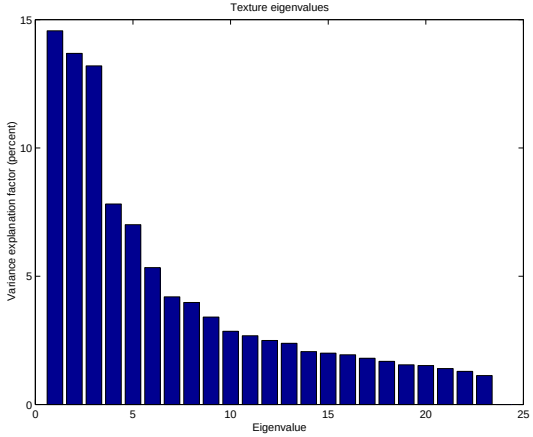


Figure A.8: Texture eigenvalues in descending order.

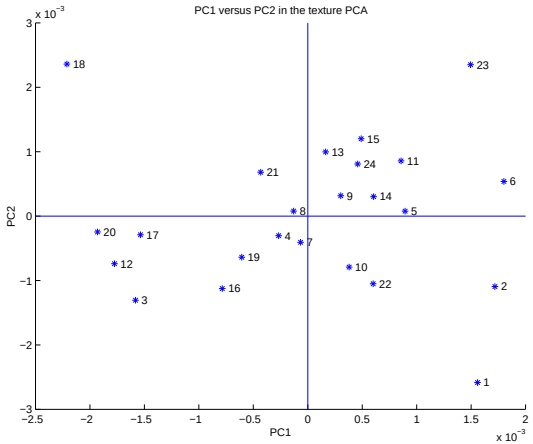


Figure A.9: PC1 ($b_{g,1}$) versus PC2 ($b_{g,2}$) in the texture PCA.

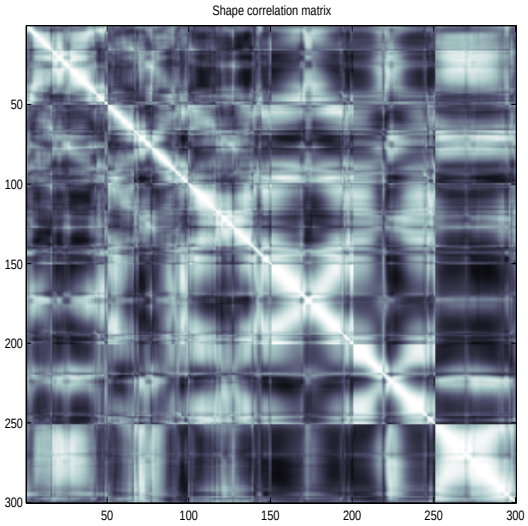


Figure A.10: Correlation matrix of the annotations.

Figure A.11: Texture variance, black corresponds to high variance.

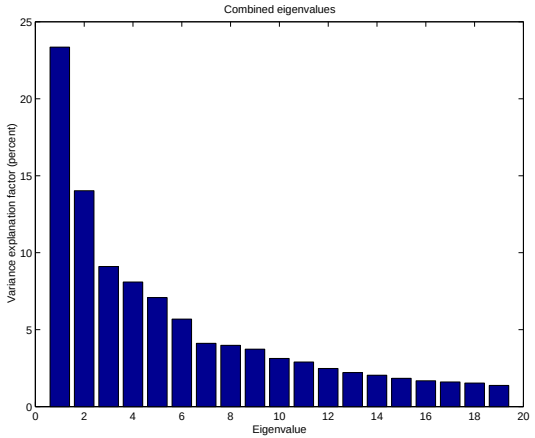


Figure A.12: Combined eigenvalues.

A.2 Cardiac MRIs – Set 1 B-Slices

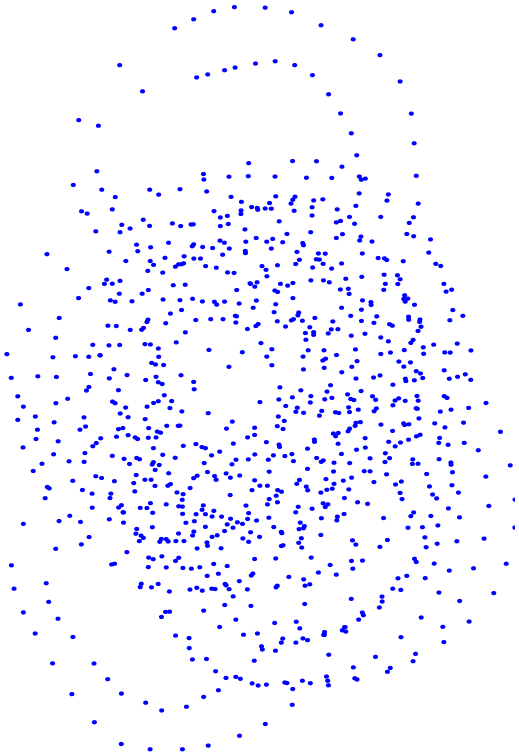


Figure A.13: Point cloud of the unaligned annotations.

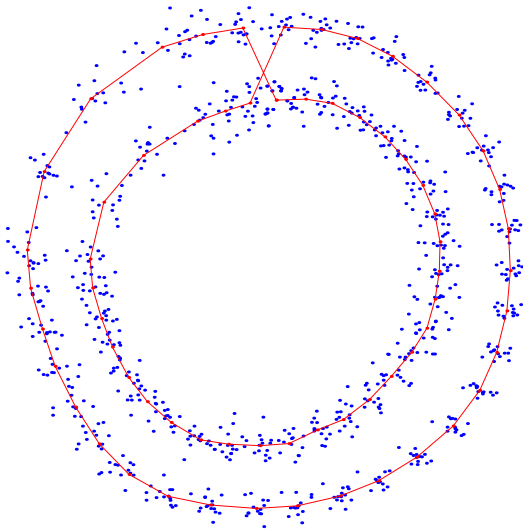


Figure A.14: Point cloud of the aligned annotations with mean shape fully drawn.

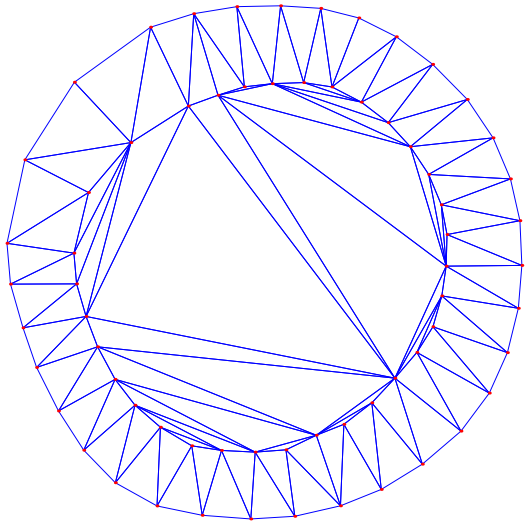


Figure A.15: Delaunay triangulation of the mean shape.

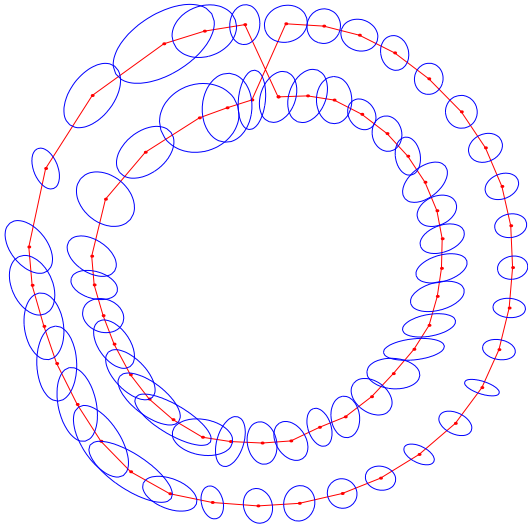
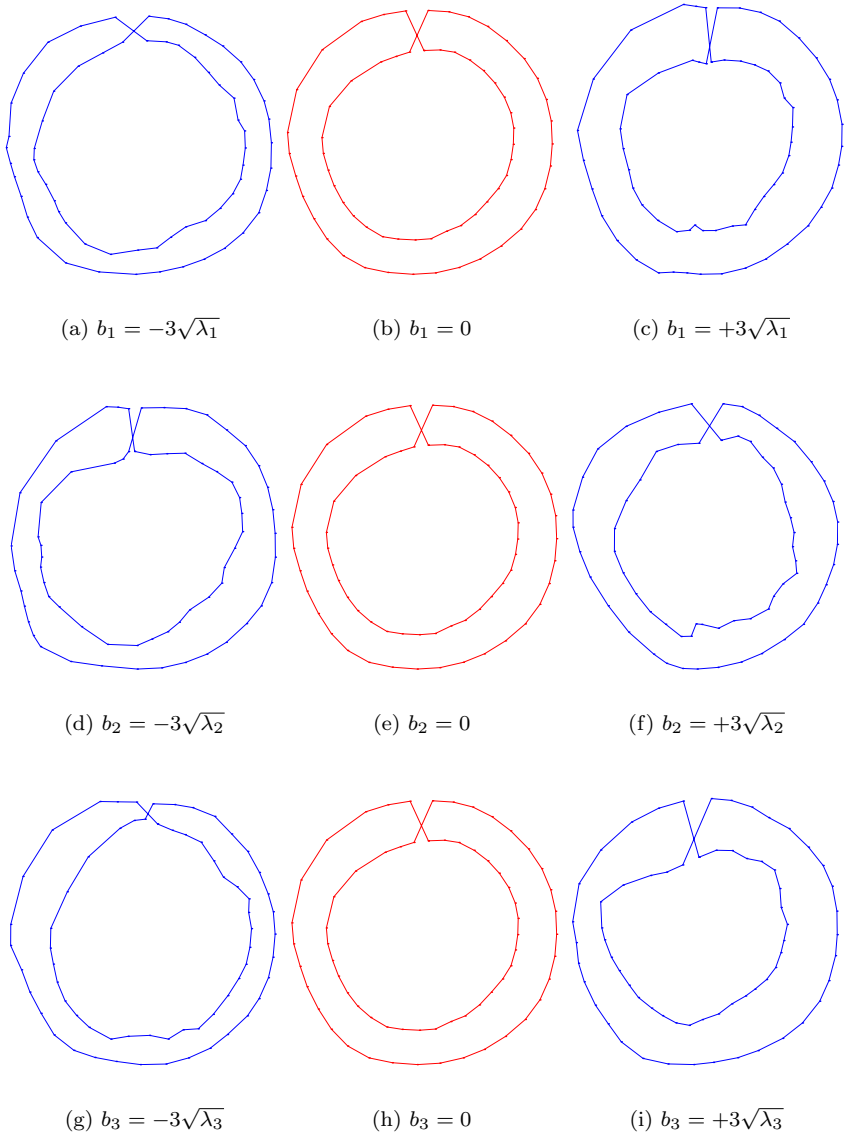


Figure A.16: Independent principal component analysis of each model point.



(a) $b_1 = -3\sqrt{\lambda_1}$ (b) $b_1 = 0$ (c) $b_1 = +3\sqrt{\lambda_1}$

(d) $b_2 = -3\sqrt{\lambda_2}$ (e) $b_2 = 0$ (f) $b_2 = +3\sqrt{\lambda_2}$

(g) $b_3 = -3\sqrt{\lambda_3}$ (h) $b_3 = 0$ (i) $b_3 = +3\sqrt{\lambda_3}$

Figure A.17: Mean shape deformation using 1st, 2nd and 3rd principal mode. $b_i = -3\sqrt{\lambda_i}$, $b_i = 0$, $b_i = 3\sqrt{\lambda_i}$.

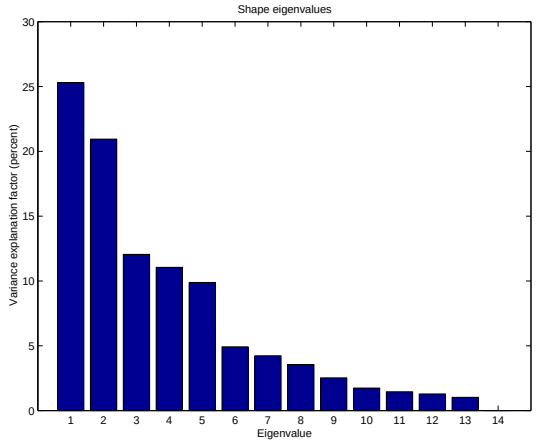


Figure A.18: Shape eigenvalues in descending order.

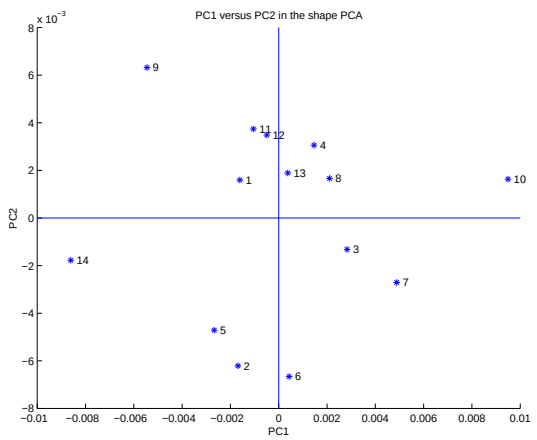


Figure A.19: PC1 ($b_{s,1}$) vs. PC2 ($b_{s,2}$) in the shape PCA.

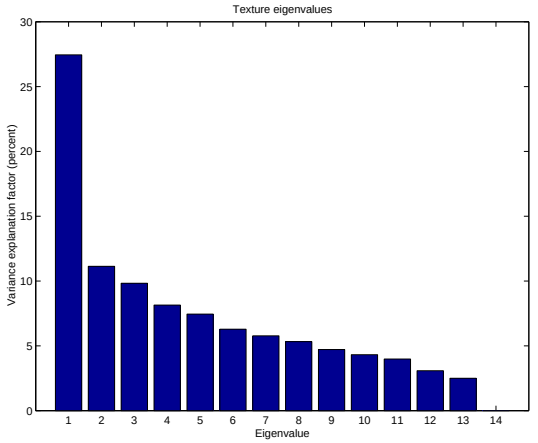


Figure A.20: Texture eigenvalues in descending order.

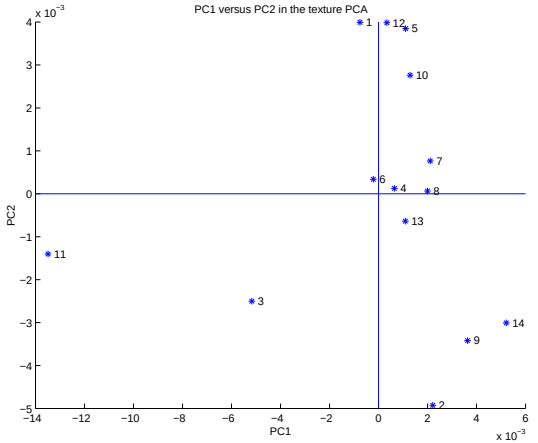


Figure A.21: PC1 ($b_{g,1}$) versus PC2 ($b_{g,2}$) in the texture PCA.

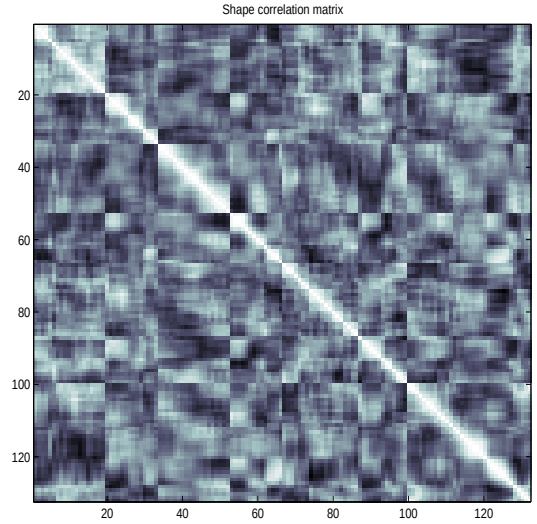


Figure A.22: Correlation matrix of the annotations.

Figure A.23: Texture variance, black corresponds to high variance.

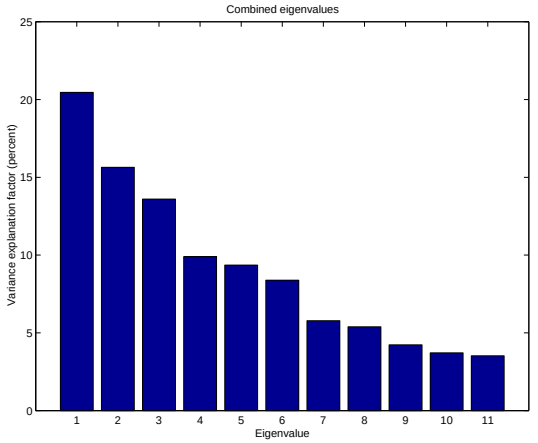


Figure A.24: Combined eigenvalues.

A.3 Cross-sections of Pork Carcasses

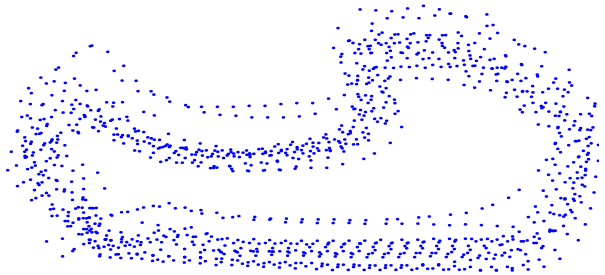


Figure A.25: Point cloud of the unaligned annotations.

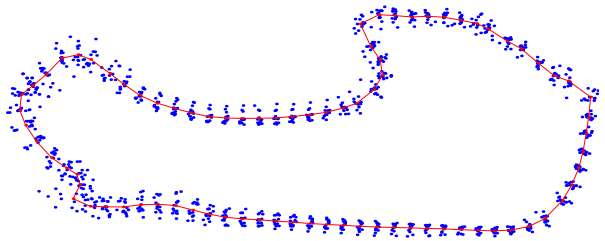


Figure A.26: Point cloud of the aligned annotations with mean shape fully drawn.

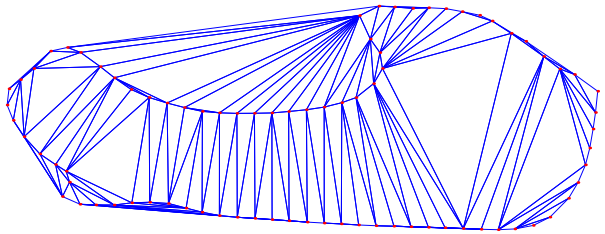


Figure A.27: Delaunay triangulation of the mean shape.

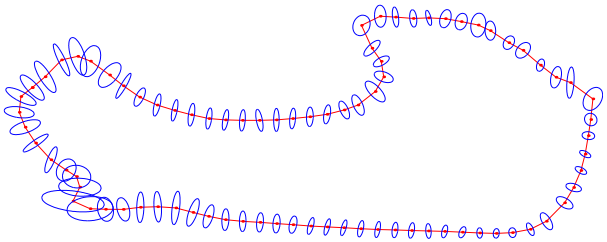
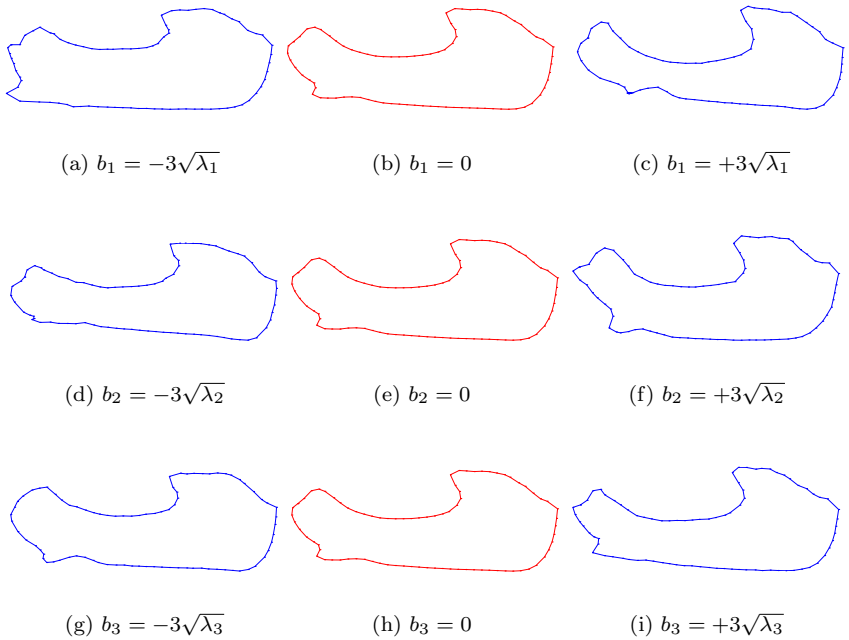


Figure A.28: Independent principal component analysis of each model point.



(a) $b_1 = -3\sqrt{\lambda_1}$ (b) $b_1 = 0$ (c) $b_1 = +3\sqrt{\lambda_1}$

(d) $b_2 = -3\sqrt{\lambda_2}$ (e) $b_2 = 0$ (f) $b_2 = +3\sqrt{\lambda_2}$

(g) $b_3 = -3\sqrt{\lambda_3}$ (h) $b_3 = 0$ (i) $b_3 = +3\sqrt{\lambda_3}$

Figure A.29: Mean shape deformation using 1st, 2nd and 3rd principal mode. $b_i = -3\sqrt{\lambda_i}$, $b_i = 0$, $b_i = 3\sqrt{\lambda_i}$.

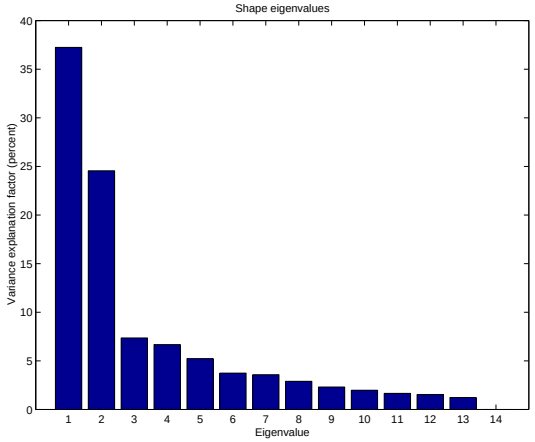


Figure A.30: Shape eigenvalues in descending order.

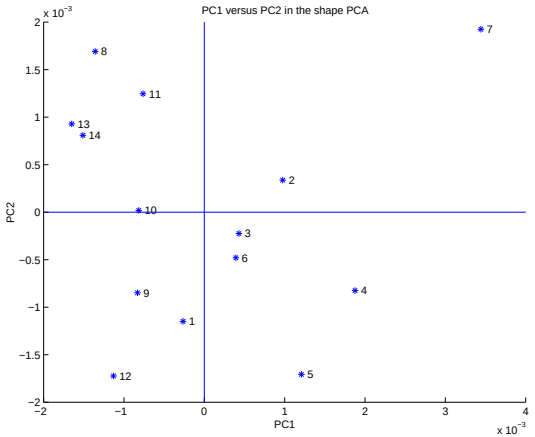


Figure A.31: PC1 ($b_{s,1}$) vs. PC2 ($b_{s,2}$) in the shape PCA.

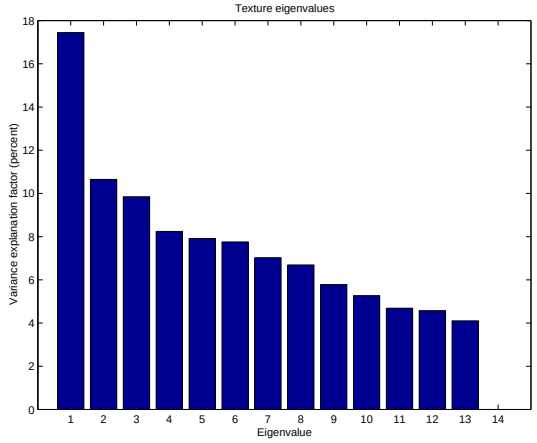


Figure A.32: Texture eigenvalues in descending order.

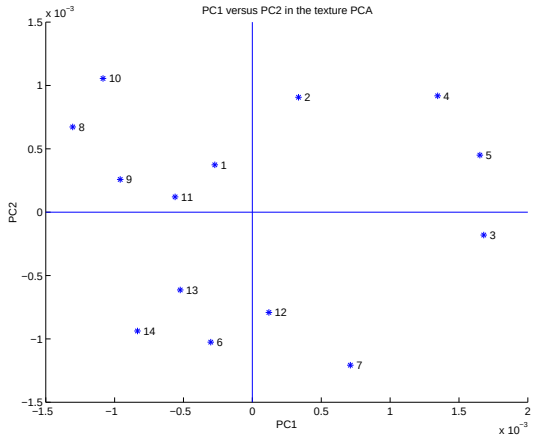


Figure A.33: PC1 ($b_{g,1}$) versus PC2 ($b_{g,2}$) in the texture PCA.

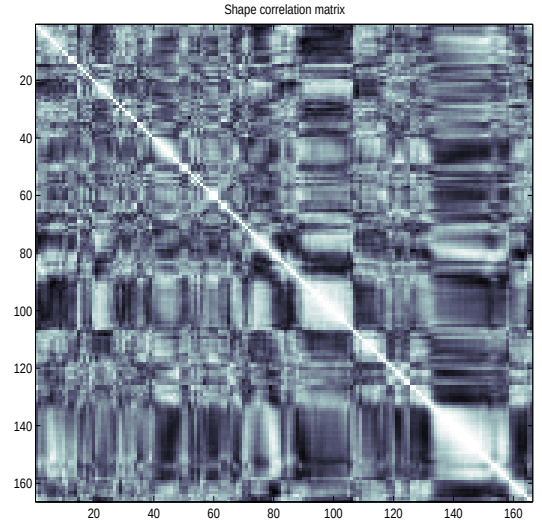


Figure A.34: Correlation matrix of the annotations.

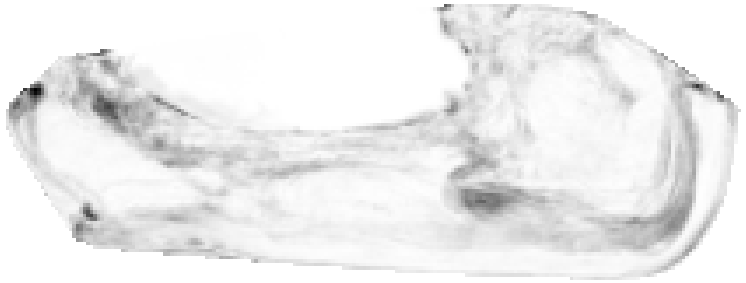


Figure A.35: Texture variance, black corresponds to high variance.

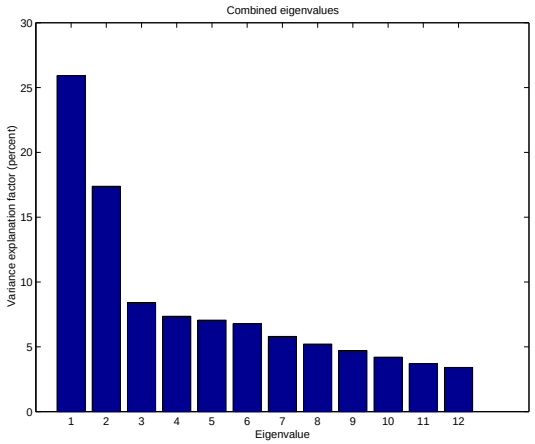


Figure A.36: Combined eigenvalues.

Appendix B

Active Appearance Models: Theory and Cases

During the six months master thesis period, a paper was prepared and submitted to the 9th Danish Conference on Pattern Recognition and Image Analysis (DANKOMB). As documentation of the workload herein, the paper is reprinted below in one-column format.

Since the paper was prepared in the middle of the thesis term, results may not be directly comparable to those reported in this thesis, due to changes in training set, implementation, evaluation methods and such. The nomenclature deviates also slightly from this thesis.

Active Appearance Models: Theory and Cases

M. B. Stegmann[1,1], R. Fisker[1], B. K. Ersbøll[1],
H. H. Thodberg[2], L. Hyldstrup[3]

[1]Department of Mathematical Modelling
Technical University of Denmark
DTU Building 321, DK-2800 Lyngby, Denmark

[2]Pronosco A/S, Kohavevej 5, DK-2950 Vedbæk, Denmark
[3]H:S Hvidovre Hospital, Kettegård Allé 30, DK-2650 Hvidovre, Denmark

Abstract

In this paper, we present a general approach towards image segmentation using the deformable model Active Appearance Model (AAM) as proposed by Cootes et al. A priori knowledge is learned through observation of shape and texture variation in a training set and is used to obtain a compact object class description, which can be used to rapidly search images for new object instances. An overview of the theory behind AAMs is given followed by an improved initialization scheme, thus making the AAMs fully automated. Finally, two cases are presented. It is demonstrated that AAMs can successfully segment bone structures in radiographs of human hands and structures of the human heart in 2D extracts of 4D cardiovascular magnetic resonance images. The observed mean point location accuracy was 1.0 and 1.3 pixels, respectively.

Keywords: Deformable Models, Snakes, Principal Component Analysis, Shape Analysis, Non-Rigid Object Segmentation, Initialization, Metacarpal Radiographs, Cardiovascular Magnetic Resonance Imaging.

[1]Corresponding author: aam@imm.dtu.dk

B.1 Introduction

In recent years, the model-based approach towards image interpretation named deformable models has proven very successful. This is especially true in the case of images containing objects with large variability.

Among the earliest and most well known deformable models is the Active Contour Model – known as *Snakes* proposed by Kass et al. [46]. Snakes represent objects as a set of outline landmarks upon which a correlation structure is forced to constrain local shape changes. In order to improve specificity, many attempts at hand crafting a priori knowledge into a deformable model have been carried out. These include Yuille's et al. [73] parameterization of a human eye using ellipsis and arcs.

In a more general approach, while preserving specificity Cootes et al. [15] proposed the Active Shape Models (ASM) where shape variability is learned through observation. In practice this is accomplished by a training set of annotated examples followed by a Procrustes analysis combined with a principal component analysis.

A direct extension of the ASM approach has lead to the Active Appearance Models [10]. Besides shape information, the textual information, i.e. the pixel intensities across the object, is included into the model. AAMs has been further developed in [14, 22, 13].

Quite similar to AAMs and developed in parallel herewith, Sclaroff & Isidoro suggested the Active Blob approach [58, 43]. Active Blobs is a real-time tracking technique, which captures shape and textual information from a prototype image using a finite element model (FEM) to model shape variation. Compared to AAMs, Active Blobs deform a static texture, whereas AAMs change both texture and shape during the optimization.

For further information on deformable models, refer to the surveys given in [44, 51].

B.2 Active Appearance Models

Below we describe the outline of the Active Appearance Model approach. AAMs distinguish themselves in the sense that segmentation can be carried out using the approach as a black box. We need only provide with

domain knowledge in the form of a training set annotated by specialists (e.g. radiologists etc.).

Described is the training of the model, the modelling of shape and texture variation and the optimization of the model. Finally, an improved method for automated initialization of AAMs is devised.

For a commented pictorial elaboration on the sections below – including the alignment process – refer to appendix A.

B.2.1 Shape & Landmarks

The first matter to clarify is: What do we actually understand by the term *shape*? This paper will adapt the definition by D.G. Kendall [20]:

> **Definition 6**: **Shape** *is all the geometrical information that remains when location, scale and rotational effects are filtered out from an object.*

The next question that naturally arises is: How should one describe a shape? In everyday conversation unknown shapes are often described by referring to well-known shapes – e.g. *"Italy has the shape of a boot"*. Such descriptions can obviously not be utilized in an algorithmic framework.

One way of representing shape is by locating a finite number of points on the outline. Consequently the concept of a *landmark* is adapted [20]:

> **Definition 7**: *A* **landmark** *is a point of correspondence on each object that matches between and within populations.*

A mathematical representation of an n-point shape in k dimensions could be concatenating each dimension in a kn-vector. The vector representation for planar shapes would then be:

$$\mathbf{x} = (x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n)^{\mathrm{T}} \tag{B.1}$$

Notice that the above representation does not contain any explicit information about the point connectivity.

B.2.2 Shape Formulation

A classical statistical method for dealing with redundancy in multivariate data – such as shapes – is the linear orthogonal transformation; *principal component analysis* (PCA).

In our application for describing shape variation by PCA – a shape of n points is considered one data point in a $2n^{\text{th}}$ dimensional space.

In practice the PCA is performed as an eigenanalysis of the covariance matrix of the shapes aligned w.r.t. position, scale and rotation, i.e. the shape analysis is performed on the true shapes according to the definition. As shape metric in the alignment the Procrustes distance [35] is used. Other shape metrics such as the Hausdorff distance [42] could also be considered.

Consequently it is assumed that the set of N shapes constitutes some ellipsoid structure of which the centroid – the mean shape – can be estimated as:

$$\overline{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \tag{B.2}$$

The maximum likelihood estimate of the covariance matrix can thus be given as:

$$\mathbf{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^{\mathrm{T}} \tag{B.3}$$

The principal axis of the $2n^{th}$ dimensional shape ellipsoid are now given as the eigenvectors, $\mathbf{\Phi}_s$, of the covariance matrix.

$$\mathbf{\Sigma\Phi}_s = \mathbf{\Phi}_s\mathbf{\lambda}_s \tag{B.4}$$

A new shape instance can then be generated by deforming the mean shape by a linear combination of eigenvectors, weighted by $\mathbf{b}_s$, also called the modal deformation parameters.

$$\mathbf{x} = \overline{\mathbf{x}} + \mathbf{\Phi}_s\mathbf{b}_s \tag{B.5}$$

Essentially, the point or *nodal representation* of shape has now been transformed into a *modal representation* where modes are ordered according to their *deformation energy* – i.e. the percentage of variation that they explain.

What remains is to determine how many modes to retain. This leads to a trade off between the accuracy and the compactness of the model. However, it is safe to consider small-scale variation as noise. It can be shown that the variance across the axis corresponding to the i^{th} eigenvalue equals the eigenvalue itself, λ_i. Thus to retain p percent of the variation in the training set, t modes can be chosen satisfying:

$$\sum_{i=1}^{t}\lambda_i \geq \frac{p}{100}\sum_{i=1}^{2n}\lambda_i \tag{B.6}$$

B.2.3 Texture Formulation

Contrary to the prevalent understanding of the term *texture* in the computer vision community, this concept will be used somewhat differently below. The main reason for this is that most literature on AAMs uses this definition of texture, probably due to the close resemblance of some of the AAM techniques to techniques in computer graphics.

In computer graphics, the term texture relates directly to the pixels mapped upon virtual 2D and 3D surfaces. Thus, we derive the following definition:

> **Definition 3:** **Texture** *is the pixel intensities across the object in question (if necessary after a suitable normalization).*

A vector is chosen, as the mathematical representation of texture, where m denotes the number of pixel samples over the object surface:

$$\mathbf{g} = (g_1, \ldots, g_m)^{\mathrm{T}} \tag{B.7}$$

In the shape case, the data acquisition is straightforward because the landmarks in the shape vector constitute the data itself. In the texture case one needs a consistent method for collecting the texture information between the landmarks, i.e. an image warping function needs to be established. This can be done in several ways. Here, a piece-wise affine warp based on

the Delaunay triangulation of the mean shape is used. Another, theoretically better, approach might be to use thin-plate splines as proposed by Bookstein [5]. For details on the Delaunay triangulation and image warping refer to [33, 60].

Following the warp sampling of pixels, a photometric normalization of the **g**-vectors of the training set is done to avoid the influence from global linear changes in pixel intensities. Hereafter, the analysis is identical to that of the shapes. Hence a compact PCA representation is derived to deform the texture in a manner similar to what is observed in the training set:

$$\mathbf{g} = \overline{\mathbf{g}} + \mathbf{\Phi}_g \mathbf{b}_g \tag{B.8}$$

Where $\overline{\mathbf{g}}$ is the mean texture; $\mathbf{\Phi}_g$ represents the eigenvectors of the covariance matrix and finally $\mathbf{b}_g$ are the modal texture deformation parameters.

Notice that there will always be far more dimensions in the samples than observations thus leading to rank deficiency in the covariance matrix. Hence, to efficiently compute the eigenvectors of the covariance matrix one must reduce the problem through use of the Eckart-Young theorem. Consult [14, 65] or a textbook in statistics for the details.

Combined Model Formulation

To remove correlation between shape and texture model parameters – and to make the model representation more compact – a 3rd PCA is performed on the shape and texture PCA scores of the training set, **b** to obtain the combined model parameters, **c**:

$$\mathbf{b} = \mathbf{Q}\mathbf{c} \tag{B.9}$$

The PCA scores are easily obtained due to the linear nature of the model:

$$\mathbf{b} = \begin{pmatrix} \mathbf{W}_s \mathbf{b}_s \\ \mathbf{b}_g \end{pmatrix} = \begin{pmatrix} \mathbf{W}_s \mathbf{\Phi}_s^{\mathrm{T}} (\mathbf{x} - \overline{\mathbf{x}}) \\ \mathbf{\Phi}_g^{\mathrm{T}} (\mathbf{g} - \overline{\mathbf{g}}) \end{pmatrix} \tag{B.10}$$

– where a suitable weighting between pixel distances and pixel intensities is obtained through the diagonal matrix $\mathbf{W}_s$. An alternative approach is to

perform the two initial PCAs based on the correlation matrix as opposed to the covariance matrix.

Now – using simple linear algebra – a complete model instance including shape, **x** and texture, **g**, is generated using the **c**-model parameters.

$$\mathbf{x} = \overline{\mathbf{x}} + \mathbf{\Phi}_s \mathbf{W}_s^{-1} \mathbf{Q}_s \mathbf{c} \tag{B.11}$$

$$\mathbf{g} = \overline{\mathbf{g}} + \mathbf{\Phi}_g \mathbf{Q}_g \mathbf{c} \tag{B.12}$$

Regarding the compression of the model parameters, one should notice that the rank of $\mathbf{Q}$ will never exceed the number of examples in the training set.

Observe that another feasible method to obtain the combined model is to concatenate both shape points and texture information into one observation vector from the start and then perform PCA on the correlation matrix of these observations.

B.2.4 Optimization

In AAMs the search is treated as an optimization problem in which the difference between the synthesized object delivered by the AAM and an actual image is to be minimized.

In this way by adjusting the AAM-parameters (**c** and pose) the model can deform to fit the image in the best possible way.

Though we have seen that the parameterization of the object class in question can be compacted markedly by the principal component analysis it is far from an easy task to optimize the system. This is not only computationally cumbersome but also theoretically challenging – optimization theory-wise – since it is not guaranteed that the search-hyperspace is smooth and convex.

However, AAMs circumvent these potential problems in a rather untraditional fashion. The key observation is that each model search constitutes what we call a *prototype search* – the search path and the optimal model parameters are unique in each search where the final model configuration matches this configuration.

These prototype searches can be performed at model building time; thus saving the computationally expensive high-dimensional optimization. Below is described how to collect these prototype searches and how to utilizes them into a run-time efficient model search of an image.

It should be noticed that the Active Blobs approach is optimized using a method quite similar to that of AAMs named *difference decomposition* as introduced by Gleicher [34].

Solving Parameter Optimization Off-line

It is proposed that the spatial pattern in $\delta\mathbf{I}$ can predict the needed adjustments in the model and pose parameters to minimize the difference between the synthesized object delivered by the AAM and an actual image, $\delta\mathbf{I}$:

$$\delta\mathbf{I} = \mathbf{I}_{image} - \mathbf{I}_{model} \tag{B.13}$$

The simplest model we can arrive at constitutes a linear relationship:

$$\delta\mathbf{c} = \mathbf{R}\delta\mathbf{I} \tag{B.14}$$

To determine a suitable $\mathbf{R}$ in equation (B.14), a set of experiments are conducted, the results of which are fed into a multivariate linear regression using principal component regression due to the dimensionality of the texture vectors. Each experiment displaces the parameters in question by a known amount and measuring the difference between the model and the image-part covered by the model.

As evaluation of the assumption of a linear relationship between the model and pose parameters and the observed texture differences, figure B.1 shows the actual and the mean predicted displacement from a number of displacements. The error bars correspond to one standard deviation.

Hence, the optimization is performed as a set of iterations, where the linear model, in each iteration, predicts a set of changes in the pose and model parameters leading to a better model to image fit. Convergence is declared when an error measure is below a suitable threshold.

As error measure, we use the squared L_2 norm of the texture difference, $|\delta\mathbf{g}|^2$. To gain a higher degree of robustness, one might consider using the

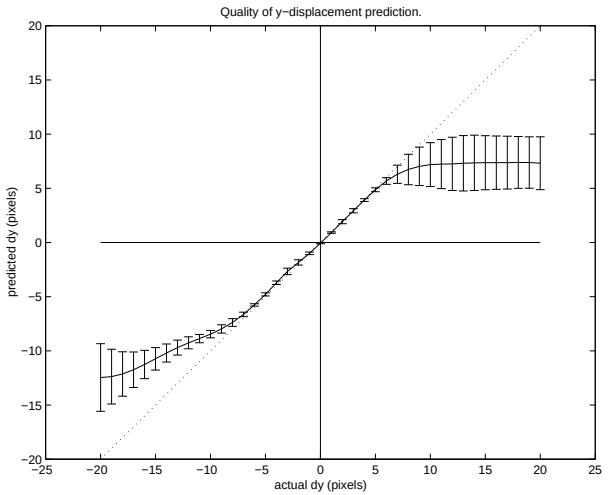


Figure B.1: Displacement plot for a series of y-pose parameter displacements. Actual displacement versus model prediction. Error bars are 1 std.dev.

Mahalanobis distance or a robust norm such as the Lorentzian error norm [58]. Fitness functions allowing for global non-linear transformations such as the mutual information [68, 70] measure might also be considered.

B.2.5 Initialization

The optimization scheme described above is inherently sensitive to a good initialization. To accommodate this, we devise the following search-based scheme thus making the use of AAMs fully automated. The technique is somewhat inspired by the work of Cootes et al. [22].

The fact that the AAMs are self-contained or generative is exploited in the initialization – i.e. they can fully synthesize (near) photo-realistic objects of the class that they represent with regard to shape and textural appearance. Hence, the model, without any additional data, is used to perform the initialization.

The idea is to use the inherent properties of the AAM-optimization – i.e. convergence within some range from the optimum. This is utilized to narrow an exhaustive search from a dense to sparse population of the hyper-

space spanned by pose- and **c**-parameters. In other words, normal AAM-optimizations are performed sparsely over the image using perturbations of the model parameters.

This has proven to be both feasible and robust. A set of relevant search configuration ranges is established and the sampling within this is done as sparsely as possible.

Consider the graph given in figure B.1, which demonstrates that it should be safe to sample the y-parameter with a frequency of at least 10 pixels. One could also claim that as long the prediction preserves the right sign it is only a matter of sufficient iteration.

To achieve sensitivity to pixel outliers we use the variance of the square difference vector between the model and the image as error measure:

$$e_{fit} = V[\delta \mathbf{g}^2] \tag{B.15}$$

As in the optimization this could easily be improved by using more elaborate error measures. In pseudo-code, the initialization scheme for detecting one object per image is:

1. Set $e_{min} = \infty$ and m to a suitable low number (we use $m = 3$)
2. Obtain application specific search ranges within each parameter (e.g. $-\sigma \leq c_1 \leq \sigma$ etc.)
3. Populate the space spanned by the ranges – as sparsely as the linear regression allows – by a set of sampling vectors $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$.
4. For each vector in $\mathbf{V}$
5. Do AAM optimization (max m iterations)
6. Calulate the fit, e_{fit}, as given by (B.15)
7. If $e_{fit} < e_{min}$ Then $e_{min} = e_{fit}$, $\mathbf{v}_{fit} = \mathbf{v}_n$
8. End

The vector $\mathbf{v}_{fit}$ will now hold the initial configuration.

Notice that the application specific search ranges in step 2 is merely a help to increase initialization speed and robustness than it is a requirement. If nothing is known beforehand, step 2 is eliminated and an exhaustive search is performed.

This approach can be accelerated substantially by searching in a multi-resolution (pyramidal) representation of the image.

B.3 Implementation

All experiments, illustrations etc. have been made using the Active Appearance Models Application Programmers Interface (AAM-API) developed in the C++ language and based on the Windows NT platform. This API will be released under the open source initiative in Autumn 2000[2], which means that other freely can download, use and elaborate on the AAM-API.

The foundation for the AAM-API is the Intel Math Kernel Library for fast MMX implementation of BLAS, MS VisionSDK for image handling, ImageMagick for image I/O and finally DIVA for image processing and matrix handling.

AAM-API performance compares to that of Cootes et al. As an example, the MRI optimizations took each 200 ms on average on a PII 350 MHz. Much effort has been put into providing documentation and educational features such as movies of the modes of variation, model search etc.

Further info on AAMs, the AAM-API and full source code documentation can be obtained at the AAM-site.[3]

B.4 Experimental Results

Segmentation in medical images has always posed a difficult problem due to the special image modalities (CT, MRI, PET etc.) and the large biological variability. To assess the performance of AAMs in such environments, our implementation has been tested upon radiographs of human hands and cardiac MRIs.

B.4.1 Radiographs of Metacarpals

Segmentation in radiographs (x-rays) pose a difficult problem due to large shape variability and inherent ambiguity of radiographs. This forms a suitable challenge. Other attempts to perform segmentation in radiographs include the work of Efford [25], where ASMs and other methods were used.

[2] Probably under the GNU Public License.
[3] `http://www.imm.dtu.dk/~aam/`

Twenty radiographs of the human hand were obtained and three metacarpal bones were annotated using 50 points on each. The annotation of metacarpals 2,3 and 4 were concatenated into a 150-point model. To incorporate a more substantial texture contrast into the model, additional 150 points were placed along the normal of each model point, thus arriving at a 300-point shape model. The texture model consisted of approx. 10.000 pixels. Using 16 model parameters, 95% of the variation in the training set were explained.

Automated initialization of the model followed by optimization reached a mean location accuracy of 0.98 pixels (point to associated border [9, 17]) when testing on four unseen images with ground truth annotations. The mean texture error was approx. 7 gray levels (input images were in the byte range).

Examples of initialization and optimization are given in the figures B.2-B.5. Notice a fairly good fit even in the distal (upper) and proximal (lower) end of the metacarpals where radiographs are rather ambiguous.

To asses the performance *within* points, the mean point to point distance is plotted in figure B.6. Not surprisingly, problems arise in the distal and proximal end of the metacarpals due the large shape variability and the ambiguous nature of radiographs in regions of overlap.

Figure B.2: Model border after automated initialization (cropped).

Figure B.3: Optimized model border.

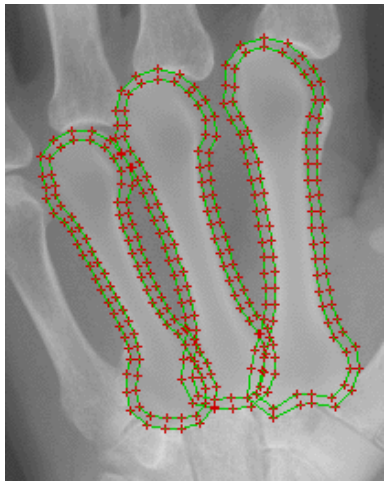


Figure B.4: AAM after automated initialization (cropped).

B.4.2 Cardiac MRIs

Another application in medical imaging is locating structures of 4D (space, time) cardiovascular magnetic resonance images. Temporal registration

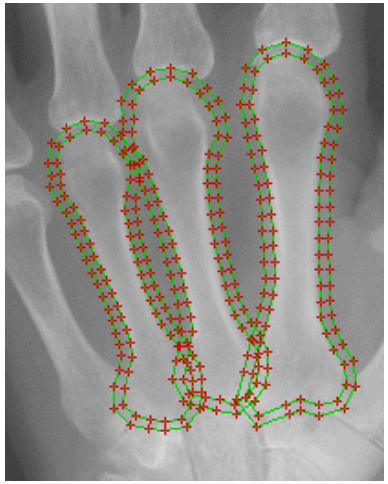


Figure B.5: Optimized AAM (cropped).

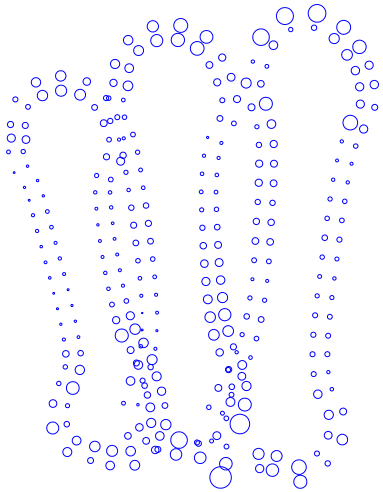


Figure B.6: Mean point to point deviation from the ground truth annotation of each metacarpal. Low location accuracy is observed at the distal and proximal ends.

relative to the heart cycle has been done using ECG-triggered image acquisition. The pixel depth was 8 bits.

An AAM has been built upon only four – spatially and temporally corresponding – 2D slices of four different hearts. The endocardial and epicardial contour have been annotated using 30 points each. The resolution of the 2D slices was 256x256 pixels resulting in a texture model of approx. 2600 pixels. The combined PCA explains 84 % of the variation in the training set using two model parameters.

The cardiac AAM was then used to search in a unseen image spatially and temporally similar to those in the training set. The described initialization technique reached the result seen in figures B.7 and B.9. A final optimization reached a mean point accuracy of 1.7 pixels (point to associated border). The result can be seen in figures B.8 and B.10. The mean texture error was approx. 11 gray levels. By incorporating a model neighborhood similar to the metacarpal model the mean point accuracy was increased to 1.3 pixels.

Future work on the cardiac AAM will include models built upon extracts differing spatially and temporally, thus leading to a somewhat unified AAM of a larger subspace from the original 4D. For a commented pictorial of the cardiac AAM, refer to appendix A.

AAM segmentation of 2D cardiac MRIs has previously been done by Mitchell et al. [52]. A total of 102 images were used for the training set reaching a mean point accuracy of approx. 1 pixel on the endocardial and epicardial contour. Annotated structures were the right ventricle and endocardial and epicardial contours. The model was initialized manually.

B.5 Discussion & Conclusions

In this paper we have presented the basic theory of AAMs and devised a method providing sufficient initialization of AAMs.

The performance of AAMs has been assessed on two different image modalities - i.e. radiographs and magnetic resonance images reaching a mean point location accuracy of 1.0 and 1.3 pixels, respectively. In both cases the location accuracy was noteworthy increased by adding a suitable neighborhood to the outer contours of the model, thus enhancing textual contrast.

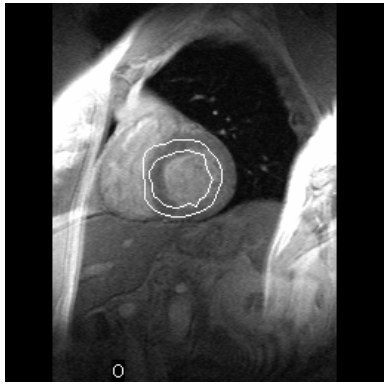


Figure B.7: Model border after automated initialization.

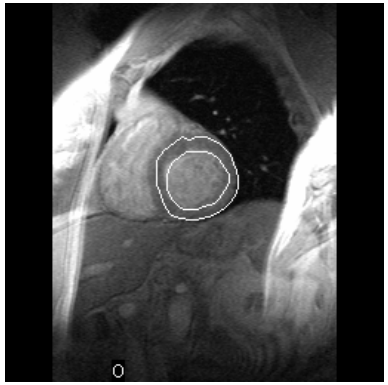


Figure B.8: Optimized model border.

In the MRI case, we have shown that even with a training set as small as four examples, very good segmentation results can be obtained. This leads towards the straightforward assumption that the less variation observed in the object class in question, the smaller the training set one can allow. However – more than four images would probably still be desirable.

The two cases stress the fact that the AAM approach is a example of a general vision technique that capture domain knowledge through observation. Contrary to this the bulk part of model-based vision techniques has hand

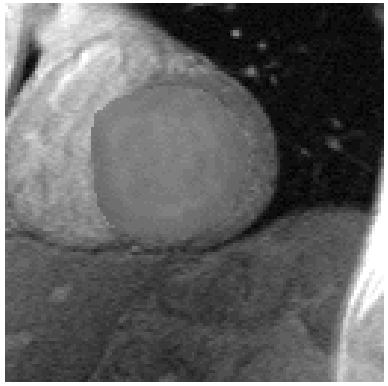


Figure B.9: AAM after automated initialization (cropped).

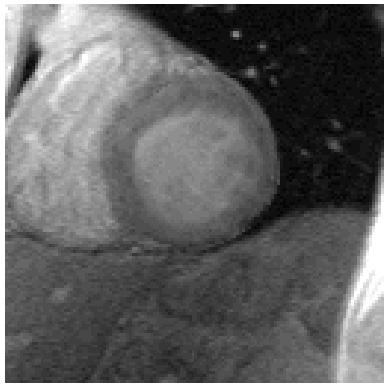


Figure B.10: Optimized AAM (cropped).

crafted a priori knowledge by design and programming. Furthermore, we have experienced the AAM approach to be data-driven (non-parametric), self-contained and fast. We also notice that AAMs extend to multivariate imaging and higher spatial dimensions - i.e. into 3D models etc.

More information on AAMs, papers, presentations, movies of model searches etc. can be obtained at the AAM-site at `http://www.imm.dtu.dk/~aam/`. A more comprehensive treatment of the work of M. B. Stegmann can be found in [65].

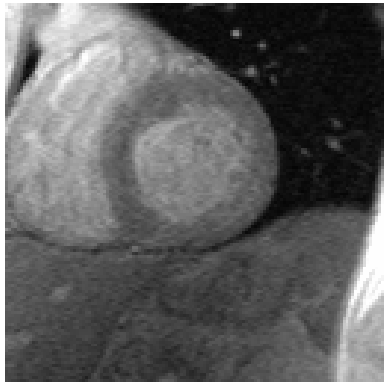


Figure B.11: Original image (cropped).

B.6 Acknowledgements

Cardiac MRIs were provided and annotated by M.D. Jens Christian Nilsson and M.D. Bjørn A. Grønning, H:S Hvidovre Hospital. M.Sc.Eng. Torben Lund provided the annotation tool.

Dr. Rasmus Larsen, M.D. Jens Christian Nilsson and M.D. Bjørn A. Grønning is acknowledged for their very useful comments on the manuscript.

The work of M. B. Stegmann is in part supported by a grant from Pronosco A/S.

B.7 Illustrated Cardiac AAM

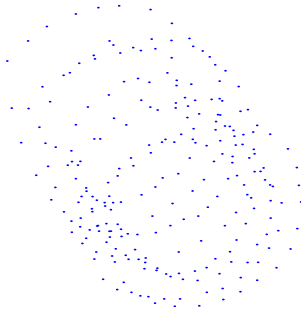


Figure B.12: Point cloud of four unaligned heart chamber annotations.

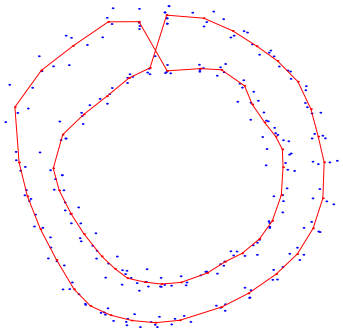


Figure B.13: Point cloud of four aligned heart chamber annotations with mean shape fully drawn.

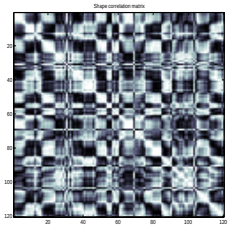


Figure B.14: Correlation matrix of the four annotations. Observe the obvious point correlations.

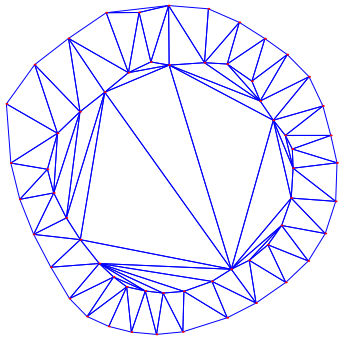


Figure B.15: Delanay triangulation of the mean shape.

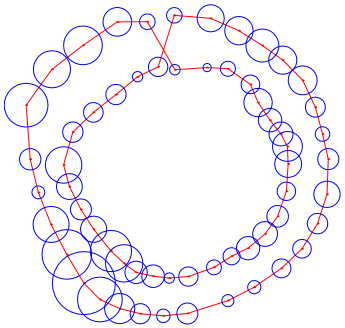


Figure B.16: Point variation of the four annotations; radius = $\sigma_x + \sigma_y$. Notice the large point variation to the lower left.

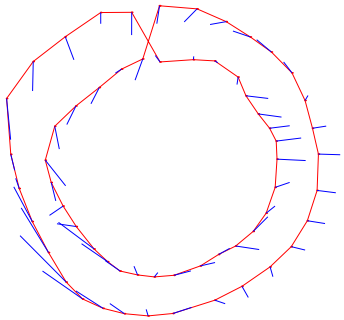


Figure B.17: The first eigenvector plotted as displacement vectors. Notice that the large point variation observed in figure B.16 is point variation along the contour, which only contributes to a less compact model contrary to explaining actual shape variation.

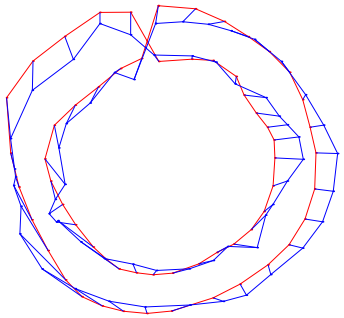


Figure B.18: Mean shape and shape deformed by the first eigenvector. Notice that this emphasizes the point above; that a lot of the deformation energy does not contribute to any actual shape changes.

Appendix C

The AAM Web-site

As supplementary documentation on the thesis work, a web-site has been produced. The primary motivation was to:

- Provide a high quality, interactive way to view the produced results since major parts of the results is very well (educational-wise) presented as movies.
- Provide long-term easy public access to this thesis work in a manner similar to other university projects this thesis work has benefited from.

Among the featured contents are:

Model search movies Follow the optimization using shape and textural deformation through a movie sequence.

Modes of variation Deform an AAM in real-time using the principal components of shape, texture and combined variation.

Source code documentation Cross-referenced and downloadable in HTML, PS & PDF format.

Additional documentation The thesis itself, papers, slide presentations etc.

The web-site can be reached at:

`http://www.imm.dtu.dk/~aam/`

Appendix D

Source Code Documentation

All source code developed during this master thesis work has been annotated with special tags for the purpose of performing automatically extraction of relevant source code documentation. In this way, it is simple and fast to obtain an updated documentation in perfect synchronization with the actual code.

The tool used – Doc++[1] – provided methods for generating cross-referenced HTML and LaTeX/PDF output. Both of these can be viewed online or printed[2] from the AAM-site:

`http://www.imm.dtu.dk/~aam/`

Full Class Listening

Below is listed all classes of the AAM-API. On top of that comes the classes developed for the console interface, AAMC, which is excluded from the documentation.

[1] Doc++ resides at `http://www.linuxsupportline.com/~doc++/`
[2] In the PostScript and PDF format.

CAAMCore
Core Active Appearance Model functionality

CAAMDelaunay
Delaunay Triangulator.

CAAMFEMDeform
Shape deformation using a finite element model.

CAAMInitEntry
Container for initialization results.

CAAMInitCandidates
Initialization candidate container.

CAAMInitialize
Abstract base class for all AAM initialization classes.

CAAMInitializeStegmann
Implementation of the Stegmann intialization method.

CAAMLinearReg
Performs multi-variate linear regression on a set of experiments.

CAAMPoint
Point container.

CAAMTriangle
Triangle container with built-in hit test.

CAAMMesh
2D triangular mesh container.

CAAMMovie
Handling and manipulation of AVI movie files.

CAAMOptRes
Calculates and stores optimization results.

CAAMOptimize

General purpose optimization of the AAM.

CAAMShapePointInfo
Auxiliary point data.

CAAMShape
Shape container.

CAAMShapeCollection
Shape collection container and shape-aligner.

CAAMUtil
Utility methods for the AAM project.

CAAMPropsReader
Simple lo-fi property reader.

CAAMWarp
Base class for 2D warp classes.

CAAMWarpLinearCacheEntry
Cache entry class used by CAAMWarpLinear.

CAAMWarpLinear
Piece-wise affine warping between two shapes.

CAAMWarpLFEM
Piece-wise linar warp class using FEM deformation.

Appendix E

AAM-API File Format Examples

E.1 AMF – AAM Model File

```
########################################################################

Active Appearance Model File

Written            : Friday July 21 - 2000 [15:58]

Format version     : 0.92

Images             : 13

Shape points       : 166

Texture samples    : 14959

Model reduction    : 3

Add Shape Extents  : 3

Convex hull used   : Yes

Additional doc.    : No

Tangent space used : Yes
```



```
Learning Method    : 0

Variance Exp. Level: 95

Parameters used    : 11

FEM used           : No

Image names      :
    .\F1011flb.HIPS
    .\F1019flb.HIPS
    .\F1031flb.HIPS
    .\F1051flb.HIPS
    .\F1053flb.HIPS
    .\F1059flb.HIPS
    .\F1064flb.HIPS
    .\F1079flb.HIPS
    .\F1083flb.HIPS
    .\F1096flb.HIPS
    .\F1101flb.HIPS
    .\F1102flb.HIPS
    .\F1103flb.HIPS

Combined mode variation :
    1    26.89%     ( 26.89%)
    2    16.99%     ( 43.88%)
    3     8.40%     ( 52.29%)
    4     7.70%     ( 59.99%)
    5     7.34%     ( 67.32%)
    6     6.63%     ( 73.96%)
    7     5.76%     ( 79.71%)
    8     5.36%     ( 85.07%)
    9     4.28%     ( 89.35%)
    10    3.86%     ( 93.21%)
    11    3.51%     ( 96.72%)

Shape mode variation :
    1    38.77%     ( 38.77%)
    2    23.62%     ( 62.38%)
    3     7.74%     ( 70.13%)
    4     6.49%     ( 76.62%)
    5     5.66%     ( 82.28%)
    6     3.95%     ( 86.23%)
    7     3.86%     ( 90.09%)
    8     2.80%     ( 92.88%)
    9     2.19%     ( 95.08%)
    10    1.88%     ( 96.96%)
    11    1.73%     ( 98.69%)
```

```
    12    1.31%     (100.00%)

Texture mode variation :
    1    18.61%     ( 18.61%)
    2    12.33%     ( 30.93%)
    3    10.66%     ( 41.59%)
    4     8.73%     ( 50.32%)
    5     8.56%     ( 58.88%)
    6     7.54%     ( 66.42%)
    7     7.35%     ( 73.77%)
    8     6.24%     ( 80.01%)
    9     5.69%     ( 85.70%)
    10    4.99%     ( 90.69%)
    11    4.85%     ( 95.54%)
    12    4.46%     (100.00%)
    13    0.00%     (100.00%)

#########################################################################
```

E.2 ACF – AAM Config File

```
#########################################################################
#
#   Active Appearance Model Builder Configuration File
#
#   Best viewed with tabsize==4
#
#########################################################################


3           # Model reduction               [1-n]   (reduction factor = 1/x)

3           # Model expansion               [0-n]   (pixels along the
                                                     model point normal)

1           # Use convex hull               [0|1]   (off/on)

0           # Verbose mode                  [0|1]   (off/on)

1           # Write registration movie      [0|1]   (off/on)

1           # Write variance image          [0|1]   (off/on)

1           # Produce model documentation   [0|1]   (off/on)

1           # Use tangent space projection  [0|1]   (off/on)
```



```
0           # Learning method (0=single displace,1=multiple displace)

95          # Variance explanation level    [100-1] (percent)
```

E.3 ASF – AAM Shape File

```
#########################################################################
#
#    AAM Shape File  -  written: 14-Jul-2000 19:44:17
#
#########################################################################

#
# number of model points
#
66

#
# model points
#
# format: <path#> <type> <x rel.> <y rel.> <point#> <connects from> <connects to>
#
0   1   0.518718    0.370570    0   32  1
0   1   0.532763    0.369818    1   0   2
0   1   0.550159    0.376979    2   1   3
0   1   0.550243    0.385660    3   2   4
0   1   0.543755    0.401262    4   3   5
0   1   0.556200    0.415997    5   4   6
0   1   0.575146    0.422829    6   5   7
0   1   0.581374    0.415047    7   6   8
0   1   0.592347    0.419907    8   7   9
0   1   0.599675    0.436490    9   8   10
0   1   0.605776    0.449764    10  9   11
0   1   0.597277    0.465434    11  10  12
0   1   0.589506    0.480103    12  11  13
0   1   0.574186    0.487015    13  12  14
0   1   0.572306    0.493672    14  13  15
0   1   0.565832    0.499288    15  14  16
0   1   0.552905    0.508478    16  15  17
0   1   0.540465    0.501954    17  16  18
0   1   0.524880    0.501810    18  17  19
0   1   0.521895    0.514605    19  18  20
0   1   0.537654    0.518689    20  19  21
0   1   0.516845    0.521957    21  20  22
0   1   0.495800    0.516490    22  21  23
0   1   0.479226    0.509700    23  22  24
0   1   0.464311    0.496042    24  23  25
```

```
0   1   0.449038    0.482457    25  24  26
0   1   0.442961    0.465572    26  25  27
0   1   0.441553    0.444790    27  26  28
0   1   0.441644    0.425711    28  27  29
0   1   0.446147    0.410962    29  28  30
0   1   0.457776    0.391605    30  29  31
0   1   0.477158    0.377265    31  30  32
0   1   0.498719    0.375223    32  31  0
1   0   0.527584    0.347407    33  65  34
1   0   0.540823    0.350820    34  33  35
1   0   0.554225    0.355885    35  34  36
1   0   0.571020    0.365448    36  35  37
1   0   0.584007    0.374006    37  36  38
1   0   0.597431    0.383626    38  37  39
1   0   0.609678    0.395252    39  38  40
1   0   0.621089    0.409628    40  39  41
1   0   0.626961    0.422494    41  40  42
1   0   0.628915    0.437005    42  41  43
1   0   0.628104    0.450329    43  42  44
1   0   0.623325    0.466479    44  43  45
1   0   0.615272    0.479872    45  44  46
1   0   0.604699    0.492765    46  45  47
1   0   0.589989    0.507293    47  46  48
1   0   0.576942    0.517555    48  47  49
1   0   0.563292    0.526087    49  48  50
1   0   0.544622    0.534137    50  49  51
1   0   0.528754    0.537687    51  50  52
1   0   0.521873    0.539179    52  51  53
1   0   0.519447    0.538970    53  52  54
1   0   0.513286    0.538763    54  53  55
1   0   0.489473    0.534848    55  54  56
1   0   0.469023    0.528177    56  55  57
1   0   0.450248    0.513003    57  56  58
1   0   0.433815    0.493809    58  57  59
1   0   0.424770    0.473959    59  58  60
1   0   0.422778    0.446886    60  59  61
1   0   0.422572    0.419639    61  60  62
1   0   0.428038    0.399094    62  61  63
1   0   0.449634    0.384622    63  62  64
1   0   0.469204    0.359666    64  63  65
1   0   0.496774    0.347135    65  64  33
```

E.4 AOF – AAM Optimization File

```
######################################################################
#
```



```
#   Active Appearance Model Optimization File
#
#   Best viewed with tabsize==4
#
######################################################################


##########################
# pose search parameters
##########################

0 0 0 0     # search region; format "x1 x2 y1 y2"
            # to use the whole image supply "0 0 0 0"

10 5        # size search; format "deviation steps"
            # hence "10 5" would search the image using
            # the following sizes 90% 95%, 100%, 105%, 110%

.1 3        # rotation search; format "deviation steps"
            # hence "10 5" would search the image using
            # the following rotations of the mean shape
            # -0.1pi, 0, 0.1pi


##########################
# model search parameters
##########################

3 7         # pc1-parameter search; format "deviation steps"
            # hence "3 7" will derform the model using
            # pc1 = -3, -2, -1, 0, 1, 2, 3 (std.dev.)
```

Appendix F

AAM-API Console Interface Usage

```
AAMC - Active Appearance Model Console Interface - version 0.8.32
Copyright (c) Mikkel B. Stegmann 2000 - aam@imm.dtu.dk. All rights reserved.
For further information see http://www.imm.dtu.dk/~aam/

USAGE:
       aamc <b|c|e|m|r|s|t|w|sm|d> [additional mode arguments]

MODES:
       aamc b <input dir> <extension> <out model> [acf file]
       aamc c <input image> <output image> [reduction factor]
       aamc e <model> <dir> <ext> [still|movie|both|none*] [pseudo|auto*]
              [sm] [ft]
       aamc m <model.amf> <type: all|shape|texture|combined> [#modes] [#frames]
       aamc r <model.amf> <input dir> <extension>
       aamc s <input model.amf> <input image> [movie filename]
       aamc t <model.amf> <input movie>
       aamc w <input image> <reduction factor>
       aamc sm <input movie> [output extension]
       aamc d

For futher help write: 'aam -help', 'aam -full' or 'aam <modename> -help'

The tool 'aamc' is the console interface to the AAM-API.

-------------------------------------------------------------------------
MODE: b - Builds an Active Appearance Model.
```



```
-------------------------------------------------------------------------

USAGE:
      aamc b <input dir> <extension> <out model> [acf file]

DESCRIPTION:

Builds an Active Appearance Model.

In this mode the principal component analysis, parameter optimization training
etc. are done.

input dir : Directory containing images and annotations.
extension : Image extension used. Ex. 'bmp', 'hips' etc.
out model : Filename (and path) of the output model file. Ex. 'model42'.
acf file  : AAM configuration file.


-------------------------------------------------------------------------
MODE: c - Image conversion with optional reduction.
-------------------------------------------------------------------------

USAGE:
      aamc c <input image> <output image> [reduction factor]

DESCRIPTION:

Image conversion with optional reduction.

Example usage:

     aamc c meta.tif meta.bmp


-------------------------------------------------------------------------
MODE: e - Evaluates an Active Appearance Model.
-------------------------------------------------------------------------

USAGE:
      aamc e <model> <dir> <ext> [still|movie|both|none*] [pseudo|auto*]
             [sm] [ft]

DESCRIPTION:

Evaluates an Active Appearance Model.

model              : The model .amf that should be evaluated.
dir                : Directory containing images and ground truth annotations.
ext                : Image extension used. Ex. 'bmp', 'hips' etc.
```

```
[still|movie|both] : Write stills of the initial and optimized model
                     and/or movies of the complete optimization.
[pseudo|auto*]     : Initialization method.
[sm]               : Similarity measure used in the optimization:
                        0 = Non-normalized L_2 norm*
                        1 = The Mahalanobis distance
                        2 = The Lorentzian error norm
[ft]               : Fine tuning of the optimization:
                        0 = None*
                        1 = Steepest Descent
                        2 = Conjugate Gradient
                        3 = Quasi-Newton, BFGS
                        4 = Pattern search
                        5 = Simulated annealing

Output is written in the input dir in the file 'results.txt'
Default settings are marked with an asterisk (*)


----------------------------------------------------------------------
MODE: m - Writes Active Appearance Model movies.
----------------------------------------------------------------------

USAGE:
      aamc m <model.amf> <type: all|shape|texture|combined> [#modes] [#frames]

DESCRIPTION:

Writes Active Appearance Model movies.

This mode documents the given AAM by generating movies showing
the shape, texture and combined variation resulting from the PCA.

Output is written in current dir.


----------------------------------------------------------------------
MODE: r - Tests the regression prediction in an AAM.
----------------------------------------------------------------------

USAGE:
      aamc r <model.amf> <input dir> <extension>

DESCRIPTION:

Tests the regression prediction in an AAM.

Output is written in the input dir in matlab format.
```



```
----------------------------------------------------------------------
MODE: s - Active Appearance Model search.
----------------------------------------------------------------------

USAGE:
      aamc s <input model.amf> <input image> [movie filename]

DESCRIPTION:

Active Appearance Model search.

Search output is written in the input dir.
Automatic initialization is performed.


----------------------------------------------------------------------
MODE: t - Performs tracking in a movie file (.avi).
----------------------------------------------------------------------

USAGE:
      aamc t <model.amf> <input movie>

DESCRIPTION:

Performs tracking in a movie file (.avi).

This mode performs a through initialization of the AAM in the first frame
and uses the convergence position as initial pose in the next frame
(and so on...).

Example usage:

     aamc t hand.avi


----------------------------------------------------------------------
MODE: w - Plots an annotation into a given image.
----------------------------------------------------------------------

USAGE:
      aamc w <input image> <reduction factor>

DESCRIPTION:

Plots an annotation into a given image.

Image input format      : .bmp .hips
Annotation input format : .m
```

```
An annotation file is expected to be placed in the
same directory as the image.

E.g.: 'c:\scan42.bmp' and 'c:\scan42.m'


----------------------------------------------------------------------
MODE: sm - Split movie file (.avi) into frames.
----------------------------------------------------------------------

USAGE:
      aamc sm <input movie> [output extension]

DESCRIPTION:

Split movie file (.avi) into frames.

Example usage:

     aamc sm shape01.avi bmp


----------------------------------------------------------------------
MODE: d - Debug/Test console mode. Don't use!
----------------------------------------------------------------------

USAGE:
      aamc d

DESCRIPTION:

Debug/Test console mode. Don't use!
```

Appendix G

ASF – AAM Shape Format Specification

Since the extended shape representation constitutes one of the contributions a thorough description of it is features is given below.

An ASF file is structured as a set of lines separated by a CR character. Anywhere in the file, comments can be added by starting a line with the percentage character. Line 1 contains the total number of points, n, in the shape. Line 1 to n+1 contains the point information (one line per point) such as the point location, type, connectivity. The format philosophy is that quick and simple access is preferred over data compactness (thus: data redundancy is allowed).

The precise format of a point definition is:

> point := ¡path#¿ ¡type¿ ¡rel. point x¿ ¡rel. point y¿ ¡point#¿ ¡connects from¿ ¡connects to¿

Each field is seperated by space(s) or tab(s).

¡path#¿ The path that the point belongs to. Point from different paths must not be interchanged (in the line order).

¡type¿ A bitmapped field that defines the type of point:

- Bit 1: Outer edge point/Inside point
- Bit 2: Original annotated point/Artificial point
- Bit 3: Closed path point/Open path point
- Bit 4: Non-hole/Hole point

Remaining bits should be set to zero. An inside artificial point which is a part of an closed hole, has thus the type: $(1 << 1) + (1 << 2) + (1 << 4) = 1 + 2 + 4 = 7$.

¡rel. point x¿ The relative x-position of the point – i.e. pixel x = 47 in a 256 pixel wide image has the position 47/256 = 0.18359375.

¡rel. point y¿ The relative y-position of the point – i.e. pixel y = 47 in a 256 pixel tall image has the position 47/256 = 0.18359375.

¡point#¿ The point number. First point is zero. This is merely a service to the human reader since the real point number is implicitly given by the line at where the point occurs.

¡connects from¿ The previous point on this path. If none ¡connects from¿ == ¡point#¿ can be used.

¡connects to¿ The next point on this path. If none ¡connects from¿ == ¡point#¿ can be used.

Path points are assumed to be defined clockwise. That is; the outside normal is defined to be on left of the point in the clockwise direction. Hole is thus defined counter-clockwise.

Points are defined in the fourth quadrant – i.e. row, column notation (0,0) = upper left corner.

Isolated points are signaled using ¡connects from¿ == ¡connects to¿ == ¡point#¿.

A shape must have at least one outer edge. If the outer edge is open, the convex hull is used to determine the interior of the shape.

Refer to appendix E for an example ASF file.