Wedgelet Enhanced Appearance Model

Master Thesis

by

Sune Darkner

IMM, DTU Lyngby DK-2800

IMM-THESIS-2004-11

March 1, 2004

Preface

This M.Sc. thesis is the final requirement for obtaining the degree: Master of Science in Engineering . This work has been carried out over a period from the 15th. of February 2003 to the 15th. of February 2004 at the Image Group at Informatics and Mathematical Modeling at the Technical University of Denmark DTU. The work has been supervised by Associate Professor Ph.D. Rasmus Larsen and Co. supervised by Associate Professor Ph.D. Bjarne K. Ersbøll.

I would like to thank Rasmus Larsen for many inspiring discussions about the topics of this thesis. I would also like to thank Mikkel B. Stegmann for assisting by supplying an API for testing the results of this thesis. Further more Karl Skoglund and Alex Dubinsky for some nice theoretical discussions and for proofreading along with Mark Wrobel. I would like to thank all of the Image and Computer Graphics Group for creating an inspiring atmosphere to work in. Furthermore, DTU students M. M. Nordstrøm, M. Larsen, and J. Sierakowski are thanked for collecting and annotating the face database used in this thesis. Finally i would like to thank my wife to be Frederikke for putting up with me especially the last two months while writing this thesis.

Kgs. Lyngby, March 1, 2004

Sune Darkner s974594

IMM-THESIS-2004-11

Abstract

Statistical region-based segmentation methods such as the Active Appearance Model (AAM) are used for establishing dense correspondences in images based on learning the variation in shape and pixel intensities in a training set. For low resolution 2D images this can be done reliably at close to real-time speeds. However, as resolution increases this becomes infeasible due to excessive storage and computational requirements. In this thesis it is proposed to reduce the textural components by modeling the coefficients of a wedgelet based regression tree instead of the original pixel intensities. The wedgelet regression trees employed are based on the triangular domains and estimated using cross validation. The wedgelet regression trees serves to 1) reduce noise and 2) produce a compact textural description. The wedgelet enhanced appearance model is applied to a case study of human faces. Compression rates of the texture information of 1:40 is obtained without sacrificing segmentation accuracy noticeably, even at compression rates of 1:115 fair segmentation is achieved. For regularization of geometric property of the wedgelet decomposition a Markov Random Field method is introduced which improves the performance of the segmentation on the wedgelet enhanced appearance model.

***Keywords**: Wedgelets, appearance model, CART, markov random field, compression, cross validation, trees, graph mathching, wavelets*

Resumé

Statistiske metoder til segmentering så som Active Appearance modeller, bliver brugt til at skabe en tæt sammenhæng imellem billeder ved at undersøge variationen i pixel intensiteter og form i et trænings sæt. Dette kan gres i realtid for billeder med lav opløsning i 2D. Men med sigende opløsning bliver dette uoverkommeligt pga. stort forbrug af hukommelse og lange beregnings tider. Denne afhandling foreslår en metode til at reducere tekstur komponenterne af modellen ved at modellere teksturen i et wedgelet baseret regressions træ i stedet for de originale pixel intensiteter. Det wedgelet baserede regressions træ er baseret p trekanter og bliver estimeret ved hjælp af krydsvalidering. Regressions træet tjener to formål: 1. At reducere støj og 2. at skabe en kompakt beskrivelse af teksturen. Den wedgelet forbedrede appearance model bliver benyttet p et studie bestende af ansigter. Kompressions rater optil 1:40 bliver opnet uden at kompromittere segmenterings nøjagtigheden i modellen. Kompressions rater helt op til 1:115 opnås med en rimelig segmentering. For at regularisere de geometriske egenskaber ved wedgelet dekompositionen introduceres en metode baseret p markov random fields, som forbedre segmenteringen opnået med den wedgelet forbedrede appearance model.

***Nøgleord**: Wedgelets, appearance model, CART, markov random field, kompression, kryds validering, træer, graf mathching, wavelets*

Contents

Introduction

The Active Appearance Model (AAM) framework [1] has since its introduction been applied successfully to segmentation of many types of deformable objects in images (e.g. faces, cardiac ventricles, brain structures [1, 2, 3, 4]). It is based on the estimation of linear models of shape and texture variation by the use of principal components analysis of landmarks coordinates and pixel intensities and subsequent inference of model parameters from unseen images by a tangent plane approximation of the image manifold.

Modeling every pixel intensity is manageable for low-resolution 2D images. But moving to high-resolution 2D and 3D images, even 3D time-series, this approach is rendered at best very slow and at worst infeasible due to excessive storage and computational requirements.

In order to overcome this problem various alternatives to modeling the raw pixel intensities have been considered. Cootes et al. [5] used a sub-sampling scheme to reduce the texture model by a ratio of 1:4. The scheme selected a subset of the pixel intensities based on the ability of each pixel to predict corrections of the model parameters. When exploring different multi-band appearance representations Stegmann and Larsen [4] studied the segmentation accuracy of facial AAMs at different scales in the range $10^3 - 10^5$ pixels obtained by pixel averaging. Wolstenholme and Taylor [6] incorporated a truncated Haar wavelet basis into the AAM framework and evaluated on a brain MRI data set at a compression ratio of 1:20. Later, Stegmann and Forchhammer [7] further evaluated the use of the Haar wavelet as well as the Cohen-Daubechies-Feauveau [8] wavelet family in the AAM framework. Compression rates of 1:40 without compromising segmentation accuracy were obtained.

Donoho [9] suggested a wedgelet representation for the texture as a means of edge detection and image compression. An image is represented by a collection of dyadically organized indicator functions with a variety of locations, scales and orientations. The classification and regression tree (CART) algorithm [10] uses sequential binary splitting of the spatial domain parallel to the coordinate axes, with splits allowed at every data point. In contrast to this the wedgelet regression tree obey special constraints. Only dyadic partitioning (i.e. recursive midpoint splitting) is allowed, with the added feature that at each terminal node a set of affine splits are also applicable. The wedgelet tree is a quad tree [11] with terminal nodes being either a dyadic (degenerate wedgelet) or a affinely split dyadic (non-degenerate wedgelet). The constrained splitting leads to fast algorithms. Within each resulting image terminal node (wedge or square) the pixel values are regressed to their mean value.

In this thesis we generalize the wedgelet transform to triangulated domains (cf. triangulated quad trees [12]). This has the major advantages of rendering the wedgelet representation independent of piece-wise affine warps of the triangulated domain. Such piece-wise affine warps is customarily chosen in AAM for their speed [4] and the triangulated wedgelet representation thus embraces this choice. The wedgelet transform results in a truncated change of basis for the texture and is represented by a regression tree. The regression tree is estimated using the minimization of the cross validation prediction error across the training set.

The segmentation accuracy in a wedgelet based AAM is evaluated for a case of human face segmentation using cross validation.

To regularize the orientation of the resulting wedgelet representation with respect to continuous edges Markov Random Field (MRF) [13][14][15] methods are tested. MRF is a frequently used method for estimations of global properties such as edges. MRF has been used for edge enhancing smoothing and improvements of classification results such as in the Potts model [16]. A MRF that fits into a regression tree based on wedgelets enforcing a geometric property of connecting edges is introduced and the results are evaluated as a visual and compared to the unregularized wedgelet based AAM.

0.1 The data

The data set used in this thesis is restricted to faces. The set consists of 37 faces of people, 7 females and 30 males. It's mostly young people between 20 and 30 years of age. Each face has been annotated with 58 landmarks by an expert. The resolution of the individual images is 640×480 and are in color. The data set could as well have been gray level, but since they were available in color, the data has been processed as color images as the results show.

Outline

First we introduce Appearance Model (AM) and AAM to clarify the domain we are working on. Then we introduce CART in chapter 2, primarily regression trees and cross validation since this is an important part of wedgelets which is the primary area of this thesis. Before we introduce wedgelets, we shortly present wavelets in chapter 3 since wavelets are closely related to wedgelets. Chapter 4 intruduces the original wedgelet formulation. Chapter 5 introduces barycentric coordinates, a very useful basis which makes it easy to define wedgelets on triangles. This lead us on to the the generalization of wedgelet decomposition to triangles in chapter 6. The results of the wedgelet decomposition in conjunction with the AAM is presented in chapter 7. Chapter 8 introduces Markov Random Fields and a recursive implementation on the wedgelet decomposition will be discussed in chapter 9. The results of the implementation is show and discussed in chapter 10 and finally in chapter 11 the conclusion and future work is presented. Appendix A contains the key algorithms used in this thesis and appendix B a paper written over the results of this thesis submitted for GMBV and appendix C contains some additional results.

Chapter 1

Active Appearance Models

We start by introducing the Active Appearance Model (AAM). This is the model that we want to improve the performance of. To understand the novelty and performance improvements this thesis describes, it is important to understand how the AAM works.

The AAM is a statistical model, highly efficient in recognizing objects belonging to a certain class. The AAM was first introduced in 1998 by Cootes et al in the award winning paper [17] and has since then been investigated and improved by numerous people including Mikkel B Stegmann [18]. The AAM has been used for e.g. face recognition, eye tracking, face tracking and for measurements of the blood circulation in MR scans of the human heart [19] with great success. But most of the tasks has been conducted off-line or on very low-resolution images due to the lack of efficiency of the AAM. When dealing with large data sets, such as high resolution images in 2D and 3D, problems emerge. Due to the amount of data the AAM becomes slow, hence there is a need to optimize the AAM in a way that will make it feasible to employ the model on e.g. high resolution medical images.

1.1 Introduction to the AAM

We Start with the observations. The object we are looking at, is considered an observation. There are a lot of ways of defining an observation when working with images. For a lot of classification problems each individual pixel is viewed as an observation, but for purposes such as this, the entire image might seem more suitable to perceive as an observation. Since we are making a highly specialized model we are only interested in objects belonging to this particular class. There is a small problem though, the perception of an object may vary from object to object or observer to observer, so we need to set some standard measurements when classifying an object. Since we are working with objects in images, the most straight forward measurements are the texture and the shape of the object, which is referred to as the appearance form here on. Based on these features we need to

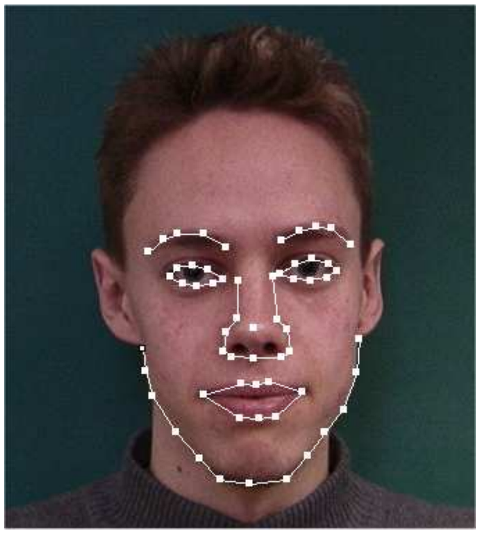


Figure 1.1: An annotated facial image .

derive an observation vector to use in a statistical model. This observation vector can be viewed as a point in a hyper dimensional feature space.

1.2 AAM Observations

The texture observation is at first glance quite simple. Working with sampled images it's simply the pixels of the image, be it gray-level or color. However, these are often highly dependent on their position in the image i.e. they depend on the shape. The other part of our observation, the shape, is another matter. We define the shape as follows.

Definition 1 *Shape is what geometrical information you have left when you remove scale, rotation and translation [20].*

First we need to define this ”shape-perturbation” to do this to. Working with faces ,which is the data set of this thesis, we define some anatomical landmarks in a certain order to make out the shape. Fig. 1.1 shows a annotated face. The outline of the face has been marked as well as the eyes, the nose and the mouth. Basically all the distinct features of the face are included in the shape. These now make up the feature vector containing the shape. Each face has to be annotated the same way, meaning that the same features have to be found in all images and ordered in the same way in the vector to ensure that the shapes span the same space. Having annotated a training set, the images have to be aligned shape wise, which can be achieved through Procrustes alignment (for details see [20]). From this, the mean shape is easily derived. All image texture are warped into this mean-shape. To decide which pixels goes where, a piece wise linear warp is performed. The outline

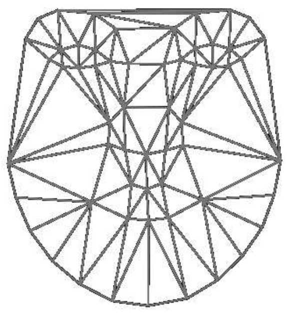


Figure 1.2: The mean shape.

of the individual areas are derived from the reference-shape which is triangulated using Delauney triangulation [21] which has the nice property of making the angles as wide as possible, avoiding some numerical problems. The triangulated mean-shape is shown in Fig. 1.2. The warp into the mean-shape makes it possible to sample all textures the same way and with the same number of samples ensuring that the texture vectors also span the same feature space. This results in the texture vectors as well as the shape vectors having the same length for all images making it possible to build a statistical model of the object class. It is computational infeasible to work with the data as they are i.e. each observation being a vector of 10000+ observations, so a very sparse model is needed. The next section will present a way to create such a sparse model.

1.3 Appearance Model

Let x_{ns} denote the shape vector and let x_{nt} denote the texture vector for the $n'th$ observation x_n so that

$$x_n = \begin{pmatrix} x_{ns} \\ x_{nt} \end{pmatrix} \tag{1.1}$$

These vectors are quite large, especially the texture vector, so it is desirable to reduce these vector-sizes. Let the training set, consisting of N observations, be denoted $X = \{x_1, x_2, \ldots, x_n\}$, $n = 1, \ldots, N$ which is then normalized. Having normalized the training set X, we then apply principal component analysis (PCA) giving the following orthogonal model for the shape variations.

$$s = \bar{s} + P_s b_s \tag{1.2}$$

where $\bar{s}$ is the mean shape, P_s is the modes of variation and b_s the shape parameters. A similar process is done to the normalized gray level components i.e. the texture vector giving $t = \bar{t} + P_t b_t$ for the texture variations. Having done this the appearance can be summarized by the two vectors b_s and b_t. Further correlation between the two vectors can be eliminated by performing another PCA on the set of concatenated vectors b given by

$$b = \begin{pmatrix} Wb_s \\ b_t \end{pmatrix} \tag{1.3}$$

where W is a diagonal weight matrix which accounts for the unit differences between shapes and textures. From this we write b as follows

$$b = \begin{pmatrix} WP_s^t(s - \bar{s}) \\ P_t^t(t - \bar{t}) \end{pmatrix} \tag{1.4}$$

which is then subjected to PCA giving the appearance model

$$b = Qc \tag{1.5}$$

b bas zero mean since the shape and the texture model both have zero mean. The shape and texture can be expressed directly as a function of c.

$$s = \bar{s} + P_s W Q_s c, \qquad t = \bar{t} + P_t^t Q_t c \tag{1.6}$$

where

$$Q = \begin{pmatrix} Q_s \\ Q_t \end{pmatrix} \tag{1.7}$$

The appearance model for normal images has now been established. By varying c it is possible to create an artificial shape and an artificial texture to warp into the shape. Fig 1.3 shows the artificial mean-face and the first three modes of variation from an appearance model changed individually +-2 standard deviations. Fig. 1.3

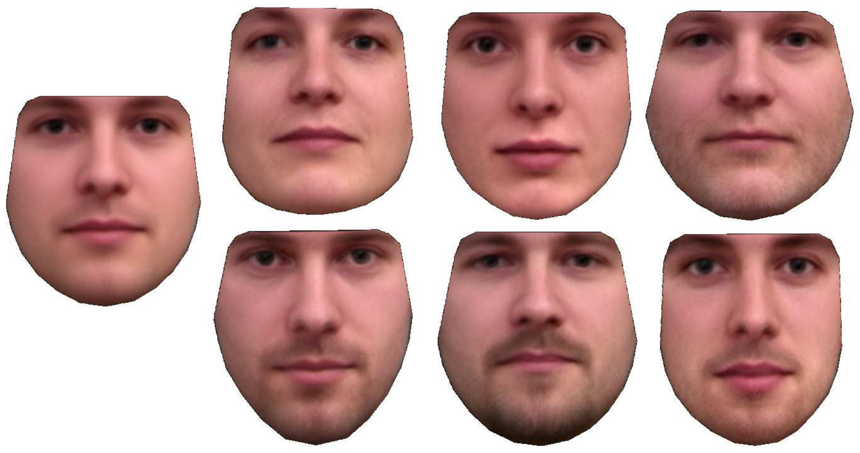


Figure 1.3: The three first modes of variation $+-2$ std.

shows the appearance model does exactly what it's supposed to do, namely it gives a low-dimensional representation of the appearance of the human faces. The use of the weight matrix in (1.3) is not essential but is as mentioned a compensation for the difference in units of measurements of the shape part of the model and the texture part of the model. For further information on how to calculate the weights extend beyond the scope of this thesis, the interested reader is referred to [17][22] for more information.

1.4 AAM

Having established the appearance model we want to make it active, in other terms we have to find a search algorithm. In his original paper [17] Cootes et al suggests a regression based method. However, since the first publication of the paper another method has been developed namely the Jacobian method [1]. This method is based on gradients which are pre-calculated in the texture-frame giving a change in the parameters directly. Let's just briefly go over this method. We define a vector that contains the differences i.e. a difference vector δI defined as follows

$$\delta\mathbf{I} = \mathbf{I_i} - \mathbf{I_m} \tag{1.8}$$

where $\mathbf{I_i}$ is the pixel values of the image, and $\mathbf{I_m}$ is the pixel values of the current model. Have in mind that this is a fitting of the model and the parameters are to be adjusted to make the model fit to the actual image as good as possible. To find the best match we define the magnitude as the measure $\Delta = |\delta\mathbf{I}|^2$ which we wish to minimize by varying the model parameters $\mathbf{c}$. Cootes states that even though this seems to be a high-dimensional optimization problem, each attempt to match a model to an image is actually a similar optimization problem. The spatial pattern of $\delta\mathbf{I}$ holds information about how the model parameters should be changed to achieve a better fit. Hence an iterative algorithm can be developed to minimize Δ by using the knowledge of the relationship between $\delta\mathbf{I}$ and $\delta\mathbf{c}$. Given the appearance model

$$\begin{aligned} \mathbf{s} &= \hat{\mathbf{s}} + \mathbf{Q_s c} \\ \mathbf{t} &= \hat{\mathbf{t}} + \mathbf{Q_t c} \end{aligned} \tag{1.9}$$

a synthesized shape can be projected on to the image and used for sampling the pixels into the shape free space. By projecting the model on the shape free space the difference between the texture and the model can be created

$$\mathbf{r}(\mathbf{p}) = \mathbf{t_s} - \mathbf{t_m} \tag{1.10}$$

where **p** are the parameters of the model. A simple and commonly used measure of difference is the sum of squares of elements of $\mathbf{r}$, $E(\mathbf{r}) = \mathbf{r}^T\mathbf{r}$. A taylor expansion of (1.10) yields

$$\mathbf{r}(\mathbf{p} + \delta\mathbf{p}) = \mathbf{r}(\mathbf{p}) + \frac{\partial\mathbf{r}}{\partial\mathbf{p}}\delta\mathbf{p} \tag{1.11}$$

Assuming our current residual is $\mathbf{r}$, we wish to calculate $\delta\mathbf{p}$ in such a way that we minimize $|\mathbf{r}(\mathbf{p}+\delta\mathbf{p})|^2$. By equating (1.11) to zero, the RMS solution gives

$$\delta\mathbf{p} = -\mathbf{R}\mathbf{r}(\mathbf{p}) \qquad \text{where} \qquad R = (\frac{\partial\mathbf{r}}{\partial\mathbf{p}}^T \frac{\partial\mathbf{r}}{\partial\mathbf{p}})^{-1}\frac{\partial\mathbf{r}}{\partial\mathbf{p}}^T \tag{1.12}$$

A fit made using the algorithm described above can be seen in Fig. 1.4 A thorough description of the fitting algorithm can be found in Cootes technical report [22].

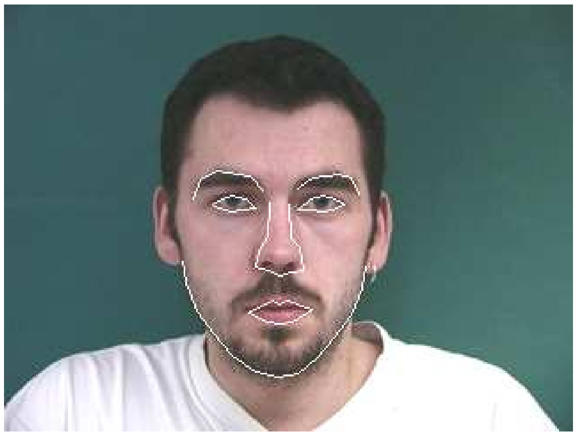


Figure 1.4: A fit if the AAM using the Jacobian method.

Chapter 2

CART

Classification And Regression Trees (CART) are an integrated part of the wedgelet decomposition we therefor shortly introduce them along cross validation which will prove very useful when combining wedgelets with the AAM

CART are a simple yet powerful tool to do both regression and classification. CART is thoroughly described in [10] by Friedman, Breiman, Olshen and Stone, furthermore CART is discussed briefly in [23]. Since CART is an integrated part of wedgelets we introduce them to help understand wedgelets little bit better. Lets start with an image which is basically a response Y i.e. the pixel value in images, to the pixel coordinate X_1 and X_2 taken on values between 0 and 1. Hence we now have some image function $Y = f(X_1, X_2)$ which we want to model by partition the space into sub parts with lines parallel to the boarders of the image. Fig 2.1 show such a split. Even though it seems easy describing the lines separating

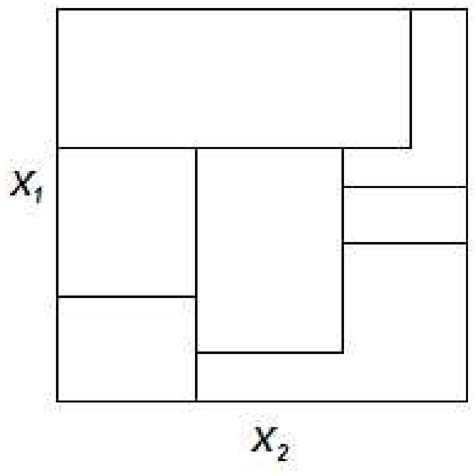


Figure 2.1: A split of feature space by partitioning with rectangles.

the partitions is quite simple, some of the partitions are actually quite difficult to describe. However, if we turn to binary trees [24] and restrict our selves to binary splits, the problem becomes easy to deal with. By using binary trees we split the feature into two regions and model the response in each partition by some function. One of the most common models for this function, is the case where the function is a constant c, often the mean value of the partition in question. Each partition is then split into two and so on until some stopping criteria is fulfilled. Fig 2.1 shows how a possible fit to the image could be. First a split of the image along $X_2 = t_1$

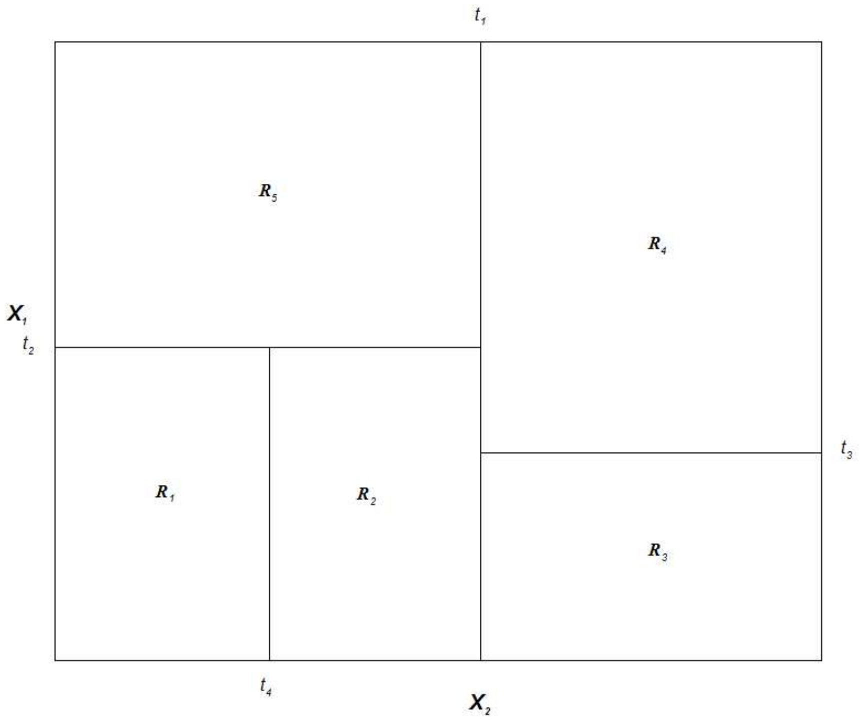


Figure 2.2: A split of feature space by partitioning with binary orthogonal splits.

then the right part is split into two along $X_1 = t_3$ and the left along $X_1 = t_2$. Finally the left lower part is split into two along $X_2 = t_4$ leaving us with five regions $R_1, R_2, \ldots, R_5$ as shown in 2.2. The resulting binary tree can be seen in Fig. 2.3 where the labels at the branches indicate the splits and the labels at the leaves tells which area. The regression model that predicts Y with a constant c_m in a region R_m is given in (2.1).

$$\hat{f}(X) = \sum_{m=1}^{5} c_m I\{(X_1, X_2) \in R_m\} \tag{2.1}$$

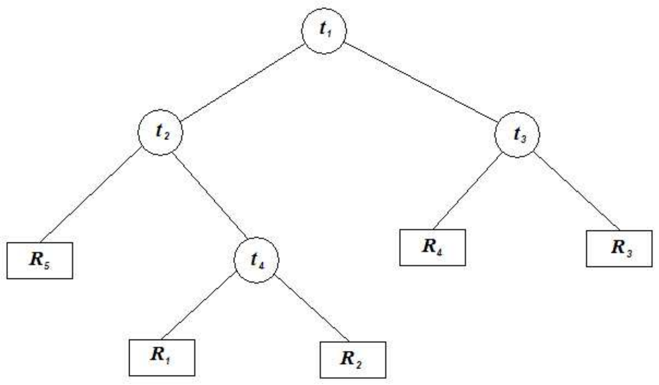


Figure 2.3: The underlying binary tree from Fig.2.2 .

2.1 Regression trees

A regression tree is a tree structure where the leaves are an approximation of the underlying data. The normal model used at the leaves is a constant, however, other functions is just as usable i.e. splines etc. Lets formalize the regression tree. For N observations (x_i, y_i), $i = 1 \dots N$ with p inputs $x_i = \{x_{i1}, x_{i2}, \dots, x_{ip}\}$ and a response y_i, an algorithm is needed that can decide on the splitting variables and points and the shape of the tree. Let us use the previous explanation as a starting point and use the residual sum of squares (RSS) (2.2) as the minimization criteria.

$$\sum (y_i - f(x_i))^2 \tag{2.2}$$

Partitioning into M regions $R_1, R_2, \dots, R_M$ and modeling the response as a constant as in (2.3) it is straight forward to see that the optimal c_m in R_m is given by (2.4).

$$\hat{f}(x) = \sum_{m=1}^{M} c_m I\{x \in R_m\} \tag{2.3}$$

$$\hat{c}_m = average(y_i | x_i \in R_m) \tag{2.4}$$

However trying all possible splitting variables and all possible split points quickly becomes infeasible. To solve this problem several algorithms are available [10][23]. Greedy top down methods where you grow the tree to some criteria and then the bottom-up approach where you start with growing a tree until a certain size of the end-nodes and then start collapsing internal nodes to create the optimal regression tree.
Searching for a sub tree $T \subset T_0$ by pruning T_0 is done by collapsing internal nodes having m regions with terminal node m representing region R_m. Letting

$|T|$ denote the number of leaves in T leads to the following.

$$\begin{aligned} \hat{c}_m &= \frac{1}{N_m} \sum_{x_i \in R_m} y_i, \\ Q_m(T) &= \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2 \\ C_\alpha(T) &= \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha |T| \end{aligned} \tag{2.5}$$

In this bottom-up method a cost complexity criterion like (2.5) is used instead of (2.2) to help avoid over fitting. The basic idea of (2.5) is to find $T_\alpha \subseteq T_0$ to minimize $C_\alpha(T)$. The parameter α is then the tuning parameter that determine the tradeoff between goodness of fit and the complexity of the tree. This results in (2.5) can be generalized and written as

$$PRSS = \sum (y_i - f(x_i))^2 + \lambda \cdot \#P \tag{2.6}$$

where $\#P$ is some complexity term and $\lambda = const \cdot \sigma$. σ is the standard deviation of the data. It is important to keep in mind that the tree need not be binary but can be of any desired complexity.
The regression tree has it's history from social science, however, CART has also been used in i.e. medicine. Here doctors use it e.g. to predict the mortality rate among patients within a certain time frame from the time that they are admitted to a hospital after having suffered a heart attack. The binary nature makes the result quite easy to interpret especially to questions with a true/false answer, something that is not easy with higher complexity of the underlying tree.

2.2 Cross Validation

It is often difficult to estimate general parameters such as those used in (2.6) and they often vary greatly between different data sets. However if there is sufficient data present, methods exists that can estimate the necessary parameters. Due to the nature of the data used by CART an often chosen way of estimating these parameters indirectly is cross validation. This method is one of the most widely used methods for estimating the prediction error. The general idea of cross validation is to divide the data into K groups hence the name K-fold cross validation. For the k part of the data fit the model on the remaining $K-1$ parts. Calculate the prediction error on the kth part. Repeat this for $k = 1, \dots, K$ and combine the K estimates of prediction error. Given a function that maps our observations as follows: Let κ:$\{1, \dots, N\} \to \{1, \dots, K\}$ i.e. map our observations into K groups, the cross validation can the be written as

$$CVE = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}^{k(i)}(x_i)) \tag{2.7}$$

using the l_2 norm i.e. RSS (2.2) as penalty function, $L((y_i, \hat{f}^{k(i)}(x_i))) = \|y_i - \hat{f}^{k(i)}(x_i)\|^2$ gives the following expression

$$CVE = \frac{1}{N}\sum_{i=1}^{N} \|y_i - \hat{f}^{k(i)}(x_i)\|^2 \tag{2.8}$$

Where $\hat{f}^k(x)$ is the function fitted with the k'th part of the data removed. The assignment function κ should assign data randomly to the K sets, and if $N = K$ we call this n-fold cross validation or leave-one-out cross validation. The parameter that needs to be estimated in (2.5) is α since we already know the complexity at the given level. So in this case we would calculate the PRSS with the proposed split and without the proposed split, then using our validation set, i.e. the set we have set aside to validate our result on, we would impose the two solutions found and calculate the PRSS for both. Having done this we select the best possible solution based on the results produced by the validation set.

Chapter 3

Wavelets

The purpose of introducing wavelets is primarily because they are the point of origin for the wedgelets. Donoho has long been a very respected researcher within the wavelet community and it is obvious that wedgelets have much in common with wavelets.

The following is primarily derived from [25] and [26] The wavelet is as it's name implies based on a wave. The idea is basically the same as with the Fourier transform (FT) namely frequency decomposition to a set of basis function hence making wavelets an image base, as the cosine and sine are when using FT. The idea is to have what we call a mother wavelet $\psi(t)$, which can be stretched, squeezed and translated to form different wavelets to make a set of basis functions which is used to decompose the signal. In image analysis the wavelet decomposition has been used in the JPEG2000 standard and Wolstenholme and Taylor [6] and Stegmann and Forchhammer [7] has used wavelets in the AAM as a mean of compression. We shortly introduce wavelets because wedgelets have a strong analogy to them. We start out with the continuous wavelet transform (CWT)

3.1 The Continuous Wavelet Transform

A wavelet $\psi(t)$ is a continuous, real or complex, function having the following properties.

- The function integrates to 0, this is also known as the admissibility condition

$$\int_{-\infty}^{\infty} \psi(t)dt = 0 \tag{3.1}$$

- The function has finite energy i.e.

$$\int_{-\infty}^{\infty} |\psi(t)|^2 dt < \infty \tag{3.2}$$

The CWT of a function $f(t)$ is given by

$$W(a,b) = \int_{-\infty}^{\infty} f(t) \frac{1}{\sqrt{a}} \psi^* \left(\frac{t-b}{a} \right) dt \tag{3.3}$$

where a is the scaling or dilation factor, b the time shift factor and $\frac{1}{\sqrt{a}}$ the energy normalizing factor. Let us set

$$\psi(t)_{a,b} \equiv \frac{1}{\sqrt{a}} \psi^* \left(\frac{t-b}{a} \right) dt \tag{3.4}$$

yielding

$$W(a,b) = \int_{-\infty}^{\infty} f(t) \psi(t)^*_{a,b} dt \tag{3.5}$$

The normalizing factor $\frac{1}{\sqrt{a}}$ ensures the following property always is true for all possible a and b

$$\int_{-\infty}^{\infty} |\psi(t)_{a,b}|^2 dt = \int_{-\infty}^{\infty} |\psi(t)|^2 dt \tag{3.6}$$

Fig. 3.1 shows a Haar wavelet, which is just one type, among other we mention the mexican hat, morlet etc. The signal which we desire to make a wavelet transform of is convolved with the desired wavelet yielding a response in scale and time. However the continuous wavelet transform is difficult to use due to the infinite number of scales and translations and hence have little practical use in the processing of images. For processing images we turn to the discrete wavelet transform (DWT).

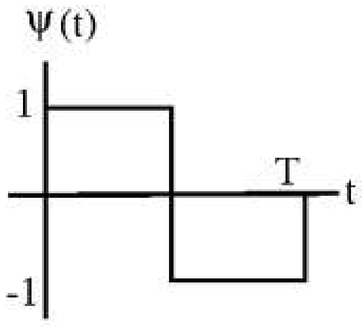


Figure 3.1: A Haar wavelet.

3.2 Discrete Wavelet Transform

Let V^j be some space spanned by 2^j orthogonal unit vectors, we call these basis functions scaling functions and denote them Φ. As an example $\Phi_0^2 = [1,0,0,0]$,$\Phi_1^2 = [0,1,0,0]$,$\Phi_2^2 = [0,0,1,0]$ and $\Phi_3^2 = [0,0,0,1]$ the basis of V^2. A simple set of these can be generated from the following

$$\Phi_i^j(x) = \Phi(2^j x - i), \qquad i = 0,1,\dots,2^j - 1 \tag{3.7}$$

$$\text{where} \qquad \Phi(x) = \begin{cases} 1 & \text{for} 0 \le x \le 1 \\ 0 & \text{otherwise} \end{cases}$$

Definition 2 *W^j is the space of all functions in V^{j+1} that are orthogonal to all functions in V^j under the chosen inner product.*

Definition 3 *Wavelets is a collection of linear independent functions $\Psi_i^j(x)$ spanning W^j*

These wavelets have three nice properties

- The basis functions Ψ_i^j of W^j together with Φ_i^j of V^j form the basis of V^{j+1}
- Every basis function Ψ_i^j of W^j is orthogonal to every basis function Φ_i^j of V^j under the chosen inner product
- All wavelets $\Psi_i^j(x)$ are orthogonal to one another

Lets start out with the *Haar* wavelets, which is a box basis. Fig 3.2 shows three orthogonal basis functions of the Haar wavelet basis, and is defined as follows

$$\Psi_i^j(x) = \Psi(2^j x - i) \qquad i = 0,1,\dots 2^{j-1} \tag{3.8}$$

$$\text{where} \qquad \Psi(x) = \begin{cases} 1 & \text{for } 0 \le x \le 1/2 \\ -1 & \text{for} 1/2 \le x < 1 \\ 0 & \text{otherwise} \end{cases}$$

All the above can of course be extended to signal of n dimensions but we continue this section with a small example on a 1D signal. Let $I = [6,8,4,6]$ be a signal we want to wavelet transform. This is done by convolving the image with our basis functions, where we in this case choose the Haar wavelets as basis. This gives us the $[c_0, d_0^0, d_0^1, d_1^1] = [6,1,-1,-1]$ with the Haar basis function $\hat{f}(x) = c_0\Phi_0^0 + d_0^0\Psi_0^0 + d_0^1\Psi_0^1 + d_1^1\Psi_1^1$. We can then totally reconstruct the signal the following way: We can now write the wavelet decomposition of a given signal $f(t)$ to a given coefficient wavelet d or scale c as

$$\begin{aligned} c_i^j &= \frac{1}{N}\sum_{t=1}^{N} f(t)\Phi_i^j(t) \\ d_i^j &= \frac{1}{N}\sum_{t=1}^{N} f(t)\Psi_i^j(t) \end{aligned} \tag{3.9}$$

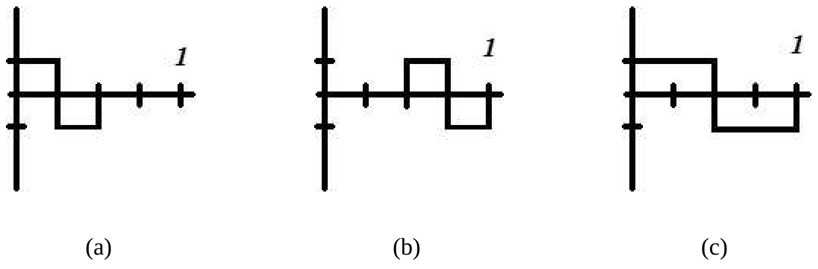


Figure 3.2: Haar wavelets basis function (a) $\Psi_0^1(x)$ (b) $\Psi_1^1(x)$ (c) $\Psi_0^0(x)$

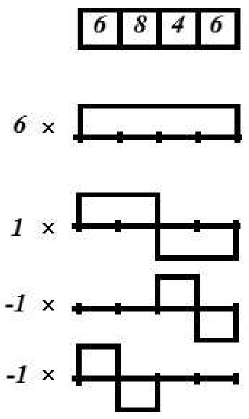


Figure 3.3: The reconstruction of the decomposed signal

and hence we write the reconstruction of f as

$$\hat{f}(t) = \sum_{j=0}^{J} \sum_{i=0}^{j} c_i^j \Phi_i^j + \sum_{j=0}^{J} \sum_{i=0}^{j} d_i^j \Psi_i^j \tag{3.10}$$

We will not get into details about compression, but just mention that this is done by truncating the coefficients hence giving a sparser basis than the original. The reconstruction of images from a wavelet compression is very good and the compressed images looks very alike the uncompressed image.

Chapter 4

Wedgelet Decomposition

Having established the background and context of the wedgelets, the following will introduce the wedgelets in their original form on a dyadic domain.

Wedgelets was first presented in [9] by D. Donoho in (1999) and are closely related to wavelets. D. Donoho has long been a very respected researcher within the wavelet community with numerous publication within this field. The use of wedgelets becomes very interesting in conjunction with appearance models, but they need a little modification to work on the AM. Before we do this, we start by introducing wedgelets and how they work.

4.1 Wedgelet

The basic idea of wedgelets is to represent changes in the texture of an underlying image with wedges. When dealing with images one of the most distinct features is edges, these are represented as steep changes in the pixel values and thus becomes visible as edges. To get a better perception of what wedgelets are, we start out with binary images. Fig 4.1 shows a binary image with some edge between the white and the black part of the image. The question is how we approximate the edge in a reasonable fashion. The most obvious way would be simply to draw a line across the image from the intersection of the edge on the right side to the intersection on the left side as shown in fig 4.2(a). This however, is not a good solution for contours with higher curvature than the one presented in Fig. 4.1 and hence the approximation presented by the line is not satisfying. This is, however, a good starting point. If we, instead of representing the curve as a straight line, represent the curve as a piecewise straight line, the result would easily become much better. Fig 4.2(b) shows the piecewise linear approximation to Fig. 4.1. It shows that the deviation from the real image becomes significantly smaller. The problem is to find and describe these points where the piecewise linear function changes i.e. the discontinuities in the first order derivative. This is where the wedgelet decomposition comes into the picture. We start by defining a wedgelet

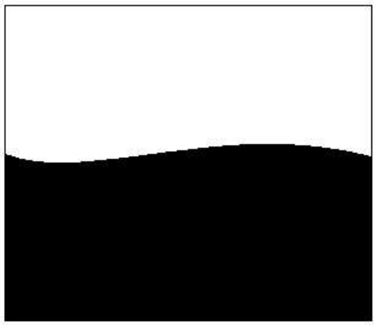


Figure 4.1: A binary image with some edge dividing the image into a black part and a white part.

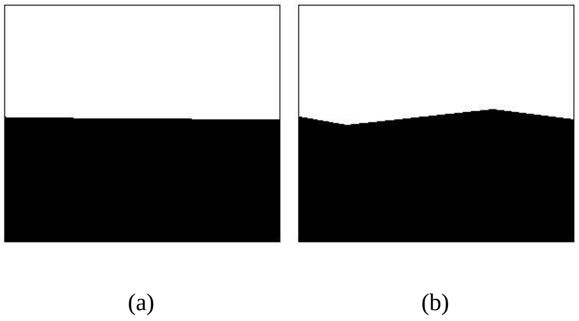


Figure 4.2: (a) A Line single approximation (b) A piecewise linear approximation.

Definition 4 *A wedgelet w is a function on a dyadic d that is piecewise constant split in two by some edge e, having the intersection points v_1 and v_2 with the perimeter of d , dividing the dyadic in to two areas with different values c_a and c_b.*

Hence we write a wedgelets as

$$w = \{c_a, c_b, e\} = \{c_a, c_b, v_1, v_2\} \tag{4.1}$$

The edge e i.e. v_1 and v_2 determine the orientation of the wedgelet and the constants c_a and c_b determines the profile, as in Fig.4.3 which shows the profile and the gray level wedgelet.

Definition 5 *A degenerate wedgelet w is a function on dyadic d that is constant on all of d taking on the value c.*

We can think of the degenerate wedgelet as a wedgelet where the edge e does not pass through d. This leaves us with two types of wedgelets, degenerate wedgelets and nondegenerate wedgelets.

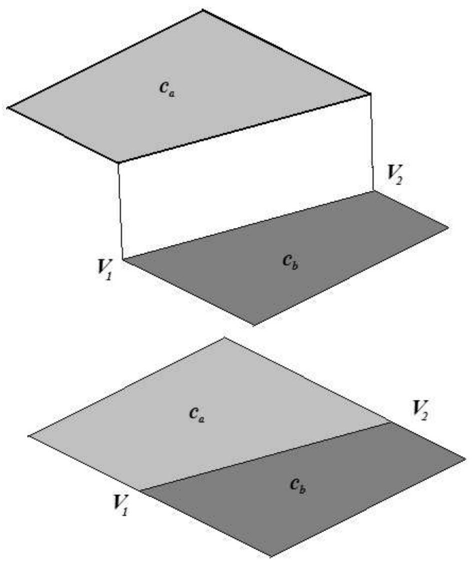


Figure 4.3: A wedgelet represented in two ways. The upper basically shows the profile of the wedgelet on an image. The lower shows the same just as gray level values.

4.2 The Wedgelet Decomposition

The wedgelet decomposition can be formalized the following way. Let I be an image on $[0,1]^2$ and $A_{j,k} \subset [0,1]^2, k = \{k_1, k_2\}$ be a dyadic at scale $j \in Z_{+} \cup \{0\}$, $A = [[k_1/2^j, (k_1+1)/2^j] \times [k_2/2^j, (k_2+1)/2^j]]$ where $0 < k_1, k_2 < 2^j$ for an integer $j \geq 0$
Each A can be divided arbitrarily into two by an edge e from v_1 to v_2, where v_1 and v_2 is the intersection of the perimeter of A and e. This is what we call a wedge. It takes on two different values $\widehat{c_a}$ above e and $\widehat{c_b}$ below, which are the respective mean values. A wedgelet is therefore denoted $w = \{v_1, v_2, \widehat{c_a}, \widehat{c_a}\}$
$\widehat{c_a}$ and $\widehat{c_b}$ is found for each orientation of the wedgelet w by averaging over the region R_a above e and over R_b the region below e, hence we write

$$\begin{aligned} \widehat{c_a} &= Average(I(A_{j,k})|R_a) \\ \widehat{c_b} &= Average(I(A_{j,k})|R_b) \end{aligned} \tag{4.2}$$

$$RSS = \parallel y - \mu \parallel^2 \tag{4.3}$$

We are now able to formulate the wedgelet decomposition. The wedgelet decomposition $W(I(A_{j,k}))$ is a collection of projections of $I(A_{j,k})$ onto a finite set of wedgelets. For each wedgelet $w(A)$ we select the one that minimizes (4.3) over $A_{j,k}$. The nondegenerate wedgelet is found through an exhaustive search where we, at the given scale for all points on the perimeter, try all possible combinations, thus making it possible to find the optimal solution. Fig. 4.4 shows all possible combinations from one point on the perimeter. For an example of a wedgelet

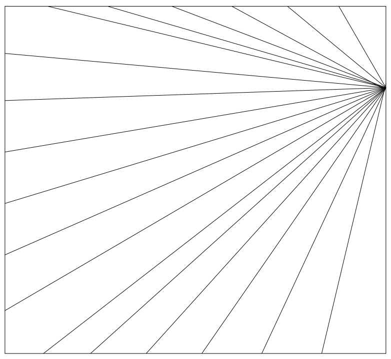


Figure 4.4: All possible configurations from a single point on the perimeter of a dyadic at a given scale.

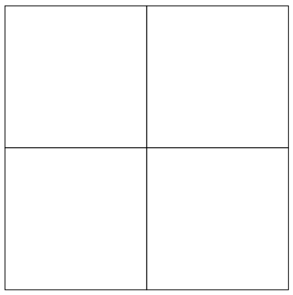


Figure 4.5: The dyadic split used in wedgelet decomposition.

decomposition se fig 4.6(b). This however, haven't brought us any further than the situation in fig 4.2(a), but if we worked at a finer scale this could solve some of the problems in the decomposition and bring us closer to our goal. To do so we make the wedgelet decomposition multi scale by splitting the dyadic into 4 new dyadic by splitting at the middle of each side as shown in 4.5. This leaves us with three templates for the wedgelet decomposition at each scale as fig. 4.7 shows. By going into multi scale and by embedding this into a quad tree-structure this virtually becomes a regression tree with orthogonal splits except at the leaves. Here the splits are affine and non-orthogonal but still explicitly defined and actually the basis function at this given site at this given scale like a wavelet would be. The resulting tree-form and wedgelet scales could look like fig. 4.2. This obviously leads us to use the penalty associated with this as explained in sec. 2. (4.4) shows the complexity penalized residual sum of squares (CPRSS) in a simplified form

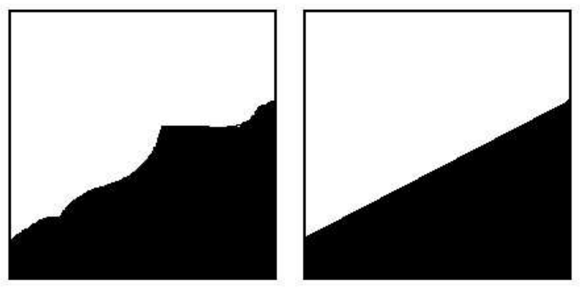


Figure 4.6: (a) $A_{j,k}$ (b) $w = \{v_1, v_2, \widehat{c_a}, \widehat{c_a}\}$.

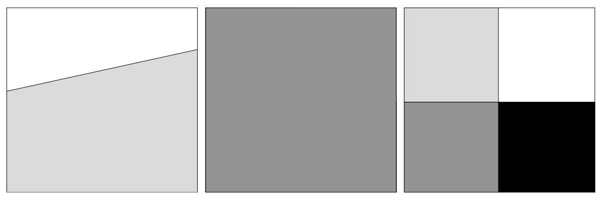


Figure 4.7: (a) Wedgelet (b) Degenerate wedgelet (c) Step through scale space.

which we will use for the wedgelets.

$$CPRSS = \| y - \mu \|^2 + \lambda \cdot \#P \tag{4.4}$$

where $\lambda = const \cdot \sigma^2$ and $\#P$ is the complexity

Having put the wedgelets into this tree structure and shown that in fact the wedgelet decomposition is a regression tree, we now state the multi scale wedgelet decomposition
The optimal multi scale decomposition $W^j(I)$ is the collection of $W(I(A_{j,k}))$ for all $A_{j,k}$ that minimizes (4.4). We write the decomposition as

$$W^j(I) = \{W(I(A_{j,k})) : j = 0..J, k_1..k_n = 0..2^j\} \tag{4.5}$$

It can be shown that the optimal complexity penalty for a wedgelet decomposition based on CART is given by

$$\lambda = (\xi \cdot \sigma(1 + \sqrt{2log_e(\#W)}))^2 \tag{4.6}$$

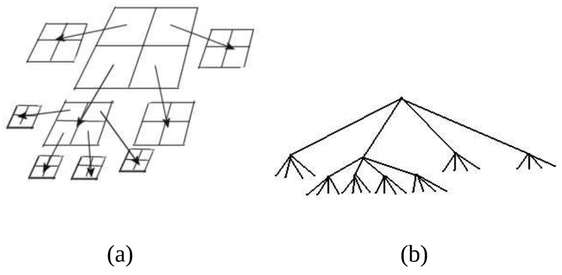


Figure 4.8: (a) A wedgelet decomposition in which the wedgelets can be degenerate or non-degenerate. (b) The regression tree associated with the decomposition in 4.8(a).

where $\xi < 8$ is a fixed constant, $\#W$ is the total number of wedgelets as $|T|$ in (2.6) and σ denotes the variance of the image. A thorough discussion and a proof can be found in [9]. Fig. 4.9 shows how a possible decomposition in the dyadic domain might look.

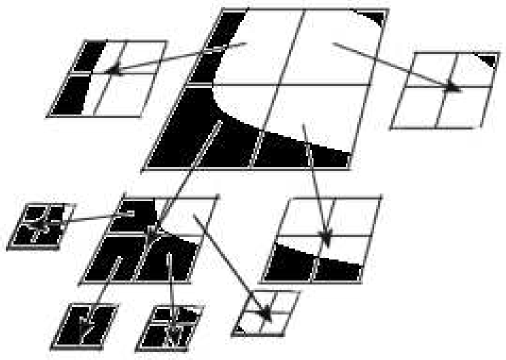


Figure 4.9: A possible wedgelet decomposition of a given image.

4.2.1 The Wedgelet Decomposition Algorithm

If we just went head on and used a top down approach, we would run into some computational difficulties. Say we have an image I on a dyadic d and each side can be split into two n times, we would easily end up with an algorithm having a complexity in the proximity of $O(n^4)$ since n^4 is the number of possible edges. However, if we look at the approach used in regression trees where the entire tree is

grown first. The resulting regression tree is then created by collapsing inner nodes. Using this approach we have an algorithm that suits our purpose and has a more acceptable performance, namely a complexity in the proximity of $O(n^2)$. This is due to the fact that the number of possible edges are reduced to approximately $log(n)n^2$ by employing the dyadic split and only allowing edges from boarder to boarder of the dyadic at all scales.
We will now draw the outline of the decomposition algorithm for multi scale wedgelets

1. On the Image I grow the entire regression tree so the each leaf has size $\frac{1}{J}^n$, where n is the dimensionality of the space the image lives in.
2. Set the level $l = log_2(n)$
3. At level l for all wedgelets calculate the $a \leftarrow CPRSS$ for each wedgelets without split, and the $b \leftarrow argmin(CPRSS)$ with a split
4. At level ($l - 1$) calculate the $CPRSS\ c$
5. $d \leftarrow$Select $min(a, b, c)$
6. If $d = a$ or $d = b$ mark this wedgelet terminal and leave branch
7. Else if no children of a wedgelet at level ($l - 1$) is terminal, collapse them
8. set $l = l - 1$
9. Repeat from 3 until finished

This can be stated as a recursive algorithm which can be found in app.A

4.3 The Origin of Wedgelets

As stated in the beginning of this chapter, the wedgelet has a lot in common with some other interesting methods. Donoho has a great interest in both statistics and in harmonic analysis and this is reflected in the wedgelets. The following sections briefly describe the relations between wedgelets and to two other methods, one from statistics and one from harmonic analysis.

4.3.1 The Wedgelet Decomposition and CART

The wedgelet decomposition have a lot in common with the regression tree presented in chapter 2. First of all they are both trees, and secondly they are both composed by orthogonal splits at the internal nodes. Finally the decomposition suggested in [9] uses the CPRSS (2.6) (complexity penalized residual sum of squares). Further more the algorithm suggested to use in the wedgelet decomposition is very similar to the one of the standard algorithms used in CART as explained in chapter 2 . This let us state that the wedgelet decomposition is basically a regression tree, with affine splits at the leafs.

4.3.2 Wavelets and Wedgelets

The wedgelet decomposition consists of basis functions at multiple scales in the same way as the wavelets. But there is a significant difference between the two. The wedgelet decomposition borrows some properties from regression trees, namely an ability to locally adapt the basis function to the image. Furthermore the reconstruction of the image from a wedgelet decomposition only rely on one basis function for the reconstruction of a given site or pixel whereas the wavelet depends on multiple basis functions to reconstruct the same site. However wavelets tend to give a result more pleasing to the eye than wedgelets, which is due to the wedgelets step function like basis which produces an image that is more visually displeasing. Wedgelets, however, are very good at reconstructing edges as to wavelets that can produce ringing, a phenomenon know from Fourier-analysis, and it is not given that even though we have a more displeasing image using wedgelets, the loss of information is higher.

4.4 Connecting Edges

Donoho mention in his paper [9] that having a function $f \in [0;1]^2$ the lattice created by splitting the dyadic can be used a an ’espalier’ where it is possible to ’string’ the function to the frame. Then by examining the intersections of the curve and the lattice a method of connecting the approximation to the function f i.e. the edgelets across the boundaries can be coupled into chains, but a real geometric constraint is not introduced. Another method for improving the edge structure can be found in [27]. This method, however, is not based on CART but on a Markov model which is solved through dynamic programming yielding good results at least for binary images. This model incorporates geometric constraints. We will return to this in chapter 9

Chapter 5

Barycentric coordinates

Before we modify the wedgelets a new coordinate system has to be introduced. This will make the triangle based wedgelet decomposition much easier to formulate.

Most images used today in image processing are represented by pixels which have the well known quadratic form. The pixels are a standard mostly adopted from the way that the image is sampled on CCD's etc. However, as we have seen in the appearance model, the mean shape has been triangulated and the texture is warped into a shape free environment piecewise linearly. This inspires to the thought of changing the basis of the image to something which directly fits into the triangles of the mean shape used in the AM. Instead of the rectangular coordinate system used with square pixels we want to use triangles and thus find a suitable coordinate system for these. As fig. 5.1 shows, triangles is just as good a basis for images as squares. The lower right part of the image is a normally sampled image with square pixels, and the top left part of the image is with triangular pixels. A coarser sampling of the upper left part of the same image is shown in fig. 5.2. It should be noted that these images are of cause computer generated and the basic sample unit is therefore still quadratic pixels, but the images still gives the impression intended, that triangles are as good as squares. This of cause also changes the shape of the image as the figure shows, but this can be compensated for with two triangles. In fig. 5.1 there is no visible difference between the two parts with different base and in right the difference is due to differences in sample density. To formalize this we state that in an image $I[0,1]^2$ with $n \times n$ pixels, a given pixel $o \in I$ at position $p = (p_1, p_2), 0 \leq p_1, p_2 < 1$ can be thought of as a function f that is constant over the interval $[p_1, p_1 + 1/n]x[p_2, p_2 + 1/n]$. The pixels based on triangles can be defined in the same way, however to do this we need a more suitable coordinate system that support this.

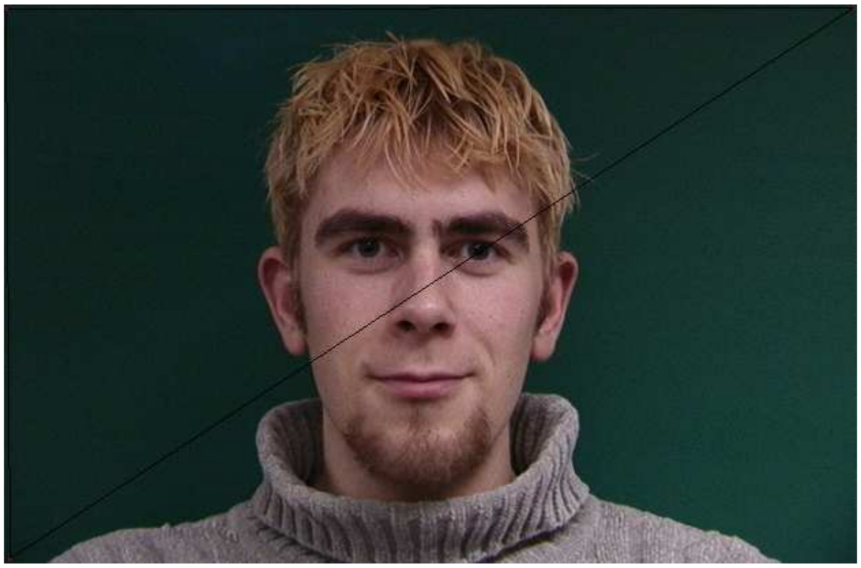


Figure 5.1: An image where the upper left part is based on triangles instead of normal square pixels.

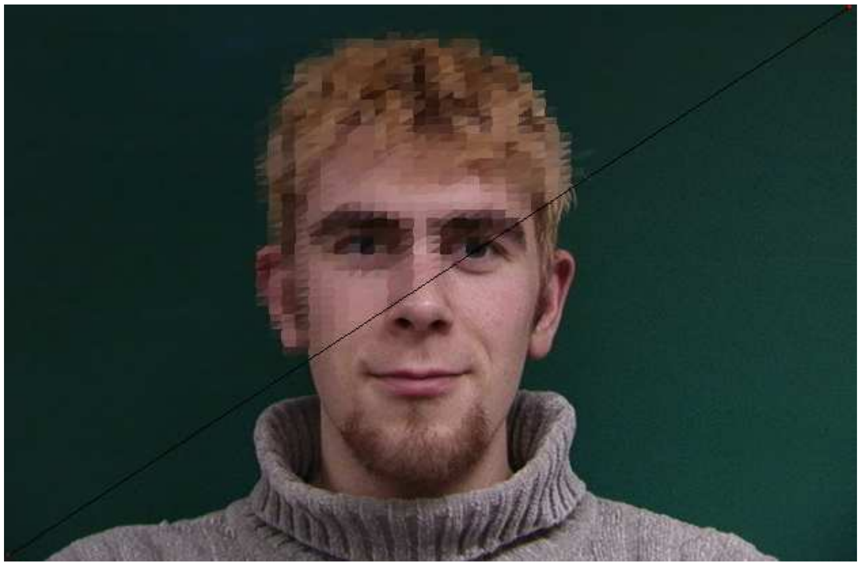


Figure 5.2: The same as fig. 5.1 just with triangles on a coarser scale.

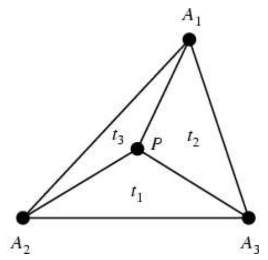


Figure 5.3: Barycentric coordinates.

5.1 Homogeneous Barycentric Coordinates

To help solve our problem homogeneous barycentric coordinates known from computer graphics and e.g. convex analysis are introduced. Original barycentric coordinates was introduced by Mobius [28] in 1827. A barycentric coordinate system is a local coordinate system for triangles in two dimensions, but can easily be extended to tetrahedrons of higher dimensionality. Barycentric coordinates are triples of numbers (t_1, t_2, t_3) corresponding to masses placed at the vertices of a reference triangle $\triangle A_1A_2A_3$. These masses determine a point P, which is the geometric centroid of the three masses Fig. 5.3. The vertices of the triangle are given by $A_1 = (1, 0, 0)$,$A_2 = (0, 1, 0)$ and $A_3 = (0, 0, 1)$ since they are homogeneous. For such a homogeneous coordinate system within a triangle the areas of $\triangle A_2PA_3$, $\triangle A_1PA_3$ and $\triangle A_1PA_2$ are proportional to t_1, t_2 and t_3. For homogeneous barycentric coordinates the following applies:

Definition 6 $\sum_i t_i = 1$, *hence every point* P *with coordinate* $c = (t_1, t_2, \ldots, t_i)$ *where* $0 \leq t_1, t_2, \ldots, t_i \leq 1$ *lie within* $\triangle A_1A_2 \ldots A_i$, $i \in N^+$.

We restrict ourselves to $i = 1, 2, 3$ i.e. 3-dimensional barycentric coordinates, namely triangles in 2D. Moving from image coordinates to barycentric coordinates constitutes a shift of basis. We simply project our image coordinates onto a plane, here in 3 dimensions as shown in Fig. 5.4. It can be shown that this is a basis and it is obvious that it is orthogonal since the three base vector are linear independent. So now we have established a new image base, which will fit the triangulation used in the AAM and we formalize these pixels the following way. In an image $I[0, 1]^3$ with n^2 pixels, a given pixel $x \in I$ at position $p = (p_1, p_2, p_3), 0 \leq p_1, p_2, p_3 < 1$ can be thought of as a function f that is constant over the interval $[p_1, p_1 + 1/n] \times [p_2, p_2 + 1/n] \times [p_3, p_3 + 1/n]$. The barycentric coordinate system is a warp-free space for tetrahedrons where we have a unique transfer function, giving point to point correspondence among tetrahedrons of the same dimensionality. Assuming that we know the position of the triangles vertices v_1, v_2 and v_3 in the image the area A of the triangle can be calculated. The projection $p_{x,y} \curvearrowright p_{a,b,c}$ from the

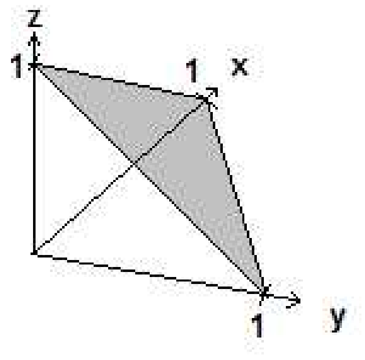


Figure 5.4: Barycentric coordinates in 3D space.

quadratic pixel domain to the barycentric is then given by

$$f(p_{x,y}) = (\frac{(v_3 - v_2) \times (p - v_2)}{2A}, \frac{(v_1 - v_3) \times (p - v_3)}{2A}, \frac{(v_2 - v_1) \times (p - v_1)}{2A})$$

$$= (p_a, p_b, p_c) = p_{a,b,c} \tag{5.1}$$

where $\times$ denotes the cross product and $p_{a,b,c} = (p_a, p_b, p_c)$ is the result in homogeneous barycentric coordinates. The inverse projection $p_{a,b,c} \curvearrowright p_{x,y}$ is hence given by

$$f(p_{a,b,c}) = (p_a v_1 + p_b v_2 + p_c v_3) = p_{x,y} \tag{5.2}$$

where $p_a v_1$ is the vector $(p_a x_{v1}, p_a y_{v1})$

Chapter 6

Triangle Based Wedgelets

We have now established how the wedgelet decomposition originally was formulated. This chapter will generalize the wedgelet decomposition to a triangular domain.

The original formulation of wedgelets cannot be used in conjunction with the AM, since the shape of the model has been triangulated. However, this does not mean that the wedgelet decomposition becomes unusable. Instead of using dyadic wedgelets we want to use triangular wedgelets. Since there is no formulation for these one must be made. However, to save a lot of work we want to preserve as much of the original formulation as possible. Donoho states that this is just straight forward for any geometry and since our proposed changes are minute we proceed without hesitation. First we have to define our new geometric settings.

6.1 Geometric Settings

We want to have a similar setting to the one we have with dyadic, meaning we have to make a degenerate wedgelet, a non-degenerate and a step through scale space i.e. some consistent subdivision scheme. We start with the last problem, hence how do we consistently subdivide a triangle. A way is to find the center of the triangle and draw a line from each vertices to the center, splitting the triangle into 3 new triangles see Fig. 6.1(a). However, this approach will make two of the angles smaller and smaller for each subdivision which is not desirable since it would lead to numerical inaccuracy. It would be far better to split each triangle into new triangles similar to their ancestor with the same angels but half the side length i.e. an angle preserving split. This can be achieved by placing a point on exactly the middle of each side, and then connect each point to the two other points on the middle of the sides of triangle as shown in Fig. 6.1(b). By using barycentric coordinates as described in chapter 5 this becomes a very simple task. The points are given by $[0.5, 0.5, 0]$, $[0.5, 0, 0.5]$ and $[0, 0.5, 0.5]$ all that has to be done is to connect them. To get a better understanding of what is happening, we project the

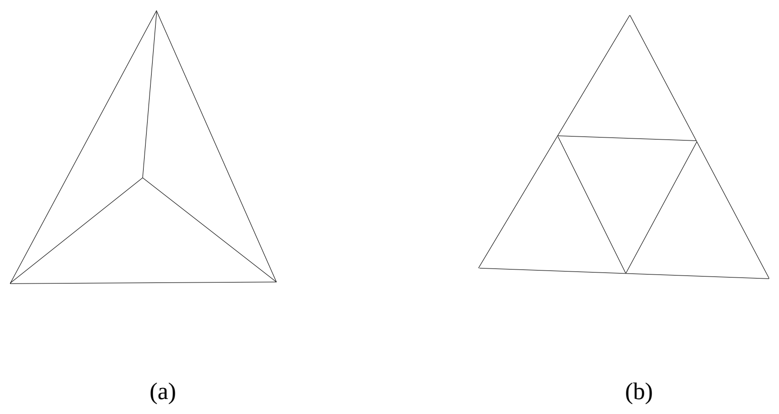


Figure 6.1: (a) A consistent subdivision of a triangle where the angels are not preserved but split in half (b) A consistent subdivision of a triangle where the angels are preserved.

triangle onto the barycentric space as shown in fig. 6.1. Here we can see that these subdivision lines, from the middle of each sid to the middle of one of the other sides, is the intersection of a plane parallel to the planes spanned by the axis at 0.5 on each of the axis. The intersection of these planes and the triangle represents exactly the solution we are looking for. Fig. 6.1 shows two of the planes, a triangle in the barycentric coordinate system and the intersection between them. The plane parallel to the yz-plane an hence the intersection of the plane and the triangle is given by

$$\begin{array}{rl} plane & f(x=0.5,y,z)=0.5+x+y \\ intersection \quad line & f(x=0.5,y,z)=0.5+x+y=1 \end{array} \tag{6.1}$$

The last intersection can be made as the intersection of the plane parallel to the xy-plane and the triangle. The expression for the line is derived using the knowledge that the barycentric coordinates always sum to one. A similar expression can of cause be derived for the other lines used in the subdividing the triangle. Note that this sub-division is totally independent of the original shape and size of the triangle. Another property of the barycentric coordinates is the easy indexing along the sides of the triangle. This comes in handy when constructing the non-degenerate wedgelets. Fig. 6.3 shows a possible scheme on a given triangle for finding the optimal non-degenerate wedgelet as described in chapter 4. The formulation for the nondegenerate and degenerate wedgelets on the triangulated domain is almost the same as the formulations given in chapter 4 Def. 4 and 5.

Definition 7 *A nondegenerate wedgelet w is a function on triangle t that is piece-wise constant split in two by some edge e, having the intersection points v_1 and v_2*

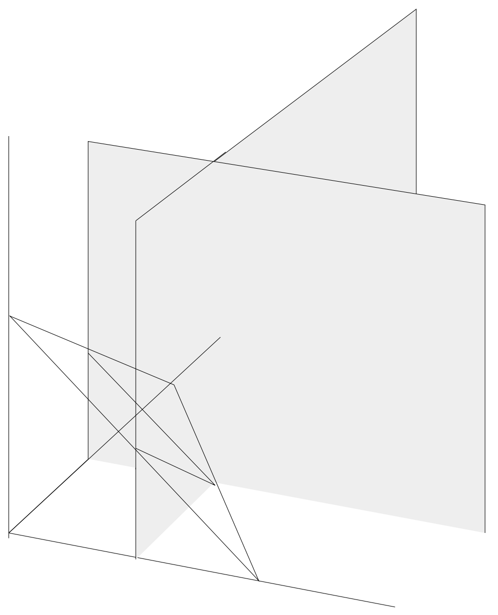


Figure 6.2: The intersection of two of the tree planes that are parallel to the planes spanned by the basis vectors of the coordinate system, and a triangle projected into the barycentric coordinate system.

with the perimeter of t , dividing the triangle in to two areas with different values c_a and c_b.

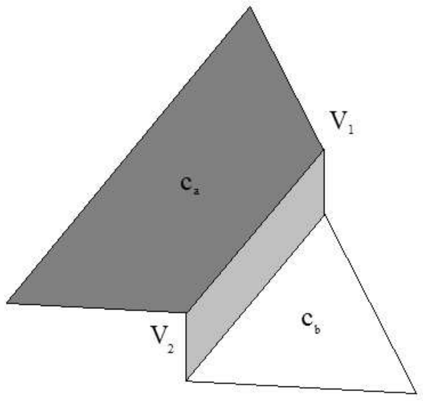
Hence writing a wedgelets on a triangulated domain as

$$w = \{c_a, c_b, e\} = \{c_a, c_b, v_1, v_2\} \tag{6.2}$$

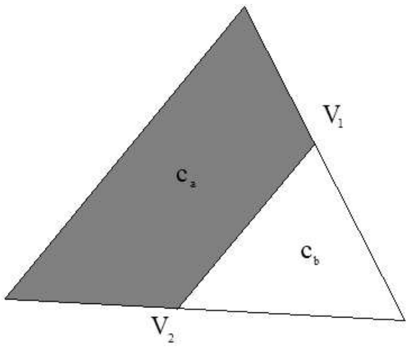
4.9

Definition 8 *A degenerate wedgelet w is a function on triangle d that is constant on all of t taking on the value c.*

We now have a way of splitting each triangle consistently and we have defined our three templates or wedgelets in correspondence with the formulation in chapter 4. Lets us have a look on the resulting wedgelet templates and their dyadic counterpart. Fig. 6.1 shows these figures and the correspondence is clear. Fig. 6.5(b) and 6.5(e) corresponds, Fig. 6.5(a) and 6.5(d) corresponds, and Fig. 6.5(c) and 6.5(f) corresponds, giving us a complete set of wedgelets on this triangular form.



(a)



(b)

Figure 6.3: (a) The nondegenerate wedgelet with triangular basis as profile (b)The nondegenerate wedgelet with triangular basis as gray level image.

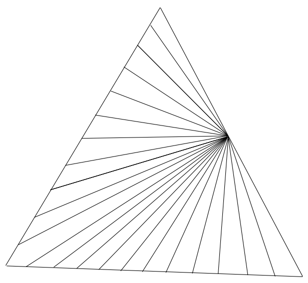


Figure 6.4: All possible edges in a triangle from a single point.

6.2 Triangle Based Wedgelet Decomposition

Having established the geometric foundation we can move along to the wedgelet-decomposition. We will now formulate the Wedgelet decomposition for triangulated domains in barycentric coordinates.

Let I be an image on $[0,1]^3$ and $A_{j,k} \subset [0,1]^3, k = \{k_1, k_2, k_3\}$ be a triangle at scale $j \in Z_+ \cup \{0\}$, $A = [k_1/2^j, (k_1+1)/2^j]] \times [k_2/2^j, (k_2+1)/2^j]] \times [k_3/2^j, (k_3+1)/2^j]]$ where $0 < k_1, k_2, k_3 < 2^j$ for an integer $j \geq 0$. Each A can be divided arbitrarily into two by a line l from v_1 to v_2, where v_1 and v_2 is the

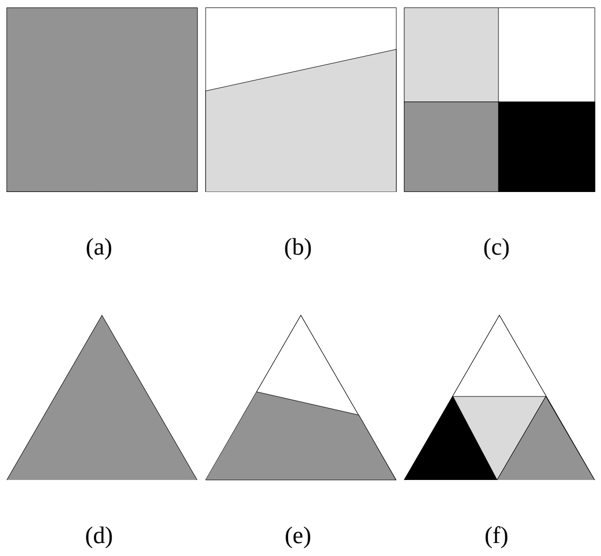


Figure 6.5: (a) degenerate dyadic wedgelet (b) nondegenerate dyadic wedgelet (c) scale dyadic wedgelet (d) degenerate triangular wedgelet (e) nondegenerate triangular wedgelet (f) scale triangular wedgelet.

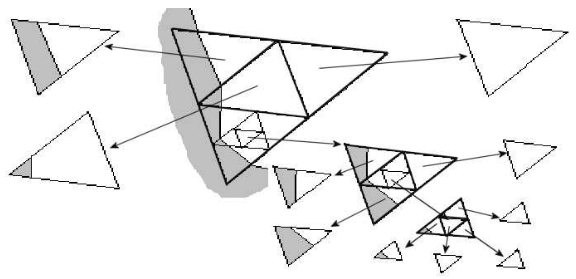
intersection of the perimeter of A and l. This is what we call a wedge. It takes on two different values $\widehat{c_a}$ above l and $\widehat{c_b}$ below, which are the respective mean values. A wedgelet is therefore denoted $w = \{v_1, v_2, \widehat{c_a}, \widehat{c_a}\}$.
We are now able to formulate the wedgelet decomposition. The wedgelet decomposition $W(I(A_{j,k}))$ is a collection of projection of $I(A_{j,k})$ onto a finite set of wedgelets. For each wedgelet $w(A)$ we select the one that minimizes (4.3) over $A_{j,k}$. The wedgelet decomposition $W(I(A_{j,k}))$ is $w = \{v_1, v_2, \widehat{c_a}, \widehat{c_a}\}$ that minimizes (4.3) over $A_{j,k} \in I$. The optimal multi scale decomposition $W^j(I)$ is the collection of $W(I(A_{j,k}))$ for all $A_{j,k}$ that minimizes (4.4).

$$W^j(I) = \{W(I(A_{j,k})) : j = 0..J, k_1..k_n = 0..2^j\} \tag{6.3}$$

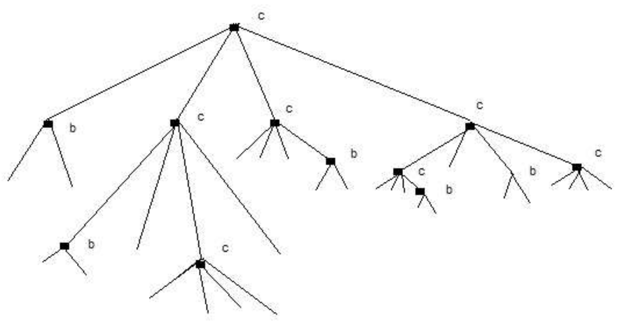
This is as can be seen almost the exact same formulation as in chapter 4, and the example shown in fig. 6.6(a) shows a great resemblance with 4.9

6.3 Results From a Wedgelet Decomposition

Having formulated this decomposition, we have to test the algorithm. There is one slight disadvantage with this method, we have to do an initial triangulation to start this algorithm. Given an image we simply draw a diagonal from one corner to the opposite, and thus dividing the image into two triangles. The wedgelet decomposition has been used in two setting. First an ordinary image which has been divided



(a)



(b)

Figure 6.6: (a) Example of a wedgelet decomposition based on triangles (b) The corresponding regression-tree.

as described above. Due to the practical use in the AAM of this wedgelet decomposition, it has also be tested on this geometry to show that it can be used in these settings. We have chosen a single image for starters which can be seen in Fig. 6.7 . Fig. 6.8(a) and 6.8(b) shows a normal image decomposed by the triangle base wedgelet decomposition at different compression ratios. As can be seen from the images it seems to work yielding results that corresponds to the underlying image. This can be observed at the edges which are reconstructed quite well. However there are some artifacts which have their origin in the implementation (numerical inaccuracy) and not the actual wedgelet decomposition algorithm. We now turn to the mean shape of the AAM or more correctly the AM. These geometry settings are a little bit different from normal images since they come with an initial triangulation. This is the reason for adopting the wedgelet formulation to triangles. However there is one small obstacle which we have to overcome. The root of the

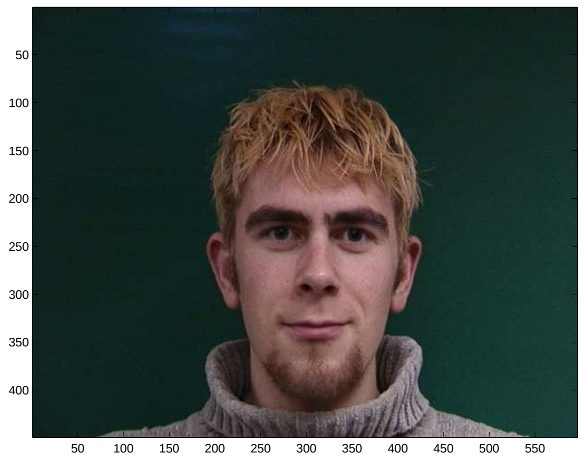
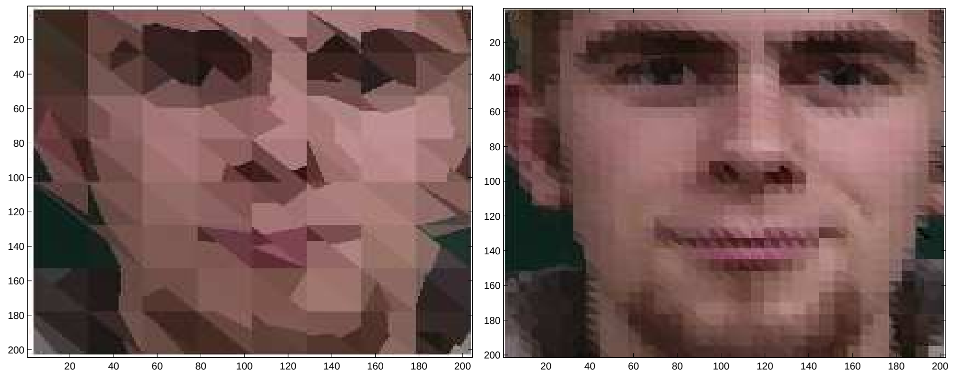


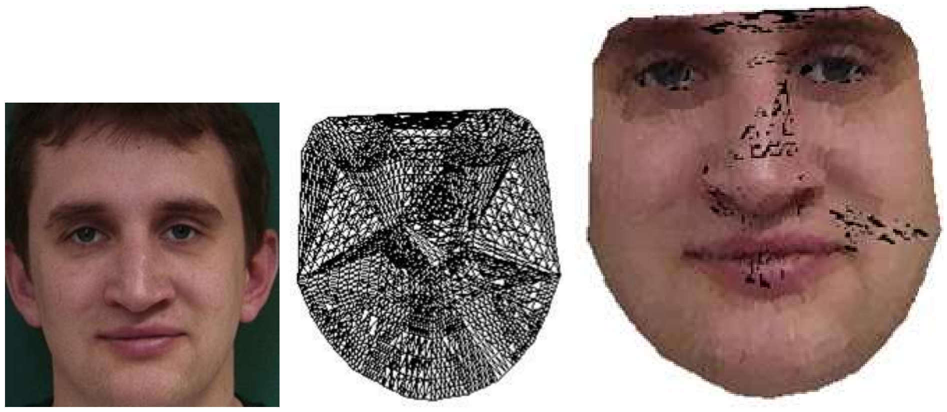
Figure 6.7: The original image.



(a) (b)

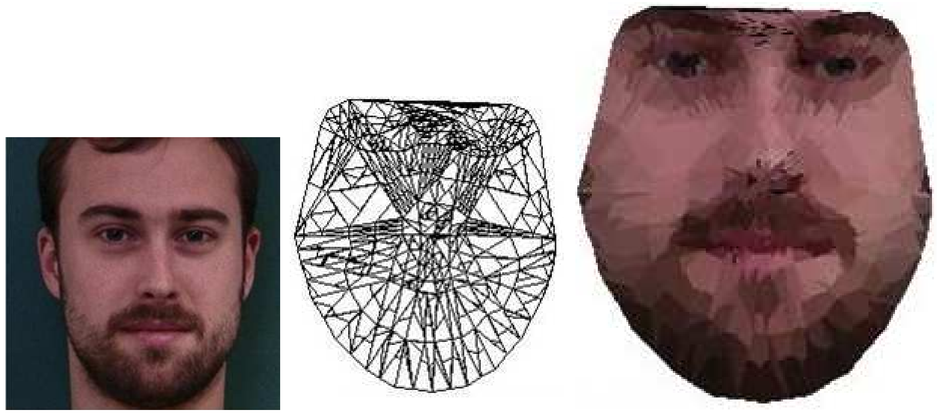
Figure 6.8: (a) A high compression ratio on a normal image (b) A high compression ratio on a normal image.

tree cannot be a normal node as the other nodes in the quad tree. We have to choose between either having lots of individual trees or a tree containing a lot of branches at the root and 4 at every other internal node. For obvious reasons we choose the last, since we then have the decomposition in a single tree. The initial triangulation sets a maximum limit to the compression ratios that can be achieved, but still this would be a rather hard compression. Fig. 6.3 and 6.10 show the skeleton of the wedgelet decomposition ,the super imposed skeleton and finally the result. This also produces the expected results, however, there are still some artifacts created by the implementation (numerical inaccuracy). Aside from these artifacts, the results



(a) (b) (c)

Figure 6.9: 2846 triangles and 1007 wedges



(a) (b) (c)

Figure 6.10: 494 triangles and 295 wedges

are very pleasing. To take the wavelets into the comparison here, it is obvious that the wavelets produce a result much more pleasing to the eye than the wedgelets but there is still more to come.

6.4 Wedgelets Enhanced Appearance Model

There are still some problems that need to be solved. Even though we have shown that wedgelets can be used in conjunction with basic shape of the AM, we still need to modify the decomposition to fit the demands imposed by the construction of the training set, the appearance model is build from. As explained in chapter 1 the AM consists of shape landmarks, which are selected by an expert and ideally should be exactly the same for each face. Then the texture is warped into a shape free space or a reference shape to sample the texture from the same space. The same constraints applies for the wedgelet decomposition. The shape issue is taken care of by the AM, however since we are transforming the texture the resulting texture or decomposition has to meet the demands the texture meets in the shape free space. The problem lies within the tree structure. If we compress each image individually, we would then end up with just as many different trees as there are images in the training set. Fig. 6.11 show the dissimilarity in the tree structure. It is not only the tree it self but also the end nodes that are very likely to be different since theses are not orthogonal splits but just affine and therefor have many possible configurations. If we just decomposed each image individually we would end up with images living in different feature spaces. It is actually possible to end up with similar tree structures but different leaves since it is actually these that make up the basis for the image. This dissimilarity results in a problem when we create the statistical model, the model need totally isomorphic trees as shown in Fig. 6.12 otherwise it is not possible to create a statistical model. Obviously this is due to the fact that the observations need to live in the same space, if they did not live in the same space then a model would not make any sense whatsoever. So this has to be solved or the wedgelet decomposition in conjunction with the AM becomes more or less useless and the wavelets would be a better decomposition since they produce a more pleasing result visually. There are several ways to ensure the

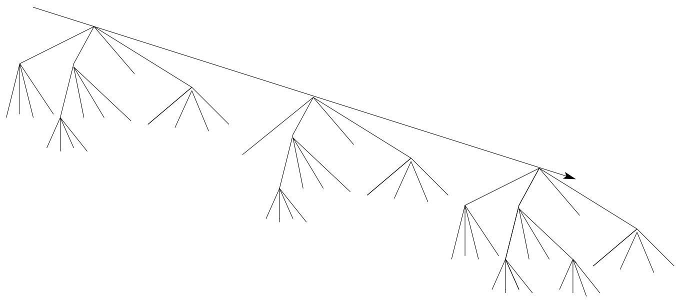


Figure 6.11: 3 nonisomrphic trees.

isomorphic sub trees. One way is to employ graph matching however the wedgelet decomposition almost gives us the instant answer to how to go about this. Since a very well known way of making parameter estimation in regression trees is cross

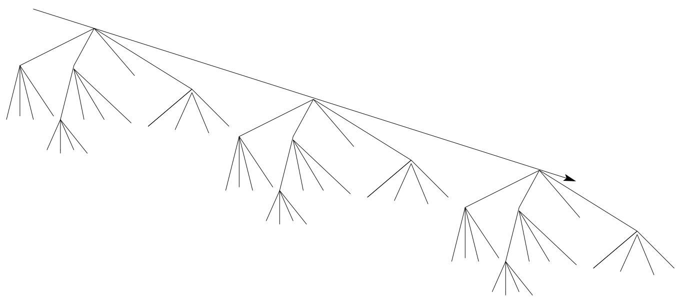


Figure 6.12: 3 isomorphic trees.

validation, why not use it in these settings. The easiest way to ensure that the trees in the decomposition, is simply to make a joint wedgelet decomposition of all the images at the same time. This leads to the use of cross validation as explained in section 2.2, to ensure that we get the best possible solution taking all the images into consideration.

6.5 Cross Validation in the Wedgelet Decomposition

Lets formulate how cross validation is used in the construction of the wedgelet tree. Let the regression model that predicts the pixel intensity at the ith image coordinate $\mathbf{x}_i$ in the kth image, y_{ki}, with a constant c_{km} in each region R_m be given by

$$\hat{f}_k(\mathbf{x}) = \sum_m c_{km} I\{\mathbf{x}_i \in R_m\} \tag{6.4}$$

For the sum of squared error loss criterion $\sum_i \|(y_{ki} - f_k(\mathbf{x}_i)\|^2$ we see that optimal $\hat{c}_{km}$ is just the average of y_{ki} in region R_m, $\hat{c}_{km} = \text{ave}(y_{ki}|\mathbf{x}_i \in R_m)$.The optimal partitioning is found by a bottom-up approach as explained in chapter 4. For each triangle at each level we seek the model that minimizes the K-fold cross-validation estimate of the prediction error across all affine splits/no split. Let $\kappa : \{1, \ldots, n\} \mapsto \{1, \ldots, K\}$ be an indexing function that indicates the partition to which training object (image) $k = 1, \ldots, n$ is allocated by randomization, and denote by $\hat{s}(-\kappa(k))$ the split estimated with the $\kappa(k)$'th part removed. $R_{a(s)}$ and $R_{b(s)}$ are the regions resulting from splitting a triangle by an affine split s, and R_c is the entire triangle. Furthermore, let the regression parameters from the kth image resulting from applying the split s be

$$\hat{c}_{km} = average(y_{ki}|\mathbf{x}_i \in R_m), \qquad m \in \{a(s), b(s), c\}$$

Then the cross validation errors become

$$\begin{aligned}\text{CVE}_{\text{split}} &= \sum_{k=1}^{n} \sum_{\mathbf{x}_i \in R_{a(\hat{s}(-\kappa(k)))}} \|y_i - \hat{c}_{k,a(\hat{s}(-\kappa(k)))}\|^2 \\ &+ \sum_{\mathbf{x}_i \in R_{b(\hat{s}(-\kappa(k)))}} \|y_i - \hat{c}_{k,b(\hat{s}(-\kappa(k)))}\|^2 \\ \text{CVE}_{\text{no split}} &= \sum_{k=1}^{n} \sum_{\mathbf{x}_i \in R_c} \|y_i - \hat{c}_{k,c}\|^2\end{aligned}$$

The crossvalidation is computational demanding, however there is a shortcut. Since the model is linear, that is across the regression trees i.e. a leaf can be written as **y=ax+b** the Generalized Cross Validation (GCV) which is an approximation to the leave one out cross validation should be able to improve the performance of the algorithm significantly. The GCV is written as

$$GCV = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - \hat{f}(x_i)}{1 - S_{i,i}}\right]^2 \tag{6.5}$$

where S is the hat-matrix in the regression.

6.5.1 Complexity Penalty

Next we focus on the complexity term. The entire algorithm is designed to be recursive, this means that we do not know the total number of leaves, however, we know the local area and we know the area of the root. We can also calculate the variance over the data set i.e. image, a normal factor to include in CPRSS, hence we get the following penalty for complexity

$$\begin{aligned}\#P &= \frac{A_{root}}{A_{triangle}} \qquad giving \\ CP &= const \cdot \sigma \cdot \#P\end{aligned} \tag{6.6}$$

We can now use the wedgelet decomposition on a training set for an appearance model. Fig. 6.13 show the algorithm at work and we are able to decompose the images using the same decomposition for all images. There are of cause other methods for ensuring isomorphic trees, as mentioned they come from the field of graph theory. Methods such as tree multiplication could be used, which would produce the largest common isomorphic subtree. But also a method that grow and collapses branches can be used to modify the trees so that they become isomorphic. As can be seen from fig. 6.13 the triangulation is exactly the same for all images, this means that their wedgelet trees are isomorphic. Further more if we move them into the same reference shape then the mean value within each subdivision would be the only way of distinguishing between the faces. We now have the desired building blocks for creating a appearance model. The wedgelets now becomes our observations where the degenerate are one observation an the nondegenerate are two, one for each value. Fig. C.10(a), C.10(b), C.7(c) and 6.14(d) shows the resulting wedgelet based AM. For further examples see app. C. So now we have shown that it is possible to use wedgelet decomposition for creating AM, and that the models are informative. What happens is that in the AM the shape model is unchanged but the texture vector which often reaches sizes of 10000+ and 30000+ for color images as here are reduced considerably. Hence the real reduction in the model comes from the compression of the texture which has actually been changed to a wedgelet vector. This reduction of transforming the texture to wedgelets is the actual optimization of the AAM as will be explained in the next chapter.

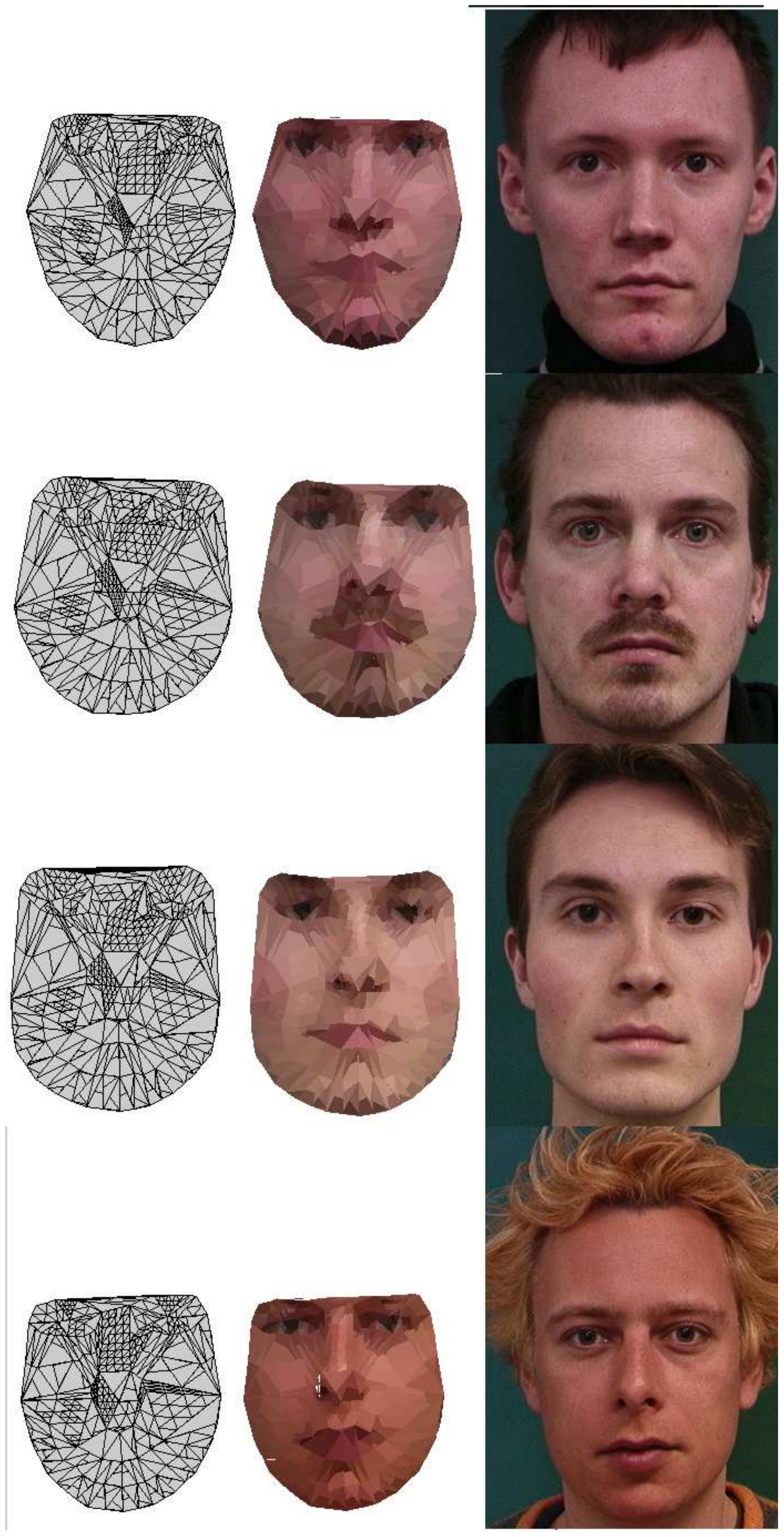


Figure 6.13: 4 Isomorphic faces decomposed at the same time

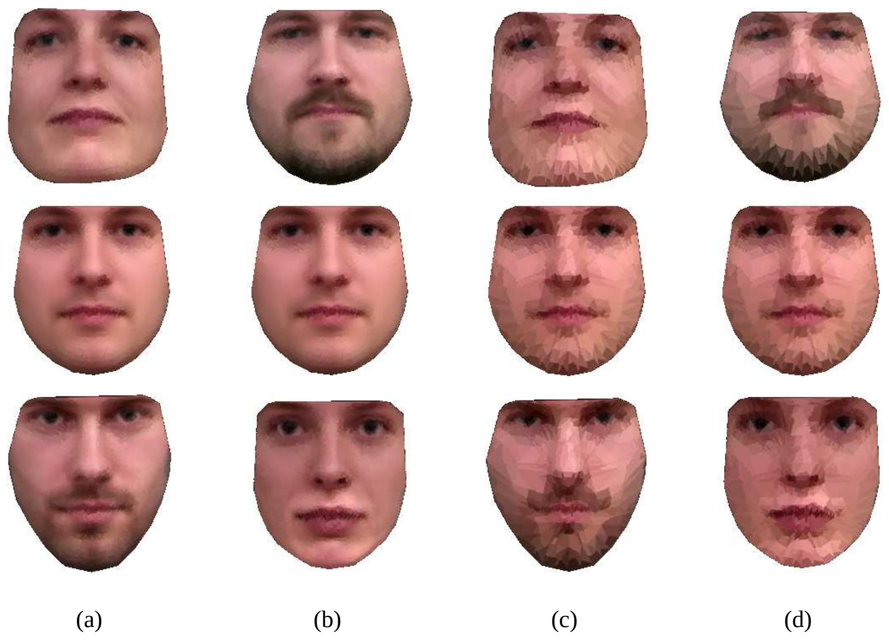


Figure 6.14: (a),(b),(c) and (d) shows the first two combined principal components +- 3 std. and the mean shape. (a) and (b) with ratio $1:3$ (c) and (d) with $1:40$.

Chapter 7

Results of the Wedgelet Enhanced AM

Having made all the building blocks for the wedgelet enhanced appearance model, it's time to view and discuss the results. First we mention that the work presented so far has been used to write an article submitted GBMV(Generative-Model Based Vision) in conjunction with CVPR, the article can be found in app. B.

7.1 Compression Ratios

We seek to improve the performance of the AAM through compression so let us briefly discuss this. In this discussion there are two types of compression ratios. There is the actual, as used normally as a measure in image compression i.e. the ratio, which is not as straight forward to determine for wedgelets since they do have some built in geometric property. Secondly there is the effective which depends on the context in which the compressed data is used, in this thesis for increasing the performance of the AAM. For normal image compression a wedgelet decomposition strictly rely on the roots position, but given this we can reconstruct the image. Still for each nondegenerate wedgelets we need to store two extra parameters, hence the two intersections of the splitting edge v_1, v_2 furthermore the two mean values c_a and c_b. Due to these facts the actual compression ratio is approximately

$$ratio = \frac{m_d + 4m_n}{n_i} \tag{7.1}$$

where n_i is the number of pixels in the original image I, m_d is the number of degenerate wedgelets and m_n is the number of nondegenerate wedgelets. The actual placement of each triangle or dyadic is given in the tree structure relative to the root. Therefor no parameters for this has to be stored within the decomposition tree. For the use of wedgelets in conjunction with the AAM there is a lot of considerations so the compression ratios are not directly comparable in the normal sense. The following section will elaborate on this.

7.2 Performance Improvements in the AAM

The idea in this thesis to improve the performance of the appearance model by reducing the amount of data in the calculations. The performance gain however is not something we have been able to measure directly due to the implementation task. In order to measure the exact improvement in performance, a totally new AAM would have to be implemented. However, we can give some understanding as to where the improvement lies. We have stated that the compression ratio is no directly comparable to normal ratios, and the true gain of using wedgelets does not come from the actual compression ratio, but rather the way that the wedgelets are represented. As we know from chapter 1 the AM consists of a texture part and of a shape part. This is where the optimization lies i.e. the performance gain. The shape is unchanged but the texture is replaced by wedgelets and since the texture is dependent on the shape some of the geometrical overhead in wedgelets can be ignored. The AM is now based on shape and wedgelets which is a much sparser representation of the image. How does this then reduce the computational complexity of the AAM?. The optimization lies within the core of the AAM i.e. the fitting described in chapter 1. If we go back to the fitting of the model on to a new image we know that first a shape is projected onto the image we want to sample from. Having projected the shape onto the image we are able to sample the texture into the reference shape. These two steps remains the same for the AAM, however, the next step where we project the texture on to the reference shape is change or improved. We recall (1.9)

$$\mathbf{t} = \hat{\mathbf{t}} + \mathbf{Q_t c}$$

which is the projection of the texture onto the reference shape. For each iteration this projection has to be made in order to estimate the changes in the parameters c. Lets say that the texture consists of 10000 samples and the sparse model of the AM consists of 23 parameters, then we have a projection matrix Q_t of 10000×23. But using a wedgelet decomposition with a pseudo compression ratio of 1:10 we have a texture vector or wedgelet vector of only a 1000×23 which makes the ratio count as saved calculations in each iteration of the appearance model regarding the texture. This is a huge improvement of the AAM since this enables us to use it on images with much higher resolution. Actually having the wedgelet in the AAM it would be possible to directly use (1.10) repeated here for convenience

$$\mathbf{r(p)} = \mathbf{t_s} - \mathbf{t_m}$$

where t_m is the model texture in this case the wedgelet value and t_s is the texture sampled from the image. This reduces the computational power needed significantly in the inner loop. There is however still some calculation that cannot be reduced. The sampling of the texture from the image after having projected the shape on to the image is hard to optimize since the original image is not wedgelet decomposed. Other methods has been used for applying compression to the AAM, Wolstenholme, Taylor [6] and Forchhammer and Stegmann [7] has both used wavelets for compressing the texture model with great success with respect to the fit. However, performance wise the wavelet decomposition has proven to slow the calculations considerably. The explanation for this lies in the basis shift that has to be performed to move the data from the pixel basis to the wavelet basis and vice versa. To compare pixel intensities as in (1.10) the basis has to be the same and since the model is build on wavelets, the data has to be transformed in order to make the comparison needed to calculate the perturbation for the next iteration. Wedgelets compared to wavelets are directly transferable to the pixel domain i.e. the operation needed for the basis shift is much simpler than the ones needed for the wavelets, hence with wedgelets we will get a faster algorithm. Having this in mind, it seems all though the wavelet decomposition offers some compression this reduction in information cannot be used in the AAM at present. For optimizing the AAM wrt. speed wedgelets would be an obvious choice.

7.3 Results of Fitting

We now turn to asses the quality of the performance achieved with the wedgelet enhanced appearance model. Using a AAM the goodness of the fit is more vital than the speed, so if the wedgelets enhanced AAM has an unacceptable goodness of fit this way of optimizing the AAM has to be abandoned. The experiments for evaluating the performance of the wedgelet based AAM has been carried out using N-fold cross-validation in the construction of both the wedgelet decomposition of the training set and the construction of the AAM. A total of 37×8 training sets has been decomposed and 37×9 AAMs has been created. The average landmark distance from model to ground truth is used to measure the performance. The models are initialized using a displacement of 10% of the width and the height from the optimal position in the x and y direction. First lets see the results of some of the fittings achieved by the wedgelet enhanced AAM. The two models shown in fig 6.14 has been used for fitting and the results can be seen in fig. C.12 and 7.2. Both of the models perform quite good and there is not a huge difference in the fit of the models even though they have a significant difference in compression ratios. The model with the lowest compression $(1:3)$ could be expected to have a better performance, this is also the case but it is quite difficult to see the difference with the naked eye. So To get a better idea of the performance lets take a look at the results for all 8 compression ratios shown in fig. 7.3. To left we have the original appearance model without compression and as can be seen from the plot there is a sleight increase in the inaccuracy along with the increase compression ratio. As could be expected none of the wedgelet based models out perform the original model. On the other hand the results are quite good and a deviation from the original model of approximately a pixel or less is considered as a very good result and hence lead us to state that wedgelet enhanced appearance model is success full. Experiments on compression of the texture using wavelets [7] has shown that the images can be compressed to a ratio of $1 : 44$ without significant reduction

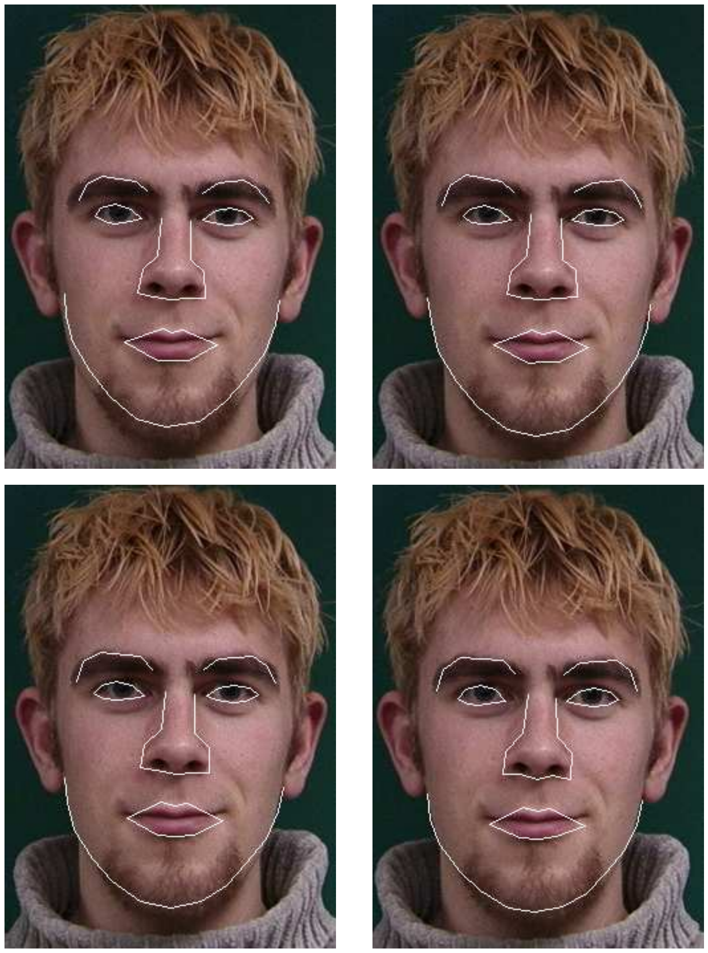


Figure 7.1: The fit of the AAM at $1:3$ ratio.

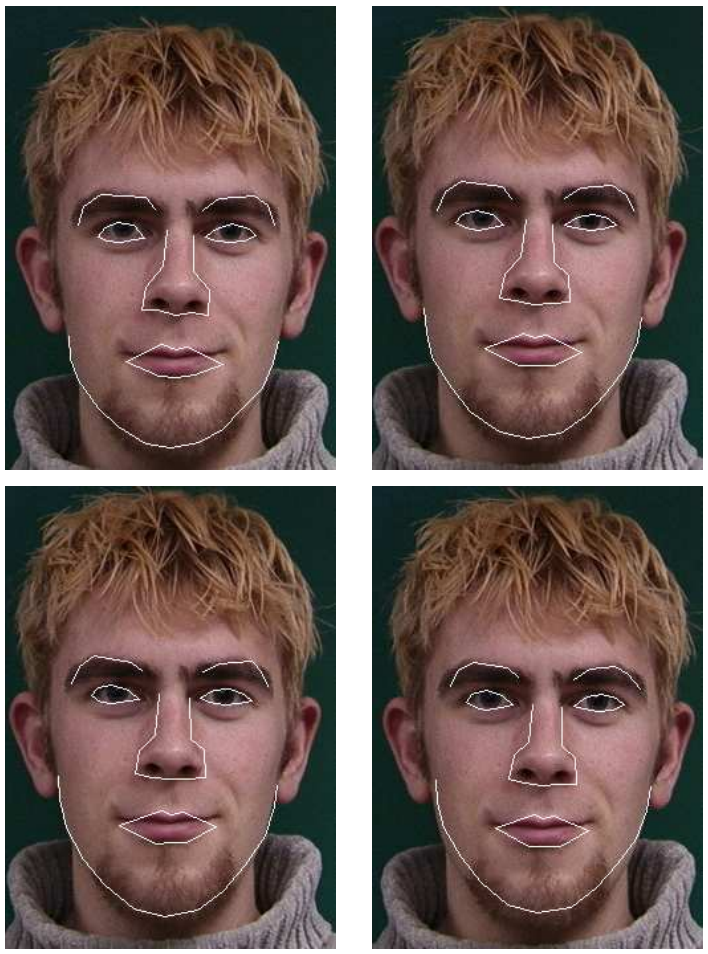


Figure 7.2: The fit of the AAM at $1 : 44$ ratio.

in segmentation performance. However, these experiments were carried out on gray scale images and therefore the results are not directly comparable with the experiments conducted in this thesis. The slight decrease in performance seen as the compression ratio increases is limited to approximately a pixel over the increase from $1:3$ to $1:44$ see Fig 7.3

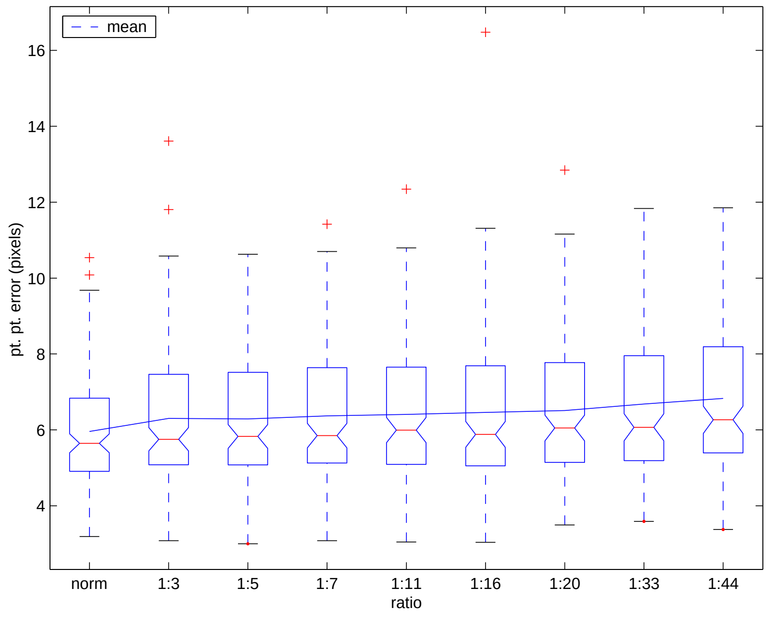


Figure 7.3: The average landmark distance from model to ground truth, a n-fold cross-validation test of the performance of the AAM.

7.3.1 Compression Limit

There is an upper limit for the compression ratio that can be achieved with a wedgelet decomposition in these settings, due to the fact that we have a reference shape consisting of 95 triangles. This ratio is approximately $1:300$, and the same goes for wavelet. To take the wedgelet model to the limit additional experiments has been conducted. Due to the excessive amount of calculations, i.e. a time consuming task, the results here are only based on 6 models Fig 7.4. Anyway they show with conviction that the model performs well with compression ratios up to 1:115 results, ratios unreachable with wavelets. The results in Fig 7.4 are leave-one out using 37 images, but only on 6 first images. Even though the figure shows that the performance increases at higher ratios, there is some uncertainty that makes the results for the ratio of $1:152$ questionable.

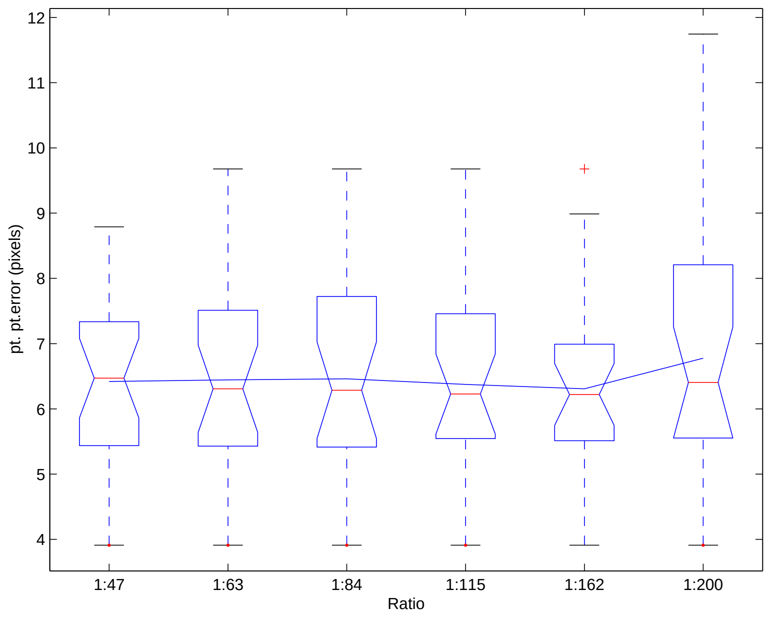


Figure 7.4: The average landmark distance from model to ground truth, a cross-validation test of the performance of the AAM at high ratio.

7.4 Implementation Issues

The algorithms used for doing this implementation has primarily been held in a recursive nature. This of course has some disadvantages but they are far exceeded by the benefits. The choice of recursive algorithms has obviously been inspired by the tree structure, since most algorithms for tree structures are recursive. All the formulations of the algorithms cam be found in appendix A. The choice of language has been Java, but a more appropriate language would have been c or c++ because of the computational demanding algorithms, however, there is a small benefit, the implementation runs on all platforms supporting Java. Some of the decomposition have taken up to 8 hours to complete on a high performance PC, so future implementations should be optimized. It is important to mention that these algorithms developed for the wedgelet decomposition are very easy to implement using parallel programming techniques. We would therefor advice people that wish to implement this to pay a lot of attention to optimizing of the calculations and execution structure of the code.

Chapter 8

Markov Random Fields

Markov Random Fields (MRF) is now introduced . The purpose of the introduction is to use MRF for regularization on the geometry of the wedgelet decomposition.

MRF has its origin in Markov chains described in the literature by the Russian mathematician A. A. Markov (1856-1922) who was one of the first to venture into this field. The subject has since then been studied and applied in various fields and is closely associated with Gibbs Random Fields (GRF). However MRF is merely the means to get to the goal. The problem that MRF is the solution to has its origin in Bayesian estimation theory, and therefore a good start is here with a short introduction.

8.1 Bayesian Estimation Theory

The basic idea is that we have some observation I say an image which is a glimpse of some underlying model W. Let us call this the world where I is some observation with occlusions and noise. A very good example is an image I of a forest W. Here the image taken is first of all discretized which gives a loss of information. Since the image is of a forest we know that the trees probably obscure the sight so that some other trees, animals etc. are occluded. The camera might add some noise to the image and we than end up with a observation which lacks some information. What we really want is to find the most likely estimate of the world W given the observation I called maximum a posterior. So to formalize this a little bit we write.

$$P(W|I) = \frac{P(W)P(I|W)}{P(I)} \tag{8.1}$$

This is the basically Bayes theorem for conditional probability, however, this is also the model we want to use. So what we are looking for is

$$argmax_W P(W|I) = argmax_W \frac{P(W)P(I|W)}{P(I)} \tag{8.2}$$

But since I is our observation $P(I)$ becomes a constant c then (8.1) yielding

$$\begin{aligned} cP(W|I) &= P(W)P(I|W) \\ &\text{Hence} \\ P(W|I) &\propto P(W)P(I|W) \end{aligned} \tag{8.3}$$

So to explain a little better what this model really expresses we say that $P(W|I)$ is the conditional probability for a world W given the image I. P(W) is what we call the prior probability, this tells us something about the world. In the framework here we primarily want to use this for regularization. And finally we have the likelihood which tells us how much our observation resembles a given world W. We are working with finite spaces so in theory it would be possible for a given image I to calculate the probability of all possible worlds, however in practice this is computational infeasible. For a simple classification task on a 50 by 50 pixel image with two classes to discriminate between it would be 2^{2500} possible configuration that would have to be estimated. However MRF proves very useful for solving this kind of problem as the following will show.

8.2 The Origin of Markov Random Fields

Going back to the origin of the MRF we start with the Markov chain. The Markov chain is Markov property + positivity (i.e. positive probability of all outcomes at all locations). The Markov chain states that you have the same amount of information looking at the previous state that you would have if you knew of all previous states. Therefore a process is called a Markov chain iff

$$\begin{aligned} P\{X(n) = k_n | X(n-1) = k_{n-1} \wedge \cdots \wedge X(0) = k_0\} \\ = P\{X(n) = k_n || X(n-1) = k_{n-1}\} \\ \text{for all } 1 \leq k_0, k_1, k_2 \ldots, k_n \leq m, \text{ all } n \in \mathbb{N}. \end{aligned} \tag{8.4}$$

So how does this extend to images

8.3 Cliques and Neighborhood

To understand MRF and GRF a bit better lets have a look at the neighborhood definition def. 8.1. From (8.4) it is obvious that the neighborhood can be stated the following way

Definition 9 *Let S=$\{s_0, s_1 \ldots, s_2\}$ be a set of sites. A neighborhood system N=$\{N_s, s \in S\}$ is a collection of subsets of S for which $s \notin N_s$ and $r \in N_s \Leftrightarrow s \in N_r$. N_s are the neighbors of s.*

In most cases when working with images we have a quadratic grid of pixels. This means that each pixel or site in the grid have 4 neighbors if they are not at the border of the image. Pixels have 3 neighbors if they are at the boarder but not in a corner of the image and the corner pixels have 2 neighbors. Other types of neighborhoods apart from square are triangular and hexagonal see Fig. 8.1. The neighborhood

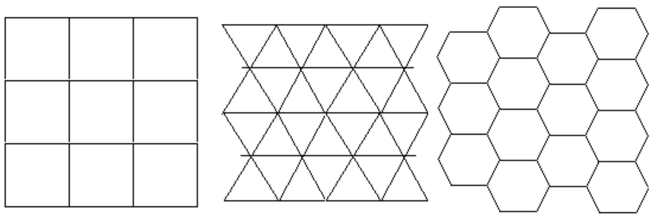
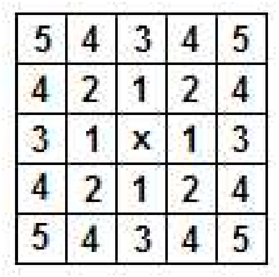


Figure 8.1: Neighborhoods: Square, triangular and hexagonal grid.

however is not limited to 2 dimensions and it is easy to imagine the voxel (pixel in 3d) version of the neighborhood. The neighborhood can include more than just the sites that have a common boarder with the site see Fig. 8.2. We talk about n-order neighborhood. Fig. 8.2 shows up till 5th order neighborhood, the lower-order are derived by excluding the sites with higher value than the desired order.



5	4	3	4	5
4	2	1	2	4
3	1	x	1	3
4	2	1	2	4
5	4	3	4	5

Figure 8.2: Up til 5th order neighborhood.

Cliques are have a slightly different definition (def. 10) and they are used to measure the sites similarity with its neighborhood.

Definition 10 *A Clique C is a subset of S for which every pair of sites are neighbors. Note that a single site is also viewed as cliques.*

If site i and j are neighbors we write $i \sim j$ The set of all possible configuration on S is called Ω.

8.4 Gibbs Random Fields

Gibbs used in 1901 the distribution in (8.5) to express the probability of a system being in a state with a certain energy. However he was not the first, Boltzmann came up with the similar distribution while investigating the probability of a

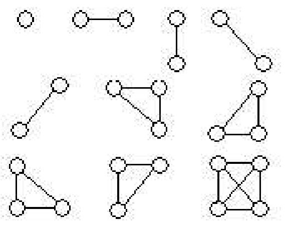


Figure 8.3: Up til second order cliques used on a rectangular grid.

molecule being in a state with a certain energy ϵ in 1877 (8.6).

let x denote a state in state space Ω and $U : \Omega \rightarrow R$ be the energy function

$$P(X = x) = \frac{1}{Z} e^{-\frac{1}{T}U(x)}$$
$$where$$
$$Z = \sum_{x \in \Omega} e^{-\frac{1}{T}U(x)} \tag{8.5}$$

$$P(\epsilon) = \frac{1}{Z} e^{-\frac{1}{Tk}\epsilon} \tag{8.6}$$

In both (8.5) and (8.6) the Z is called the partitioning function and is a normalization that makes the probabilities sum to one. The most famous application of GRF is the original Ising model. In 1925 Ising [29] used this distribution in a binary version to describe the properties of ferrite material as a magnetic dipole. This model has also been used in image analysis along with the natural extension to the multi label version namely the *Potts* [16] model which is well suited for improving Bayesian classification etc.

8.5 Markov Random Fields

Now we extend (8.4) to an n-dimensional grid, then to estimate the local value we only have to look at the local neighbors. This is a very nice property when dealing with images, and other scenarios with a huge amount of data. Now we don't have to take all data into account when estimating the probability of the local label or value we simply just include a certain limited neighborhood suiting for the task. So we define a Markov random field as follows.

Definition 11 *A random field* $\mathbf{X}$ *is a Markov random field with respect to the neighborhood system* $\mathbb{N}$=$\{N_s, s \in S\}$ *iff*

1. $P(\mathbf{X} = \mathbf{x}) > 0$ for all $x \in \Omega$

2. $P(X_s = x_s | X_r = x_r, r \neq s)$=$P(X_s = x_s | X_r = x_r, r \in N_s)$ for all $s \in S$ and $x \in \Omega$

Def. 11 and 8.4 are very alike. In (8.4) the neighborhood is obviously the previous observation and the cliques are also very simple. Basically a Markov Random Field is just a Markov chain with dimensions added to the neighborhood system. We now set $n = 2$ i.e. the number of dimensions to two because we do not need to go into higher dimensions in this thesis, however, Markov random fields can be applied any n-dimensional space where cliques and a neighborhood can be defined. As it often is with images in practice we encounter a border problem as with the Fourier transform. In this case the neighborhood size decreases at the boundary of the image so depending on the application of the Markov Random Field a solution to this problem has to be found. Most often the missing neighbors are ignored adding more weight to the remaining or they are simply set to have neutral influence hence yielding the same result. Now to show the connection between MRF and GRF we introduce the Hammersley-Clifford theorem 1

Theorem 1 *(Hammersley-Clifford)*
F is an MRF on S with Respect to N iff F is a GRF on S with respect to N

A lot of proofs of Theorem 1 exists and one of them can be found in [13] p.14.

8.6 The Energy Function

Having established the correspondence between MRF and GRF we take brief a look on the energy function. This is the function that measures the energy at a given site. The energy function has to be constructed specially to fit the given scenario e.g. the Ising model (8.8) basically measures 1 or 0, where α and the β's are weights that determines the influence of the individual cliques. This makes it possible to control the influence of the neighboring sites proportional to the influence of the label it already has. A natural extension to the Ising model is the Potts model. This is as mentioned a multi label version of the Ising model frequently used for classification in conjunction with a Bayes classifier. The reason is that when using a Bayes classifier the image and boarders between two groups tend to be a little noisy. The Potts model offers some smoothing where the neighborhood is taken into account and thus produces more homogeneous classification results.

$$U(x) = -\alpha \sum_i x_i - \beta_1 \sum_{i \leftrightarrow j} x_i x_j - \beta_2 \sum_{i \updownarrow j} x_i x_j \tag{8.7}$$

yielding the following probability for a given configuration

$$P(x = X) = \frac{1}{Z} e^{-\alpha \sum_i x_i - \beta_1 \sum_{i \leftrightarrow j} x_i x_j - \beta_2 \sum_{i \updownarrow j} x_i x_j} \tag{8.8}$$

The energy function has to reflect the properties of the world you are modeling i.e. a posteriori probability as the Ising model does. If it is edge enhancing perhaps some step function has to be included to trigger on large edges i.e. steep texture changes. This influence on the neighboring sites causes the MRF-adjustments to become iterative, we therefor need to update the sites in some order preferably independent of the neighboring sites. A way of selecting a local configuration among all local possibilities must also be found, both will be explained in the following section.

8.7 Update Scheme

There are numerous update schemes i.e. selecting the site to update and how to do this, however, to stay within the scope of this thesis we only describe the *Gibbs sampler* or *the heat bath algorithm* and random selection or the ICM. For further update schemes the reader i referred to [15]and [13] where this and other schemes are described. Given a finite MRF we want to minimize the energy to find the optimal global solution.

8.7.1 Gibbs Sampler

We start with Gibbs sampler which in general goes as follows.

1. Start with configuration x
2. Choose a site s
3. Replace x_s by a value sampled from the conditional distribution of X_s given the values of the neighborhood of s
4. If not stop the goto 2

Assuming that we have selected a site to update we calculate the possibility for each local configuration in correspondence with configuration. Having assigned a probability to a given configuration there are two obvious ways of selecting the local configuration. The first method is to choose the configuration with the highest probability in every sweep. This method is known as the ICM (Iterated Conditional Modes) and is a greedy algorithm that maximizes the local conditional probabilities. The other is the Gibbs sampler described above which is basically a randomization weighted by the probability for the given configuration. These schemes gives a basic way of updating the image so the only thing left is to decide a way to visit all sites.

8.7.2 Selecting the Visitation Scheme

We suggest two different visitation schemes. First a random visitation scheme where we chose the site randomly that is simply assign equal probability for next visit being a given site and then rolling a dice. This, however, is quite slow and is computational impractical. To optimize the speed and even enable parallel processing a checkerboard visitation scheme is proposed for rectangular grids. This means that the grids can be updated in two sweeps, first the black and then the white. This makes sure that all sites are visited after two sweeps and that no neighboring sites are visited in the same sweep.

8.7.3 Simulated Annealing

From the Gibbs distribution we still need to discuss the temperature, which can be used to optimize the solution. Originally the temperature was used in simulations of the motion of molecules in gases. From chemistry and thermo dynamics we know that molecules moves faster and due to their impact with other molecules, they move more randomly also with higher temperature. By increasing the temperature the configuration of the molecules becomes more random. This causes the probability of a given all possible configuration to become almost equal for infinite high temperature. Having a system where f is any configuration in the configuration space F then the configuration f will have the following probability

$$P(f)_T = [P(f)]^{\frac{1}{T}} \qquad T > 0 \tag{8.9}$$

where T is the temperature. From this it is easily seen that

$$\begin{gathered}\text{for } T \rightarrow \infty, P(f)_t \text{ goes towards a uniform distribution} \\ \text{for } T \rightarrow 0, P(f)_t \text{ will concentrate around peaks}\end{gathered} \tag{8.10}$$

This is exactly what we want to achieve by simulated annealing namely that all configurations initially have the same or almost the same probability and then as we cool down some states are emphasized, hereby hopefully helping the algorithm avoiding local minima. Fig. 8.4 shows the energy of a system at two different temperatures, T_1 and T_2,where T_1 is high temperature and T_2 is low. As can be seen from the figure it is easier to move towards the global minima at the high temperature than at the low temperature. After establishing what the temperature do, and seeing on how (8.5) relates to the distribution, a way of changing the temperature must be devised called a cooling scheme. The literature suggest several solution but the cooling scheme often has to be devised for the given task, however, one frequently used is which can be seen in (8.11) from [13] for other schemes the reader is referred to [13]. It should be noted that the simulated annealing only makes sense if used with the Gibbs sampler. The temperature will have no effect if it is used with the ICM.

$$T = \frac{C}{log(t+1)} \tag{8.11}$$

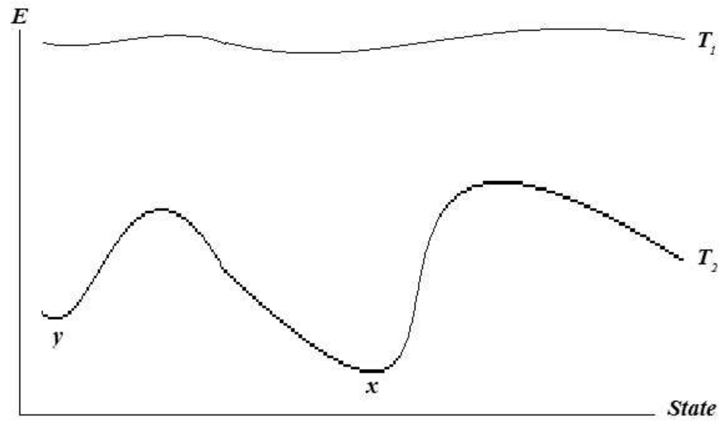


Figure 8.4: Simulated annealing: This shows the energy at two different temperatures, hence giving a more equal probability of the possible states

Chapter 9

Regularization of the Geometry

The previous chapter introduced MRF. This chapter will explain how they have been implemented on the wedgelet decomposition i.e. a regression tree.

As mentioned earlier wedgelets are basically edge detectors. This poses some problems in the wedgelet decomposition because edges are continuous even across boundaries. The basic wedgelet decomposition do not offer any solution to this. However, Donoho suggest in [9] chaining to keep the continuous property and a Markov model is suggested in [27]. The problem is that we have some observation and a model of how the wedges are placed at the leaves, but from the wedgelet decomposition we know that this has been calculated without consideration of the neighbors to the individual wedgelets. We therefor end up with the situation in fig. 9.1(a) but as we know the edges are not always local but often global. What we are rally seeking is a result like the solution in Fig. 9.1(b). This property can be

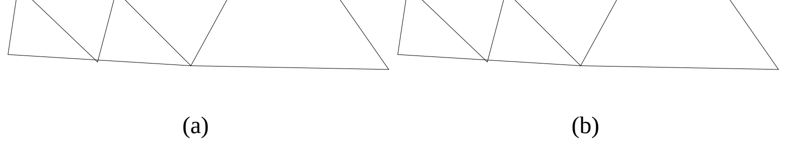


Figure 9.1: (a) The wedgelet decomposition normally (b)The desired Result of the final decomposition.

ensured during the decomposition as in [27] or as a post processing of the wedgelet decomposition. Focusing on a post processing, the problem at hand fits perfectly into the bayesian estimation theory, i.e. we have some observations of some images on which we want to impose a model. However not just any model, the most likely model considered all possible configurations. Having chapter 8 in memory it is obvious that a globally satisfying solution can be achieved through the use of MRF, a method for solving this type of problems. This however leaves us with some tough problems at hand. We have a recursive tree-structure due to the implementation of the wedgelet decomposition. We have a scale issue, hence moving away from the root through the tree structure is viewed as steps through scale space, which then leaves us with the problem of neighbors that exist on different levels in the scale space. Furthermore because of the tree structure neighbors do not have any knowledge of each others existence. So the first problem at hand is to come up with a recursive neighborhood solution in a multi scale tree-structure. The next problem is to figure out an update procedure which has to be designed this multi-scale tree structure, where a highly irregular neighborhood exist. Finally an energy function has to be designed to suit the purpose of connecting edges across boundaries in order to make the model work. We start with the first problem at hand, defining the neighborhood.

9.1 Neighborhood and Cliques

Though we are working with triangles and though a different definition of neighborhood already relationships exists, it is not as straight forward as could be expected. Normally a triangle has three neighbors as shown in Fig. 9.2 where the dark gray triangles t_n are the neighbors of t_s, however, this is almost never case when dealing with the result of a wedgelet decomposition. The wedgelets are as described stored in a tree at multiple scales which might or might not be the same among neighbors, therefore we somehow need to keep track of the neighborhood through scale space. Fig. 9.3 shows a common scenario where there are four levels of detail or four scales. This figure clearly illustrates the problems involved in implementing the MRF in a tree-structure so the first step is to find a way to handle the neighborhood through scale space. The scale space makes the most unpre-

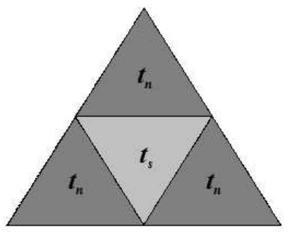


Figure 9.2: The neighborhood of t_s is the 3 triangle t_n.

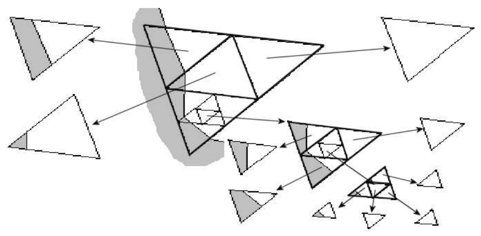


Figure 9.3: This wedgelet decomposition shows how the number of neighbors can vary

dictable property of the wedgelet decomposition the number of neighbors for each wedgelet. As Fig. 9.3 shows, the number of neighbors vary with the level of detail of the neighboring wedgelets, hence the number of neighbors can in theory vary from zero (at scale zero or the root) to infinity. In practice however, there is often only a few levels of difference between the scale of neighboring wedgelets and therefore seldom more than eight neighbors to a given wedgelet. This rather advanced neighborhood has an influence on the choice of cliques. Cliques are defined from their neighborhood, but our neighborhood varies so this has to be taken into account. First of all it's very difficult to define a direction on the neighborhood, if one has to be imposed it would be orientation from a global reference point. This is however not especially important in our case as in [15] where the grid orientation is a very important part of the MRF. We are concerned with edges so we want to define the cliques to use, but without an orientation. The neighborhood makes it difficult to make higher order cliques and it makes no sense with the purpose in mind to define a larger neighborhood so we restrict ourselves to first order. This leaves us with the cliques shown in Fig. 9.4. By adding the same weight to each of the last three cliques we achieve rotational invariance. However we cannot say

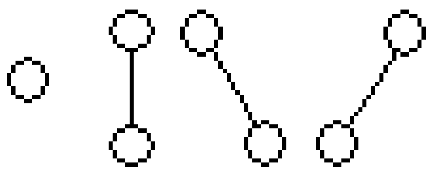


Figure 9.4: The 1st. order cliques for a regular triangle grid.

that we only and up with these configurations in the first order neighborhood. We may end up with a lot of cliques, but they will all be of the form in Fig. 9.4 with simple rotation imposed.

Having discussed the cliques and decided on the order to use in our model we move on. Since the wedgelets have no knowledge of it's neighbors we have to find a way to pass that information to it. The parent of a wedgelet, given the wedgelet is not the root, knows of at least one of the wedgelets neighbors since these are siblings and hence children of the same parent. For the center wedgelet this is sufficient and hence it can get all the needed information this way. But for the children at the vertices of the parent this is far from sufficient information. If we do this recursively then the parent of the parent might hold that information otherwise continuing recursively until the root will eventually give the needed information about the neighborhood. The solution derived from this can be formulated as a recursive algorithm see app. A. Fig. 9.5 illustrates how the information is held. The gray triangle need to know it's neighbors, it's parent hold information about two of its neighbors. However to obtain the information about the last neighbor, the parent of the parent has to be involved.

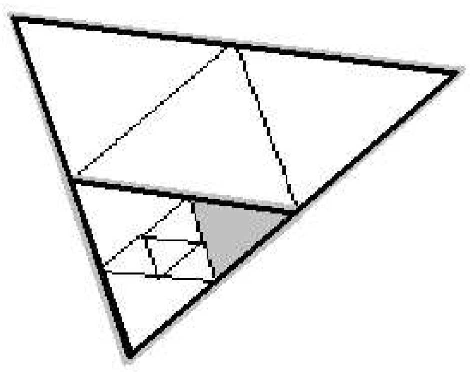


Figure 9.5: Information about neighborhood.

9.2 Boundaries and Penalties

Since we are only interested in optimizing the edges across boundaries we only need to pass the point of intersection in a nondegenerate wedgelet, to the neighbors and not the entire wedgelet. We need to somehow favor a configuration with connecting edges and penalize configurations with discontinuities. We want to emphasize this property, but we don't want to force connectivity to edges that do not exist so this also has to be balanced somehow. There has been written a lot of material about edge-enhancing MRF [13][14], and a lot of effort has been put into clarifying the properties of edges. First of all edges are continuous, but the wedgelet approach makes the boarders piecewise linear and hereby discontinuous already in the 1'st order derivative. What really is important in the configuration is the point of intersection of the border and the edge in the wedgelet hence we approach an espalier construction. It is these intersections we want to match in the neighboring wedgelets, and hence make the edges connect across the boundary. The boarder of a wedgelet is basically continuous and stretches across a finite number of pixels.

Given a discrete image as here, the steps is chosen to be approximately the size of a pixel since this is the resolution of the image. The penalty for misalignment for edges across the boundary between two wedgelets should be regarded as continuous, but will in reality be discretized by the chosen step size. Even though we in theory could make the borders continuous, it would quickly become computational infeasible since the number of configurations would increase exponential with the number of steps along each side.
To find a soothing energy function $U(x)$ we look at the wedges which we want to modify. Since we want to emphasize continuous edges across boundaries, the penalty should be a function of the distance between the intersections hereby favoring continuity. We also want to include the original CPRSS (4.4) in the model since it is this penalty the decomposition is built from. Several loss functions can be chosen all with different properties, and having the Potts model in mind we can make a similar energy function. We start with the discontinuity penalty hence we introduce a delta function that penalizes discontinuity across the boundaries.

$$U(x)_d = |a - x| + \delta(|a - x|) \cdot p,$$

where x is an internal intersection point, a is an external intersection point and p a number between 0 and 1

$$\delta(|a-x|) = \begin{cases} \text{if } (|a-x| = 0) & 0 \\ else & 1 \end{cases} \tag{9.1}$$

Calculation of $|a - x|$ is done using homogeneous barycentric coordinates which makes $0 \leq U(x)_d \leq 1 + p$. Fig. 9.6 shows a boarder between two wedgelets with misalignment, and how the misalignment penalty is calculated. This has somehow

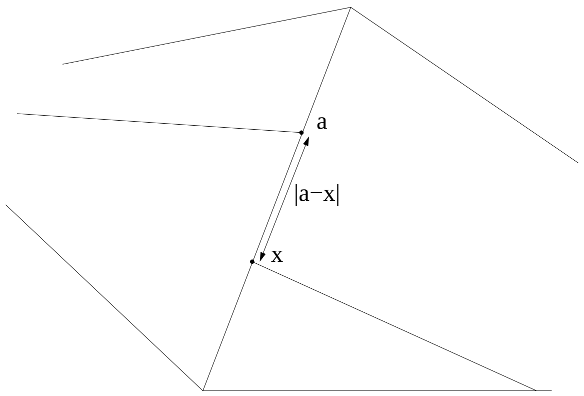


Figure 9.6: The penalty function.

to be weighted against the measure used in the wedgelet decomposition and we therefore take a look to the Potts or Ising model. We include the same penalty used in the wedgelet decomposition formed by the texture using cross validation and to save computational power we store the results during the wedgelet decomposition. This will of cause take up a lot of memory but will save the computational power since the CPRSS is constant for each configuration. The PRSS (4.3) is added to the energy function with a weight α hence the analogy to the Potts or Ising model. The penalty for misalignment can then be adjusted by the parameter β also similar to the Potts model. This yield a candidate for the energy function which is given in the following. The CPRSS (4.4) is in the following denoted $U(x)_p$ and the displacement penalty is denoted $U(x)_d$.

$$U(x) = \alpha U(x)_p + \beta U(x)_d w \tag{9.2}$$

where α and β are some constants chosen appropriate and w is some weight i.e. the actual area of the triangle. where α, β_1 and β_2 are some constants chosen appropriate. This will add some weight to the energy and hence emphasize continuities and disfavor discontinuities.

9.2.1 Simulated Annealing

To minimize the possibility of ending up in a local minima we employ simulated annealing. The cooling scheme used is (9.3). From this we derive the following probability function adapting to a Gibbs distribution.

$$T = \frac{C}{log(t+1)} \tag{9.3}$$

$$P(X = x) = \frac{1}{Z} e^{-\frac{1}{T}(\alpha U(x)_p + \beta U(x)_d)} \tag{9.4}$$

9.3 Visitation Scheme

We now have to define a visitation scheme. For the visitation scheme as with the neighborhood it is not as simple as it might look due to the tree structure and the recursive nature of the implementation. As explained in sec. 9.1 the number of neighbors is not predicable, hence a scheme that works on such a scenario is must be devised. We want to use a visitations scheme that is as similar as possible to the checkerboard approach described in chapter 8. This mean that a sweep method with multiple sweeps making sure all are updated more or less independently must be constructed. Starting with a simple setup as shown in fig. 9.2, it's straight forward to update as intended. In first sweep the center t_s and second sweep the outer ones t_n. This seems to work for this small grid, but it is not straight forward for larger grids as Fig. 9.7 left, here a four step procedure must be employed as fig. 9.7 shows to ensure a proper update scheme. If we apply the two-step procedure recursively, however, as shown if Fig 9.8 left, the results become much more like the checkerboard. Compared to the update scheme to the right, which is the same as in Fig. 9.7, there is in practice so small a difference that it can be ignored. It

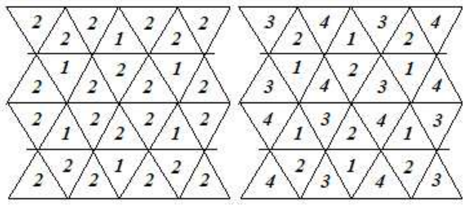


Figure 9.7: two different visitation schemes, one with two sweeps and one with four.

is also worth having in mind that this done using parallel processing hence every update sweep is actually ordered, this just adds to the confidence in the selected update scheme. We have now devised a update procedure that fits our scenario, or at least the core-update within the tree and the pseudo code can be found in app. A. However, we still have to deal with the initial triangulation of the mean shape Fig. 1.2 which also displays an irregular behavior concerning the neighborhood but in a different way. Even though each triangle at most has three triangles as neighbors, they are not of the same size and do not have the same angles. This results in some issues that has to dealt with i.e. two neighbors might share a neighbor etc. A simple look at the mean shape just emphasizes the issues that needs to be dealt with in comparison to the regular grids in Fig. 8.1. The problem however, is only present at the root of the tree and not within the tree. Regarding each triangle as the

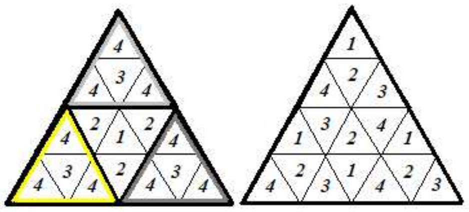


Figure 9.8: Two different visitationschemes for triangles.

root of a wedgelet decomposition tree, and having the recursive update algorithm just presented in mind, we propose a very simple update algorithm with only two sweeps. This will then be the update scheme for the wedgelet decomposition tree for the AM. The scheme consist of two vectors S_1 and S_2, S_1 containing all the trees to be updated in the first sweep and S_2 all the trees to be updated in the second sweep. All we need now is to populate the two vectors. Letting W denote all wedgelet-trees, p_1 and p_2 denoting two pointers each pointing at the $0'th$ element of S_1 and S_2 respectively, the vectors are populated in the following way.

1. Put all trees $w_t \in W$ in to the vector S.
2. Remove the first element in S and insert this into S_1

3. move the pointer p_1 to the next element in S_1
4. Trying with all elements in S, remove from S and inset into S_2 all neighbors to the element that p_1 points to.
5. Move the pointer p_2 to the next element in S_2
6. Trying with all elements in S, remove from S and inset into S_1 all neighbors to the element that p_2 points to.
7. Repeat from 3 until S is empty

Fig. 9.9 shows the result of this algorithm. As can be seen there are some neighbors that are updated in the same sweep but due to the recursive update of the trees and the fact that its the children that will be updated, this effect will be canceled out.

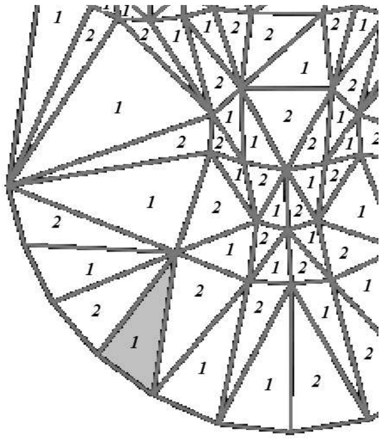


Figure 9.9: The visitation scheme at the root of the wedgelet decomposition.

9.3.1 Update Scheme

The update schemes presented in chapter 8 can used directly without modifications. The experiments have therefor be conducted using the Gibbs sampler and the ICM.

Chapter 10

Results of the MRF Implementation

We have implemented the MRF on wedgelets to help restore some geometric properties i.e. connection edges so our result must primarily be evaluated visually. This mean that we prefer is something that look a bit like geodesic curves corresponding to steep changes in the underlying texture. We assume that the initial configuration of the wedges is not far from the desired solution since we incorporate some of the measures used in the wedgelet decomposition. To verify this we start by taking a look at one of the decomposed images (Fig. 10) to see how the initial configuration of the edges look. From Fig. 10 it is obvious that the edges connect some places and don't at others. Since we are building our model from multiple images using cross validation this comes as no surprise due to the fact that the images is not the same and we therefor expect the edges in the individual images to be different. This of cause have the effect that there is no underlying edge model some places and therefor nothing to connect. However since the images have been annotated identically we expect some of the edges to match across the collection of images so it makes good sense to enforce the connecting edges property most places. This is, as explained, a balance that has to be maintained. If the edges are not connecting in the underlying model we will not enforce such a property, but if they are connecting in the underlying image we want the edges to connect. The way this balance in maintained is through experiments to determine the parameters α and β from (9.4) repeated here for convenience.

$$P(X = x) = \frac{1}{Z} e^{-\frac{1}{T}(\alpha U(x)_p + \beta U(x)_d)} \tag{10.1}$$

An obvious measure of the desired solution is the energy which we seek to minimize i.e. maximizing the probability. When seeking the optimal solution given our energy function we should see a decrease in global energy as we approach the optimal solution. Let us first take a look at some of the results. Fig. 10.2(a) is a plot of the energy against the number of iterations using the gibbs sampler. For assuring that we are in a local minima we finish off with the ICM for a couple of

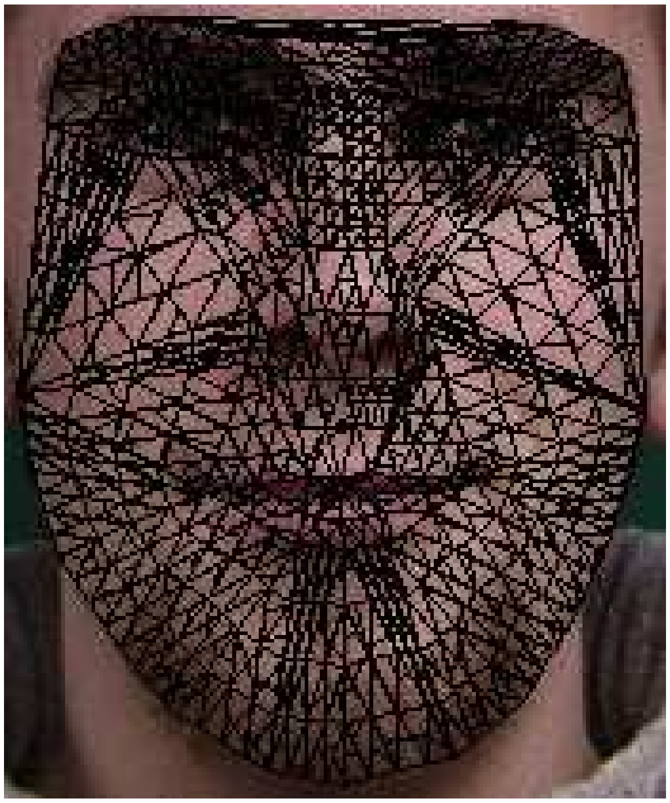
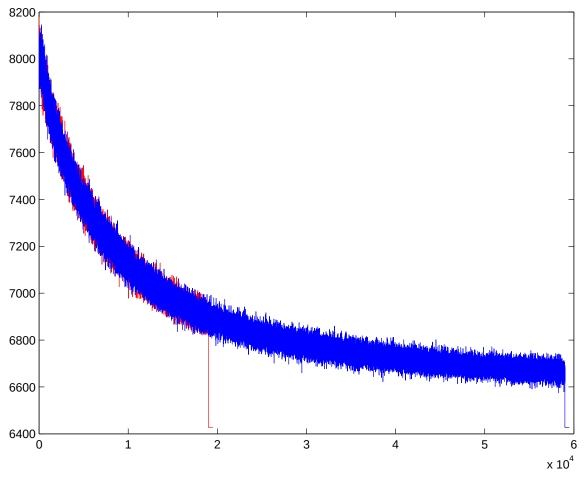
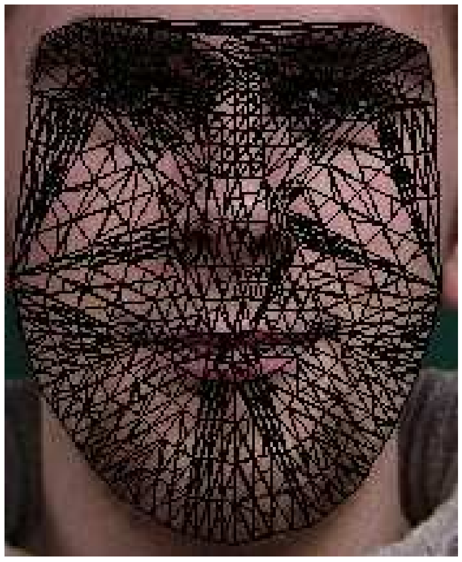


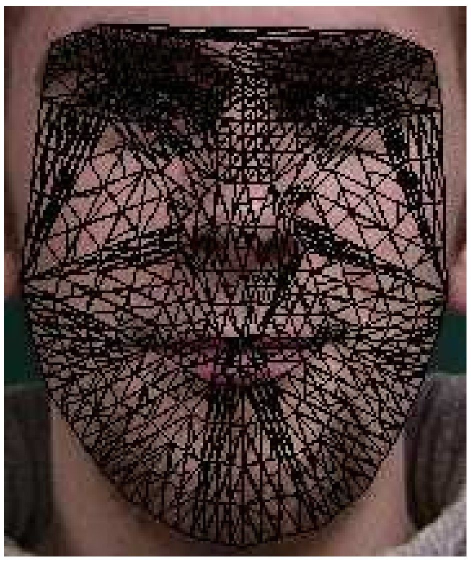
Figure 10.1: The original wedgelet decomposition without geometric constraints

iteration which removes flicker in the energy. As can be seen from the energy we end up almost exactly at the same energy level whether we use 19000 or 59000 iterations. Fig. 10.5(a) and 10.5(b) shows that both the ICM and the Gibbs sampler works and end up with almost the same solution. There are two plausible reasons for this. First, it could be that the initial solution is very close to the optimal solution and therefore we end up in the same minima no matter which scheme we use. Otherwise we are stuck in a local minima which we cannot escape from. However the experiments has shown that even if we start at a very high temperature 10^6 the solution still remains the same and we therefor discard the theory of being stuck a local minima. What is left is a single global minima which makes the ICM solution sufficient since the underlying function seems to be nice and smooth. As Fig. 10.5(a) and 10.5(b) clearly show the intended geometric property is enforced, hence the edges are connected. The balance between the connection of the edges and the underlying image seems to be maintained in both images. To find out if the ICM find the global minima several images are regularized using the ICM. The results are shown in Fig. 10.3 As can be seen the results show that the ICM is the obvious choice of update scheme when using MRF for regularizing the wedgelet



(a)





(b) (c)

Figure 10.2: (a) The Energy as a function of the number iterations (b) The result after 19000 iterations (c) The result after 59000 iterations.

decomposition and is therefore the update scheme used in the remaining experiments.

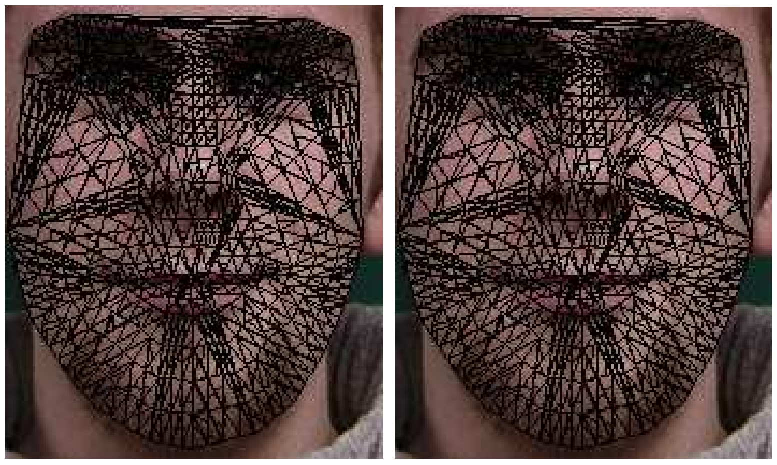


Figure 10.3: (a) Gibbs sampler and ICM to finish with (b) ICM.

10.0.2 The Energy Function

It remains to summarize the parameters used in the energy function. As described in chapter 9 the energy function used consists of a δ-function for misalignment and some displacement function based on the distance between misaligned intersections. We repeat (9.2) here for convenience

$$U(x)_d = |a - x| + \delta(|a - x|) \cdot p,$$

where x is an internal intersection point, a is an external intersection point and p a number between 0 and 1

$$\delta(|a - x|) = \begin{cases} \text{if } (|a - x| = 0) & 0 \\ \quad else & 1 \end{cases} \tag{10.2}$$

given the following energy function used in these experiments

$$P(x = X) = \frac{1}{Z} = e^{-(\alpha CPRSS + \beta_1 |a-x| + \beta_2 \delta(|a-x|))} \tag{10.3}$$

where α is set to 1, β_1 between 0.01 and 100 mostly 5 and β_2 fixed at 0.2.

10.1 Segmentation Improvement of the wedgelet enhanced AM

It seems that for higher compression ratios the MRF modified wedgelet enhanced AM performs better. The reason could be that the general contour in the underlying model enhances the performance of the wedgelet enhanced appearance model. The results here are based on a very small test set due to lack of time to do a thorough test. The indications however are quite clear since all values of the fit with the MRF modified model are less or equal in pt. to pt distance compared to their non-modified counter part. As could be expected the influence of the MRF regularization decreases with compression ratio. Fig. 10.4 shows the results achieved by the MRF modified wedgelet enhanced appearance model. Forchhammer and Stegmann [7] has achvied results with wavelets that outperform the unmodified AAM on the segmetation accuracy so this improvement comes as no supprise. Another clear indication that there is an underlying image is that even though the

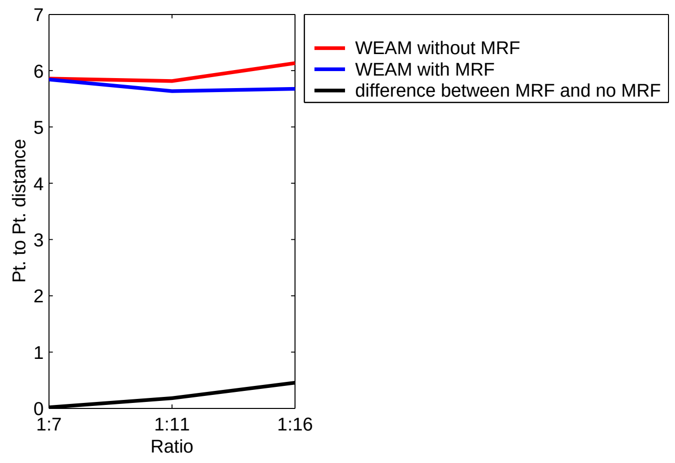


Figure 10.4: The result of the comparison between wedgelet enhanced AM with and without MRF.

models are built from slightly different set of images the model is the same as Fig. 10.5 shows. This also supports the thought of the global minima achieved by the ICM.

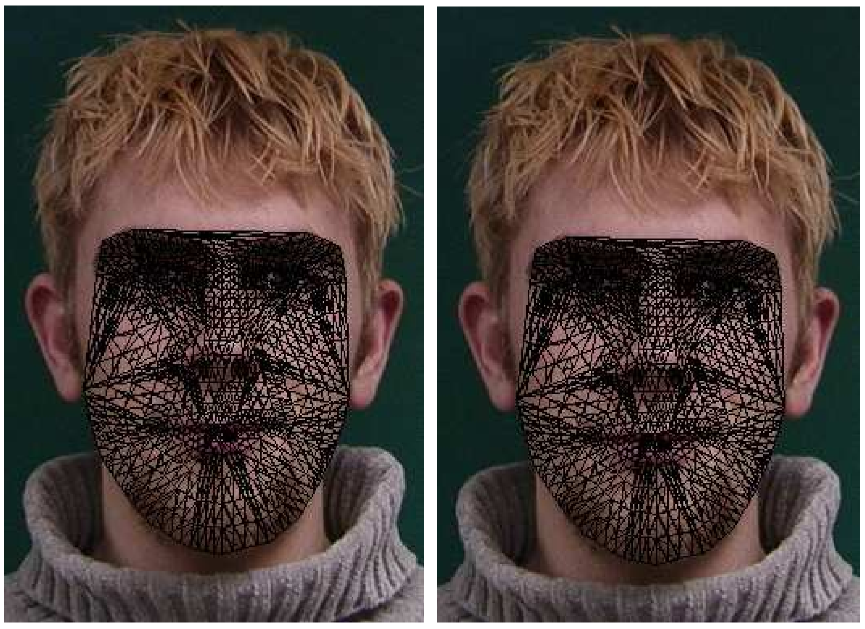


Figure 10.5: (a) Without image 16, 1:11 ratio (b) Without image 17, 1:11 ratio

Chapter 11

Conclusion and Future Work

The results presented in this thesis are very promising and show that the speed of the AAM can be increased without compromising the accuracy of the AAM. The wedgelet enhanced AAM could potentially be used to initialize an AAM and then use a full model for the last few iteration for improvement of the fit. The formulation using triangles can easily be extended to higher dimensions, however one should be aware that this also will significantly decrease the speed of the wedgelet decomposition, but since this is done offline and only once, this disadvantage will be disregarded in most cases. The Markov Random Field has proven successful for improving the geometric properties of the model, it also seems that the MRF does improve the performance of the AAM especially for high compression ratios and makes the model more visually appealing. The MRF extension to the wedgelet enhanced appearance model need however further testing to verify the results.

11.1 Conclusion

We have successfully described, implemented and tested a wedgelet decomposition based on triangles derived from the original formulation proposed by Donoho [9]. We have shown that the derived triangular wedgelets can be used as a sparse image base in the AAM without significantly reducing the performance of the AAM. Furthermore it has been shown that in a true implementation of wedgelets with triangular basis the AAM will due to the sparser basis be able to run much faster. This makes it possible to apply the AAM to images with a much higher resolution such as high resolution 2D images and the easy extension to 3D makes it possible to apply the AAM to 3D images such as CT and MR images. A Markov Random Field has been successfully applied to the wedgelet decomposition for regularizing the geometric property of connecting edges across boundaries. Further more the MRF has improved the segmentation accuracy of the wedgelet enhanced appearance model

11.2 Future Work

First of all an implementation of the AAM that takes full advantage of the wedgelet basis should be implemented to measure the actual performance gain in terms of speed. Furthermore the wedgelet decomposition of multiple images enforces a rather rigid function upon the decomposition. If we instead allow a little displacement of the edges through the images we would perhaps be able to build a better AM. This would also cause some extra landmarks to be added to the training set generated by the wedgelet decomposition. The implementation of the wedgelet decomposition should be made for 3D to test if this is feasible at all. It should be noted that Donoho discourages implementation in higher dimensions, but the use of tetrahedrons should decrease the computational complexity for 3D calculations from $12!$ to $4!$ i.e. the number of edges in a cube and a tetrahedron. Finally a thorough test of the improvement the MRF enhancements offers should be conducted to verify the results achieved in this thesis.

Appendix A

Pseudo Code

To ease further work the following section will describe all algorithms developed for this thesis in pseudo code. The primary reason is that it is very difficult to get into other peoples code, and secondly the code is better documented this way. The following will be based on objects but with a couple of modifications this should work in function based code as well. Lets shortly start with a brief overview of the notation used in this section.

A.0.1 Notation

The notation is primarily Java-like i.e. object oriented hence
w.calc
mean the function *calc* executed on *w*.
this
mean the object the function belongs to. If the reader is familiar with classes, they can be regarded as structs to which one can apply functions. A short example follows here:

Class Example

> ***var :*** **ex1** the defi nition of a variable belonging to the class Example

Classes can be embedded in other classes as follows

Class Example2

> ***Example :*** **ex1** the defi nition of class Example belonging to the class Example2

It should now be possible to read the following section

A.1 Wedgelets

We start out with the wedgelet decomposition.We are primarily focussing on making all algorithms recursive. This has of cause the disadvantage of huge memory consumption but it makes the implementation much easier. We assume the image data given and the interaction with the data is viewed as implicit where ever it might be relevant. Since we work with triangles and in Barycentric coordinates 5 we need some classes first to hold the data.

A.1.1 Classes for the Wedgelets

```
Class Vertice
    var : $x$    the $x$ coordinate if not used set to $0$
    var : $y$    the $y$ coordinate if not used set to $0$
    var : $z$    the $z$ coordinate if not used set to $0$
```

The Vertice class is thought of as a vector, and hence we calculate the cross product, inner product, sum and differences. Having defined a Vertice we define the Edgelet class

```
Class Edgelet

    vertice : $v1$    beginning Vertice
    vertice : $v2$    ending Vertice
```

and hence the first wedgelet will look like this

```
Class Wedgelet

    vertice : $v1$
    vertice : $v2$
    vertice : $v3$
    edgelet : $e$    if degenerate empty
    var : $ca$    if degenerate holds the mean value
    var : $cb$    if degenerate empty
```

We now expand the wedgelet so that it becomes a node in a tree and hereby useable in the wedgelet decomposition described in chapter 6

```
Class Wedgelet

    vertice : $v1$
    vertice : $v2$
    vertice : $v3$
    edgelet : $e$    if degenerate empty
```

var : ca if degenerate holds the mean value
var : cb if degenerate empty
wedgelet : **parent** a pointer to the parent, if root then zero
wedgelet : **childv1** a pointer to the child at $v1$ if leaf then zero
wedgelet : **childv2** a pointer to the child at $v2$ if leaf then zero
wedgelet : **childv3** a pointer to the child at $v3$ if leaf then zero
wedgelet : **childCenter** a pointer to the child at the center, if leaf then zero

A.1.2 Creating the Regression Tree

First we present the initialization algorithm. This procedure initializes the regression tree by subdividing a given number of times.

Algorithm *Wedgelet.initialize(depth)*

```
1.  if depth$<1$
2.    then return
3.    else
4.          depth-1;
5.          $p1 \leftarrow (0.5, 0.5, 0)$
6.          $p2 \leftarrow (0, 0.5, 0.5)$
7.          $p3 \leftarrow (0.5, 0, 0.5)$
8.          childv1←new Wedgelet($v1$,$p1 \cdot v1$+$p1 \cdot v2$,$p3 \cdot v1$+$p3 \cdot v3$);
9.          childv2←new Wedgelet($v2$,$p2 \cdot v2$+$p2 \cdot v3$,$p1 \cdot v2$+$p1 \cdot v1$);
10.         childv3←new Wedgelet($v3$,$p3 \cdot v3$+$p3 \cdot v1$,$p2 \cdot v3$+$p2 \cdot v2$);
11.         childCenter←new Wedgelet($p1 \cdot v1$+$p1 \cdot v2$,$p2 \cdot v2$+$p2 \cdot v3$,$p3 \cdot v3$+$p3 \cdot$
            $v1$);
12.         childv1.initialize(depth);
13.         childv2.initialize(depth);
14.         childv3.initialize(depth);
15.         childCenter.initialize(depth);
16.         childv1.parent←this;
17.         childv2.parent←this;
18.         childv3.parent←this;
19.         childCenter.parent←this;
```

where $\cdot$ denotes the inner product. We now define a function that calculates CPRSS for a wedgelet

Algorithm *Wedgelet.CPRSS(edge)*

```
1.  calculate $\hat{ca}$
2.  calculate $\hat{cb}$
3.  calculate $CPRSS$ from (??)
```

```
4.  returnCPRSS
```

A function that calculates CPRSS without an edge is needed

Algorithm *Wedgelet.CPRSSnoEdge*

```
1.  calculate $CPRSS_{noEdge}$
2.  return $CPRSS_{noEdge}$
```

We now have the building blocks for the wedgelet decomposition for detailed explanation go to page 31. We write the wedgelet decomposition the following way

Algorithm *Wedgelet.decompose()*

```
1.  childv1.decompose()
2.  childv2.decompose()
3.  childv3.decompose()
4.  childCenter.decompose()
5.  if collapsible
6.    then
7.         for all possible edges∈this
8.                $a \leftarrow min(this.CPRSS)$ and $e \leftarrow Edge$
9.         $b \leftarrow min(this.CPRSS_{noEdge})$
10.        if has children $d \leftarrow \sum_{allchildren} c_{child}$
11.        c $\leftarrow min(a, b, d)$
12.        if a=$min(a, b, d)$
13.               mark this $terminal$
14.        if b=$min(a, b, d)$
15.               this.edge$\leftarrow$ $e$ mark this $terminal$
16.        if this.terminal
17.               set this.parrent.collapsible$\leftarrow$ false
18.   else
19.        set this.parrent.collapsible$\leftarrow$ false
20. return
```

A.2 MRF Algorithms

The algorithms needed for the MRF will be described in the following. First we need an initialization that parses the information to the triangles about the external intersection as well as the internal.

Algorithm *MRFInitialize(intersections I)*

```
1.  all intersections I∈ Childv1 →I1
2.  all intersections I∈ Childv2 →I2
3.  all intersections I∈ Childv3 →I3
4.  I2+=Childv1.MRFInitialize(I1)
```

```
5.    I2+=Childv2.MRFInitialize(I2)
6.    I2+=Childv3.MRFInitialize(I3)
7.    all intersections I2∈ ChildCenter →Ic
8.    I2+=Childv3.MRFInitialize(Ic)
9.    return I2
```

we now move to the MRF iteration algorithm which is basically quite simple. To simplify this further we introduce a function that calculates the possibility for a given configuration. This as to be tailored for the given scenario i this case we use the function given in chapter 9

Algorithm *possibility(intersections I)*

```
1.    for all configurations∈ this
2.        for all I ∈ this
3.            P←calculate the probability wrt. I
4.    return P
```

and a selection function which selects the configuration given some update scheme i.e. the Gibbs sampler

Algorithm *select(probability P)*

```
1.    C ← the best configuration given some selection method
2.    return C
```

It is now possible to formalize an algorithm for MRF on a wedgelet decomposition

Algorithm *MRFIteration(intersections I)*

```
1.    if this is terminal node
2.      then
3.          P←possibility(I)
4.          C←select(P)
5.          return new intersections
6.      else
7.          all intersections I∈ Childv1 →I1
8.          all intersections I∈ Childv2 →I2
9.          all intersections I∈ Childv3 →I3
10.         I2+=Childv1.MRFIteration(I1)
11.         I2+=Childv2.MRFIteration(I2)
12.         I2+=Childv3.MRFIteration(I3)
13.         all intersections I2∈ ChildCenter →Ic
14.         I2+=Childv3.MRFIteration(Ic)
15.         return I2
```

All we nee is the sorting at the root and the algorithm for the visitation scheme here. The visitation scheme for the rest of the branches is already built-in in the *MRFIteration*. Let l contain all wedgelets at the root i.e. the triangles in the mean shape. Let $l1$ and $l2$ be vectors containing wedgelets to be updated in the 2 separate sweeps.

```
Algorithm MRFVisitation
1.  $l1 \leftarrow$ first element of $l$
2.  while $l$ is not empty
3.          $w \leftarrow$ next element in $l1$
4.          $l2 \leftarrow$ all neighbors to $w$ in $l$
5.          $w \leftarrow$ next element in $l2$
6.          $l1 \leftarrow$ all neighbors to $w$ in $l$
7.          return $l1$ and $l2$
```

All we need is the MRF at the root i.e. to update the 95 branches. Since we allready have the two sweeps in vector this is straigt forward which concludes the section with algorithms.

Appendix B

Paper

The paper presented on the following pages is in submission to GMBV at CVPR. Co authors on this paper include Rasmus Larsen, Bjarne K. Ersøll and Mikkel B. Stegmann.

Wedgelet Enhanced Appearance Models

Abstract—**Statistical region-based segmentation methods such as the Active Appearance Model (AAM) are used for establishing dense correspondences in images based on learning the variation in shape and pixel intensities in a training set. For low resolution 2D images correspondences can be recovered reliably in real-time. However, as resolution increases this becomes infeasible due to excessive storage and computational requirements. In this paper we propose to reduce the textural components by modelling the coefficients of a wedgelet based regression tree instead of the original pixel intensities. The wedgelet regression trees employed are based on the triangular domains and estimated using cross validation. The wedgelet regression trees serves to 1) reduce noise and 2) produce a compact textural description. The wedgelet enhanced appearance model is applied to a case study of human faces. Compression rates of the texture information of 1:40 is obtained without sacrificing segmentation accuracy noticably, even at compression rates of 1:150 fair segmentation is achieved.**

I. Introduction

The Active Appearance Model (AAM) framework [1] has since its introduction been applied successfully to segmentation of many types of deformable objects in images (e.g. faces, cardiac ventricles, brain structures [1]–[4]). It is based on the estimation of linear models of shape and texture variation by the use of principal components analysis of landmarks coordinates and pixel intensities and subsequent inference of model parameters from unseen images by a tangent plane approximation of the image manifold.

Modelling every pixel intensity is manageable for low-resolution 2D images. But moving to high-resolution 2D images, 3D and even 3D time-series, this approach is rendered at best very slow and at worst infeasible due to excessive storage and computational requirements.

In order to overcome this problem various alternatives to modelling the raw pixel intensities have been considered. Cootes et al. [5] used a sub-sampling scheme to reduce the texture model by a ratio of 1:4. The scheme selected a subset of the pixel intensities based on the ability of each pixel to predict corrections of the model parameters. When exploring different multi-band appearance representations Stegmann and Larsen [6] studied the segmentation accuracy of facial AAMs at different scales in the range $10^3 - 10^5$ pixels obtained by pixel averaging. Wolstenholme and Taylor [7] incorporated a truncated Haar wavelet basis into the AAM framework and evaluated it on a brain MRI data set at a compression ratio of 1:20. Later, Stegmann, Forchhammer, and Cootes [8] further evaluated the use of the Haar wavelet as well as the Cohen-Daubechies-Feauveau [9] wavelet family in the AAM framework. Compression rates of 1:40 without compromising segmentation accuracy were obtained on a set of face images.

Donoho [10] suggested a wedgelet representation for the texture as a means of edge detection and image compression. An image is represented by a collection of dyadically organised indicator functions with a variety of locations, scales and orientations. The classification and regression tree (CART) algorithm [11] uses sequential binary splitting of the spatial domain parallel to the coordinate axes, with splits allowed at every data point. In contrast to this the wedgelet regression tree obey special constraints. Only dyadic partitioning (i.e. recursive midpoint splitting) is allowed, with the added feature that at each terminal node a set of affine splits are also applicable. The wedgelet tree is a quadtree [12] with terminal nodes being either a dyadic (degenerate wedgelet) or a affinely split dyadic (non-degenerate wedgelet). The constrained splitting leads to fast algorithms. Within each resulting image terminal node (wedge or square) the pixel values are regressed to their mean value.

In this paper we generalize the wedgelet transform to triangulated domains (cf. triangulated quadtrees [13]). This has the major advantages of rendering the wedgelet representation independent of piece-wise affine warps of the triangulated domain. Such piece-wise affine warps is customarily chosen in AAM for their speed [4] and the triangulated wedgelet representation thus embraces this choice. The wedgelet transform results in a truncated change of basis for the texture and is represented by a regression tree. The regression tree is estimated using the minimization of the cross validation prediction error across the training set.

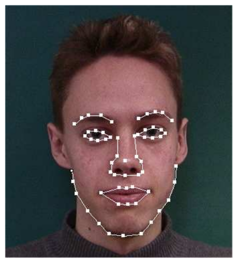
The segmentation accuracy in a wedgelet based AAM is evaluated for a case of human face segmentation using cross validation.

II. Active Appearance Models

AAMs establish a compact parameterization of object variability as learned from a training set by estimating a set of latent variables. The modeled object properties are usually shape and pixel intensities. The latter is hence forward denoted texture. By exploiting prior knowledge of the nature of the optimization space, these models of shape and texture can be rapidly fitted to unseen images, thus providing image interpretation through synthesis.

Training examples are defined by marking up each example image with points of correspondence (i.e. landmarks) over the set either by hand, or by semi- to completely automated methods. From these landmarks a shape model [14] is built. Further, given a suitable warp function a dense (i.e., per-pixel) correspondence is established between the convex hull of the landmarks in each training example. Thus allowing for modeling of texture variability.

Joint variability in shape and texture is modeled by a set of truncated principal components, estimated by an eigen-analysis of the dispersions of shape and texture across the



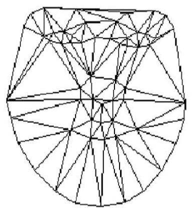


Fig. 1. (a) Human face annotated with 58 landmarks. (b) Mean shape.

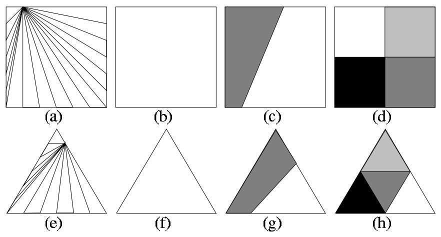


Fig. 2. Templates on the dyadic domain (top row) and the triangulated domain (bottom row). Plots (a) and (e) show the applicable affine split beginning at a particular perimeter point.

training set. The shape examples are aligned to a common mean using a Generalized Procrustes Analysis (GPA) [15], [16] where all effects of translation, rotation and scaling are removed. The obtained Procrustes shape coordinates are subsequently projected into the tangent plane to the shape manifold, at the pole given by the mean shape. The texture examples are warped into correspondence using a piece-wise affine warp and subsequently sampled from this shape-free reference. Typically, this geometrical reference shape is the Procrustes mean shape.

The resulting model consists of a joint shape and texture model. Let $\boldsymbol{s} = \text{vec}\{(x_{ij})\}$, $i = 1, \dots, P$, $j = 1, \dots, d$ be P synthesized landmarks coordinates in d dimensions, and let $\boldsymbol{t} = \text{vec}\{(I_{kl})\}$, $k = 1, \dots, N$, $l = 1, \dots, c$ be N synthesized pixel intensities in c color components. Furthermore, let $\bar{\boldsymbol{s}}$ and $\bar{\boldsymbol{t}}$ denote the mean shape and texture. Synthesized examples are parameterized by $\boldsymbol{\theta}$ and generated by

$$\begin{aligned} \boldsymbol{s} &= \bar{\boldsymbol{s}} + \boldsymbol{\Phi}_s \boldsymbol{\theta} \\ \boldsymbol{t} &= \bar{\boldsymbol{t}} + \boldsymbol{\Phi}_t \boldsymbol{\theta} \end{aligned} \tag{1}$$

where $\boldsymbol{\Phi}_s$ and $\boldsymbol{\Phi}_t$ contain the first p eigenvectors of the estimated joint dispersion matrix of shape and texture. Eq. (1) constitutes the appearance model.

In order to infer the parameters $\boldsymbol{\theta}$ as well as the four scalar parameters of a Euclidean similarity group, $\boldsymbol{\psi}$[1] of an previously unseen image a Gaussian error model between model and pixel intensities is assumed. Furthermore, a linear relation between changes in parameters and difference between model and image pixel intensities $\Delta \boldsymbol{t}$ is assumed, i.e.

$$\Delta \boldsymbol{t} = \boldsymbol{J} \begin{bmatrix} \Delta \boldsymbol{\psi} \\ \Delta \boldsymbol{\theta} \end{bmatrix} \tag{2}$$

$\boldsymbol{J}$ may be estimated by weighted averaging over pertubations of model parameters and training examples. For an in depth description of AAM and the software implementation used the reader is referred to [1], [4], respectively.

The relation in Eq. (2) is inverted using the least squares solution.

[1]scale,orientation, and translation

$$\begin{bmatrix} \widehat{\Delta \theta} \\ \widehat{\Delta \psi} \end{bmatrix} = \left(\boldsymbol{J}^T \boldsymbol{J}\right)^{-1} \boldsymbol{J}^T \Delta \boldsymbol{t} = \boldsymbol{R} \Delta \boldsymbol{t}$$

The computational problem lies in the repeated application of this relation in the innermost loop of the fitting algorithm. $\boldsymbol{R}$ is a non-sparse matrix of dimensions $(p + 4) \times (Nc)$. Nc increase exponentionally with spatial dimension. To reduce the computational burden we propose to use a truncated basis for the representation of the pixel intensities. This introduces the added overhead of transforming between image pixel intensities and this new representation. However, if this transform is based on a sparse matrix, as is the case with wavelet and wedgelet transforms the computational burden can be considerably reduced.

Wedgelet Decomposition

The wedgelet approach is a way of representing images locally, orientation adaptively and at the appropriate scale. It involves a very simple basis used at different scales. The formulation in [10] is for a dyadic domain but the nature of the AAM imposes a shift of basis from dyadic to triangulated domains.

For each node in the wedgelet tree we may consider three wedgelets types: 1) a degenerate wedgelet, this is a terminal node without an affine split (cf. Fig. 2(b) and 2(f)); 2) a non-degenerate wedgelet, this a terminal node with an affine split (cf. Fig. 2(c) and 2(g)); and 3) an interior node corresponding to a step through scale space (cf. Fig. 2(d) and 2(h)).

The wedgelet decomposition is seen to be embedded into a quadtree structure. In this structure the templates are the nodes and the step through scale space the branches. Furthermore, the terminal nodes are all either degenerate or non-degenerate wedgelets. The resulting structure is a regression tree [11]. The corresponding regression model predicts the pixel intensity, y, at each image coordinate $\boldsymbol{x}$ with a constant c_m in each region R_m:

$$\hat{f}(\boldsymbol{x}) = \sum_m c_m I\{\boldsymbol{x} \in R_m\} \tag{3}$$

For the sum of squared error loss criterion $\sum_i ||(y_i - f(\boldsymbol{x}_i)||^2$ it is easy to see that that optimal $\hat{c}_m$ is just the average of y_i in region R_m, $\hat{c}_m = \text{ave}(y_i | \boldsymbol{x}_i \in R_m)$.

The optimal partitioning is found by a bottom-up approach. For each triangle at each level we seek the model that minimizes the K-fold cross-validation estimate of the prediction error across all affine splits/no split. Let $\kappa : \{1, \dots, n\} \mapsto \{1, \dots, k\}$ be an indexing function that indicates the partition to which training object (image) k is allocated by randomization, and denote by $\hat{c}_j^{-\kappa(k)}$, $j \in \{a(s), b(s), c\}$ the regression parameters estimated with the $\kappa(k)$'th part removed. The cross validation errors are

$$\begin{aligned} \text{CVE}_{\text{split}} &= \sum_{k=1}^{n} \sum_{\boldsymbol{x}_i \in R_{a(s)}} ||y_i - \hat{c}_{a(s)}^{-\kappa(k)}||^2 \\ &\quad + \sum_{\boldsymbol{x}_i \in R_{b(s)}} ||y_i - \hat{c}_{b(s)}^{-\kappa(k)}||^2 \\ \text{CVE}_{\text{no split}} &= \sum_{k=1}^{n} \sum_{\boldsymbol{x}_i \in R_c} ||y_i - \hat{c}_c^{-\kappa(k)}||^2 \end{aligned} \quad (4)$$

where $R_{a(s)}$ and $R_{b(s)}$ are the regions resulting from splitting a dyadic or a triangle by an affine split s. R_c is the entire dyadic or triangle.

The optimal split/no split cross validation errors for a triangle and its three siblings are then compared to those for their parent in order to determine if a non-degenerate or a degenerate wedgelet should be declared or if the four siblings should be merged into a triangle a the next higher (parent) level.

In order to be able to control the compression ratio obtained, we add a complexity penalty to the error term over which we carry out cross validation. This complexity penalty is proportional to the image variance, the area of the root triangle and inverse proportional to the area of the triangle under considerations, i.e.

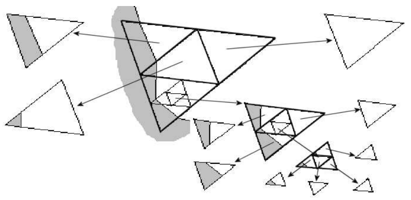
$$\text{CP}(\lambda) = \lambda^2 \cdot \sigma^2 \frac{A_{\text{root}}}{A_{\text{triangle}}} \quad (5)$$

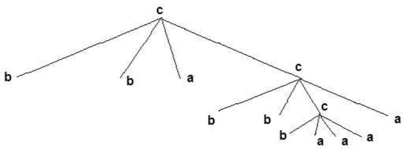
In comparison Donoho [10] proposed a similar complexity penalized residual sum of squares criterion for the case of wedgelet compression over a single image , i.e.

$$\text{CPRSS}(\mathcal{P}, \lambda) = \sum_i ||y_i - f(\boldsymbol{x}_i; \mathcal{P})||^2 + \lambda^2 \# \mathcal{P} \quad (6)$$

where f is the regression model in Eq. (3), $\mathcal{P}$ is the partition, and $\#\mathcal{P}$ is the cardinality of $\mathcal{P}$. Our complexity penalty in Eq. (5) uses an relative area weighting to compensate for local size differences across the training set.

The optimization over all affine splits is conducted as an exhaustive search over a reasonable discretization (cf. Figs. 2(a) and 2(e)). The indexing of pixels within a triangle and computation of areas are conveniently done by the use of barycentric coordinates [17] (cf. App.). Fig. 3 shows how a result on a binary image might look.


(a)



(b)

Fig. 3. A representation of a binary image and the resulting tree structure; the tree nodes are of types (a) degenerate, (b) non-degenerate, (c) interior.

When working on an AAM, an initial triangulation as available from the annotation and Delaunay triangulation on the mean shape (cf. Fig. 1(b)). This initial collection of triangles will be the root of the tree. From here each branch will be equivalent to the type of tree shown in Fig. 3.

III. Results

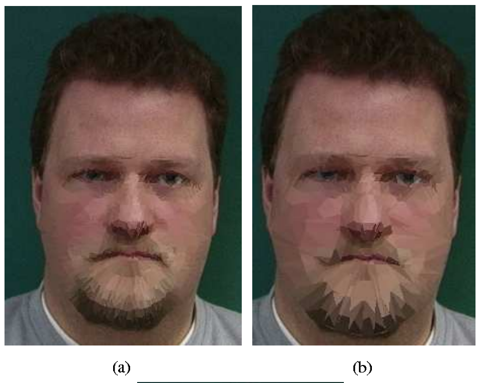
Data for the experiments is an image database of 37 annotated faces. Each image is a 640 × 480 RGB image of a face of an adult human. The data set consists of images of 7 female and 30 male faces. Each face has been manually annotated with 58 corresponding landmarks (see [4] for a detailed analysis).

Fig. 4 shows the result of compressing a single image using the approach described above. The original Delauney triangulation of the face data set (cf. Fig. 1(b)) is subdivided using the penalized complexity criterion in Eqs. (4) and 5) above. In Fig. 4 wedgelet compression at rates 1:3 and 1:40 for a single face are shown.

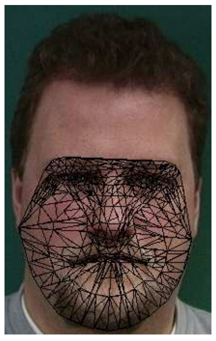
In Fig. 5 the combined principal components of the texture descriptors c_m and the tangent space aligned landmark coordinates are shown. Comparing Figs. 5(a)-5(c) with a compression ratio of 1:3 and Figs. 5(b)-5(d) with a ratio of 1:40, we see that the the first principal components contain the same variations independent of the compression rates.

In Fig. 6 examples of the segmented facial features using a wedgelet enhanced AAM with compression rates 1:3 and 1:40 are shown. Again the results are indistinguishable.

In order to compare the segmentation quality of the wedgelet enhanced AAM and its ability to perform with increasing compression ratio we have conducted a cross-validation across the construction of both the wedgelet decomposition of the training set and the construction of the



(a) (b)



(c)

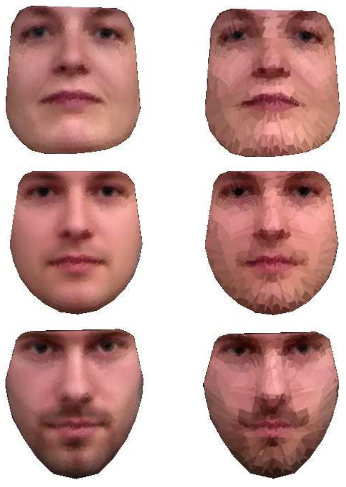
Fig. 4. Images compressed using triangulated wedgelets. (a) 1 : 3 ratio and (b) 1 : 40 using the triangulation from Fig. 1(b). (c) Result (b) superimposed with the subdivided original mesh.

AAM. The average landmark distance from model to ground truth has been used to measure the performance.

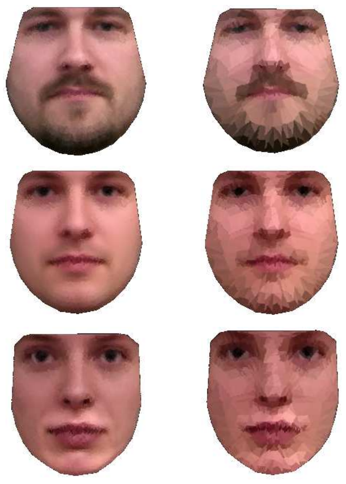
The models were initialized using a displacement of 10% of the width and the height of the mean AAM model from the optimal position in the x and y direction.

For face segmentation we show compression rates of 1:150 with a decrease in segmentation accuracy of 8%. For face segmentation the wavelet compressed AAMs reports compression rates of 1:20 with a decrease in segmentation accuracy of 7% [7]; and a compression rate of 1:40 with a decrease in segmentation accuracy of 8% [8]. These experiments were carried out on grayscale images.

As expected a slight decrease in performance is seen as the compression ratio increases see Fig 7(a). Furthermore, a better ratio can be achieved using wedgelets with good results all the way up to 1:150, see Fig 7(b). Since the method proposed can easily be extended to 3-D, this makes it possible to apply the AAM onto high resolution medical images such MR or CT images both in 2 and 3 dimensions. The major difference between wedgelets and wavelets is the resulting image and



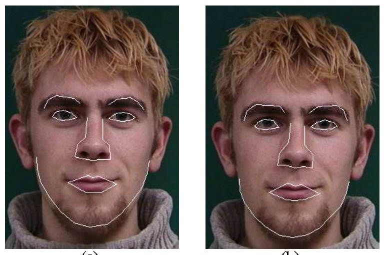
(a) PC_1 (b) PC_1



(c) PC_2 (d) PC_2

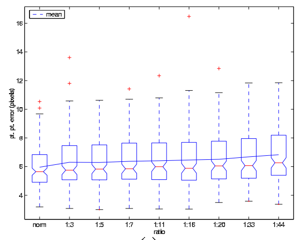
Fig. 5. 1st and 2nd principal components shown at +3 standard deviation, mean and -3 standard devciation for (a-b) wedgelet compression ratio 1:3, (c-d) wedgelet compression ratio 1:40

the computational time. On one hand the wavelet gives nice and smooth images very pleasing to the eye. However, it should be note that the purpose of the wedgelet enhanced appearance model is segmentation with minimum storage and computational cost and not image reconstruction. Therefore the model should not be evaluated on the "blockyness" of Fig. 4 . Fig. 4 serves to demonstrate where in the images important information regarding segmentation is present. Fig. 5 demonstrates that the high order principal components of the

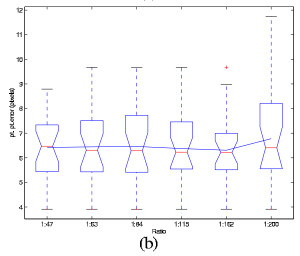


(a) (b)

Fig. 6. (a) Wedgelet AAM using a wedgelet compression rate of 1:3, (b) 1:40.



(a)



(b)

Fig. 7. Segmenation accuracy determined by cross validation across wedgelet compression and AAM segmentation.

uncompressed and the compressed data set are similar. On the other hand it is computational more demanding due to the transform. Wedgelets produce visually more coarse results than wavelets. However, they reduce the texture descriptor size and hereby reduce the computational cost and storage cost significantly due to the reduction of $\mathbf{R}$. Since the number of latent variables is almost unchanged, the size reduction of $\mathbf{R}$ is approximately the same as the ratio. This increases the speed and the storage cost of the AAM.

IV. Conclusion and Future work

Different criteria have been reported in the AAM literature for choosing model complexity. In the original formulation a threshold at 95-98% was used on the cumulated variance described by the principal components. Alternatives include 1) "elbow" identification in scree-plots of covariance matrix eigenvalues [18]; 2) statistical comparison of covariance matrix scree-plot with scrambled data covariance scree-plots [19]; 3) probabilistic PCA modelling using AIC, BIC or cross validation [20]. For the compression AAM models the following methods have been applied. For the subsampled AAM [5] the subsampling is given by the union of the $u\%$ elements (pixels) with largest absolute value in each column of R (since large regression coefficients does not necessarily mean high significance for multiple regression this may not be a good idea). For the wavelet compression AAMs [7], [8] the criterion used is based on retaining as much variance across the training set as possible for a given compression rate. The original wedgelet representation [10] uses a complexity penalized (CP) RSS criterion. For the multi object (image) situation we introduce the complexity penalized cross-validation error in Eqs. (4) and (5). The complexity penality in Eq. (5) favours relative larger wedges/triangles (compensating for scaling differences/warps between images). The multiplication of the CP with the image variance compensates for (gain) differences between training images.

We have defined a 2D wedgelet transform on triangulated domains. We have demonstrated how cross-validation can be used to arrive a truncated wedgelet representation of the texture in an active appearance model setting. The triangulated wedgelet transform embraces the the triangulated domains used in AAMs in contrast to previous attempts based on wavelet transforms. The wavelet transforms require pixelated handling of boundaries because they are inherently based on rectangular domains. The wedgelet scheme can readily be extended to higher dimensions and we aim to pursue this in near future. Also, additional constraints such as enforcing connecting edges between neighboring wedgelets as well as allowing pertubations of semi-landmarks introduced on the edges between their parent landmarks are considered.

In applying the wedgelet transform to ensembles of face images we arrive at compression rates up to 1:150 with only subtle degradation of the segmentation accuracy. Even higher compression rates will apply for 3D and 3D+time.

Appendix
Barycentric coordinates

Homogeneous barycentric coordinates known from computer graphics are introduced [17]. A barycentric coordinate system is a local coordinate system for triangles in two dimensions, but can easily be extended to higher dimensions. Barycentric coordinates are triples of numbers (t_1, t_2, t_3) corresponding to masses placed at the vertices of a reference triangle $\triangle A_1A_2A_3$. These masses determine a point P, which is the geometric centroid of the three masses. The vertices of the triangle are given by $A_1 = (1, 0, 0)$, $A_2 = (0, 1, 0)$ and $A_3 = (0, 0, 1)$. The areas of $\triangle A_2PA_3$, $\triangle A_1PA_3$ and $\triangle A_1PA_2$ are proportional to t_1, t_2 and t_3. For homogeneous barycentric coordinates the following is true $t_1 + t_2 + t_3 = 1$ and every point P with coordinate $c = (t_1, t_2, t_3)$ where $0 \leq t_1, t_2, t_3 \leq 1$ lie within $\triangle A_1A_2A_3$. Moving from image coordinates to barycentric coordinates constitutes a shift of basis.

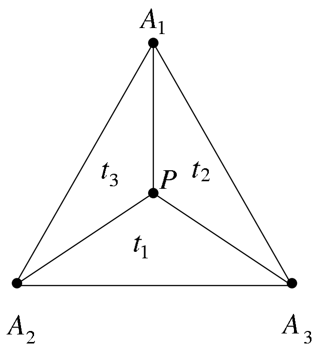


Fig. 8. Barycentric coordinates.

References

[1] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *IEEE Trans. on Pattern Recognition and Machine Intelligence*, vol. 23, no. 6, pp. 681–685, 2001.

[2] 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*, vol. 1. SPIE, 2000.

[3] S. C. Mitchell, B. P. F. Lelieveldt, R. J. van der Geest, H. G. Bosch, J. H. C. Reiber, and M. Sonka, "Multistage hybrid active appearance model matching: Segmentation of left and right ventricles in cardiac MR images," *IEEE Transactions on Medical Imaging*, vol. 20, no. 5, pp. 415–423, May 2001.

[4] M. B. Stegmann, B. K. Ersbøll, and R. Larsen, "FAME – a flexible appearance modelling environment," *IEEE Transactions on Medical Imaging*, vol. 22, no. 10, pp. 1319–1331, may 2003. [Online]. Available: http://www.imm.dtu.dk/pubdb/p.php?1918

[5] 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.*, vol. 2. Univ. Southampton, 1998, pp. 680–689.

[6] M. B. Stegmann and R. Larsen, "Multi-band modelling of appearance," *Image and Vision Computing*, vol. 21, no. 1, pp. 61–67, Jan. 2003.

[7] C. B. H. Wolstenholme and C. J. Taylor, "Wavelet compression of active appearance models," in *Medical Image Computing and Computer-Assisted Intervention, MICCAI*, 1999, pp. 544–554.

[8] M. B. Stegmann, S. Forchhammer, and T. F. Cootes, "Wavelet enhanced appearance modelling," in *International Symposium on Medical Imaging 2004, San Diego CA, SPIE*. SPIE, 2004. [Online]. Available: http://www.imm.dtu.dk/pubdb/p.php?2807

[9] A. Cohen, I. Daubechies, and J.-C. Feauveau, "Biorthogonal bases of compactly supported wavelets," *Comm. Pure and Applied Mathematics*, vol. 45, pp. 485–560, 1992.

[10] D. Donoho, "Wedgelets: Nearly minimax estimation of edges," *Annals of Statistics*, vol. 27, pp. 859–897, 1999.

[11] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and regression trees*. Monterey, California: Wadsworth & Brooks/Cole advanced books & software, 1984, 358 pp.

[12] R. A. Finkel and J. L. Bentley, "Quad trees: A data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, pp. 1–9, 1974.

[13] M. W. Bern, D. Eppstein, and J. R. Gilbert, "Provably good mesh generation," in *Proc. 31st Symp. Foundations of Computer Science*, vol. I. IEEE, Oct. 1990, pp. 231–241.

[14] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models – their training and application," *Computer Vision, Graphics and Image Processing*, vol. 61, no. 1, pp. 38–59, Jan. 1995.

[15] J. C. Gower, "Generalized Procrustes analysis," *Psychometrika*, vol. 40, pp. 33–50, 1975.

[16] J. M. F. ten Berge, "Orthogonal Procrustes rotation for two or more matrices," *Psychometrika*, vol. 42, pp. 267–276, 1977.

[17] A. F. Möbius, Ed., *Der barycentrische Calcul, ein neues Hülfsmittel zur analytischen Behandlung der Geometrie / dargestellt und angewendet*. Lpz, 1827.

[18] T. Hastie, R. Tibshirani, and J. Friedman, *Elements of Statistical Learning: data mining, inference and prediction*. Springer, 2001.

[19] R. R. Paulsen, R. Larsen, S. Laugesen, C. Nielsen, and B. K. Ersbøll, "Building and testing a statistical shape model of the human ear canal," in *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2002, 5th Int. Conference, Tokyo, Japan,*, ser. Lecture Notes in Computer Science, vol. 2488. Berlin Heidelberg: Springer Verlag, 2002, pp. II–373–II–380.

[20] R. Larsen and K. B. Hilger, "Probabilistic generative modelling," in *13th Scandinavian Conference on Image Analysis (SCIA) , Gothenburg, Sweden*, ser. Lecture Notes in Computer Science, J. Bigün and T. Gustavsson, Eds., vol. 2749. Springer, jul 2003, pp. 861–868. [Online]. Available: http://springerlink.metapress.com/openurl.asp?genre=article\&issn=0302-%9743\&volume=2749\&spage=861

Appendix C

Additional Results

The following sections contain additional results to those presented in the chapters.

C.1 Additional Wedgelet Enhanced AM

This section contains a wedgelet enhanced AM for each compression ratio used to generate the results in capter 7 except for the 1:3 ratio. These are the model where the 39th image has been left out of the constrction.

C.2 Additional MRF Results

Here we present a few extra results from the MRF modified wedgelet decomposition.

C.3 Additional Fits

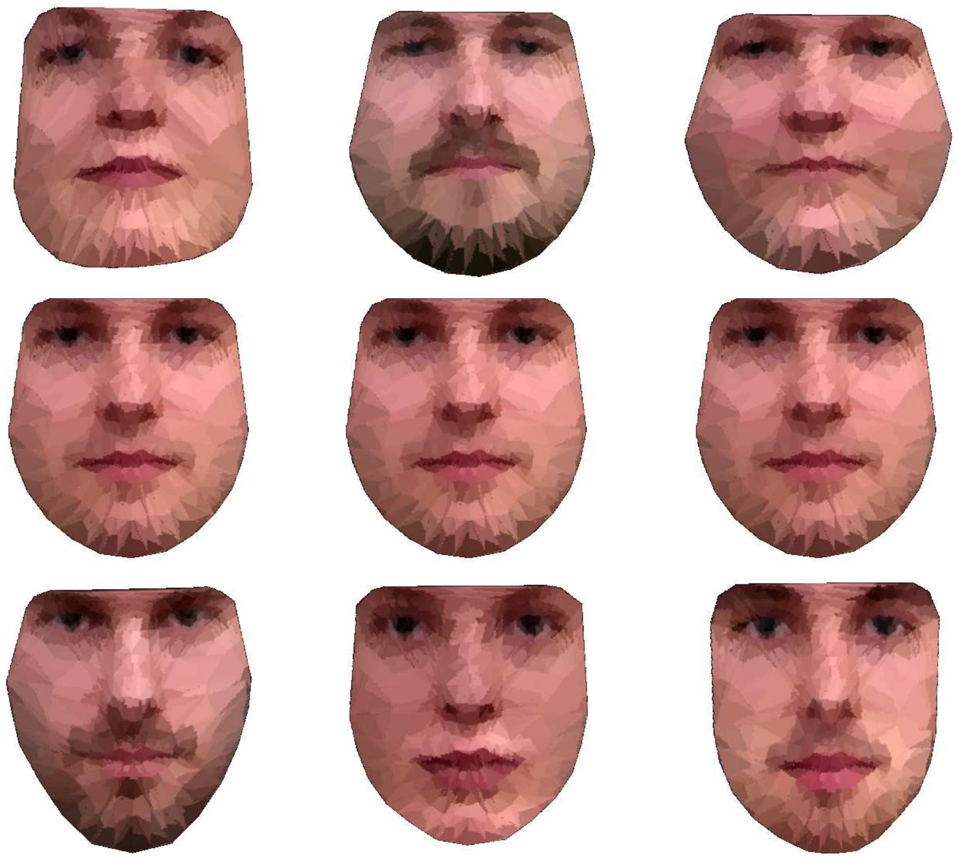


Figure C.1: (a),(b), (c) shows the first three combined principal components +- 3 std. and the mean shape ratio 1:44.

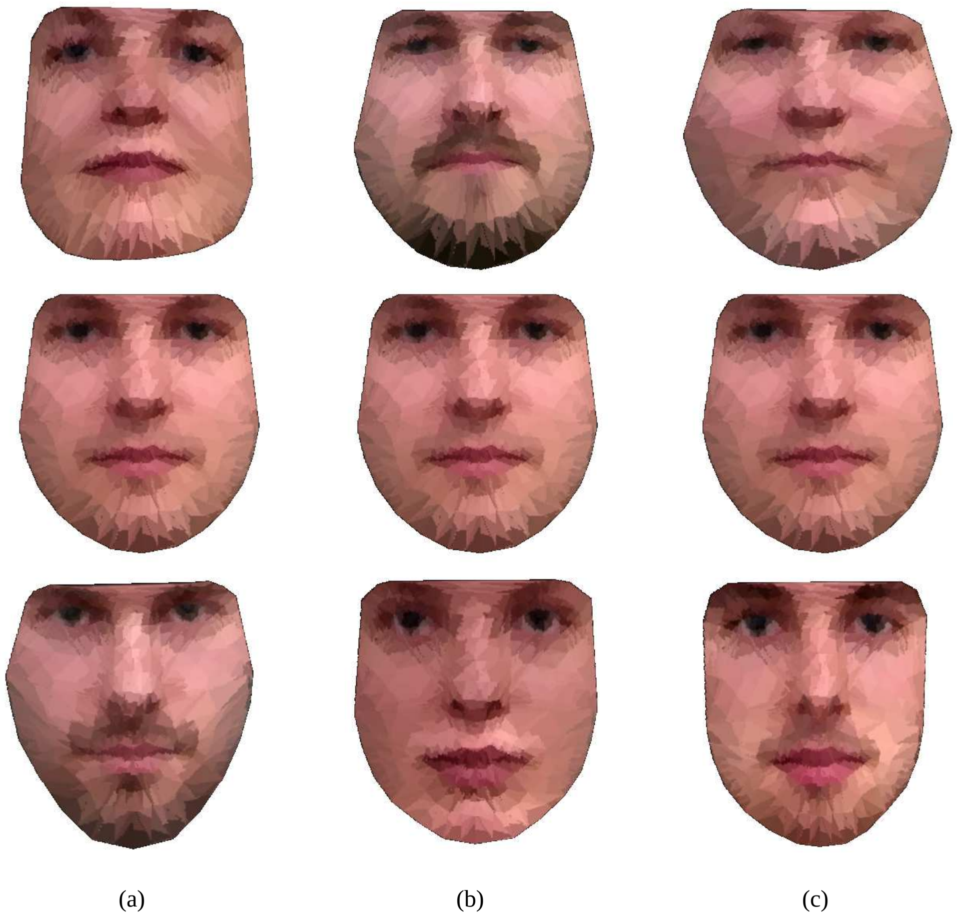


Figure C.2: (a),(b), (c) shows the first three combined principal components +- 3 std. and the mean shape ratio 1:33.

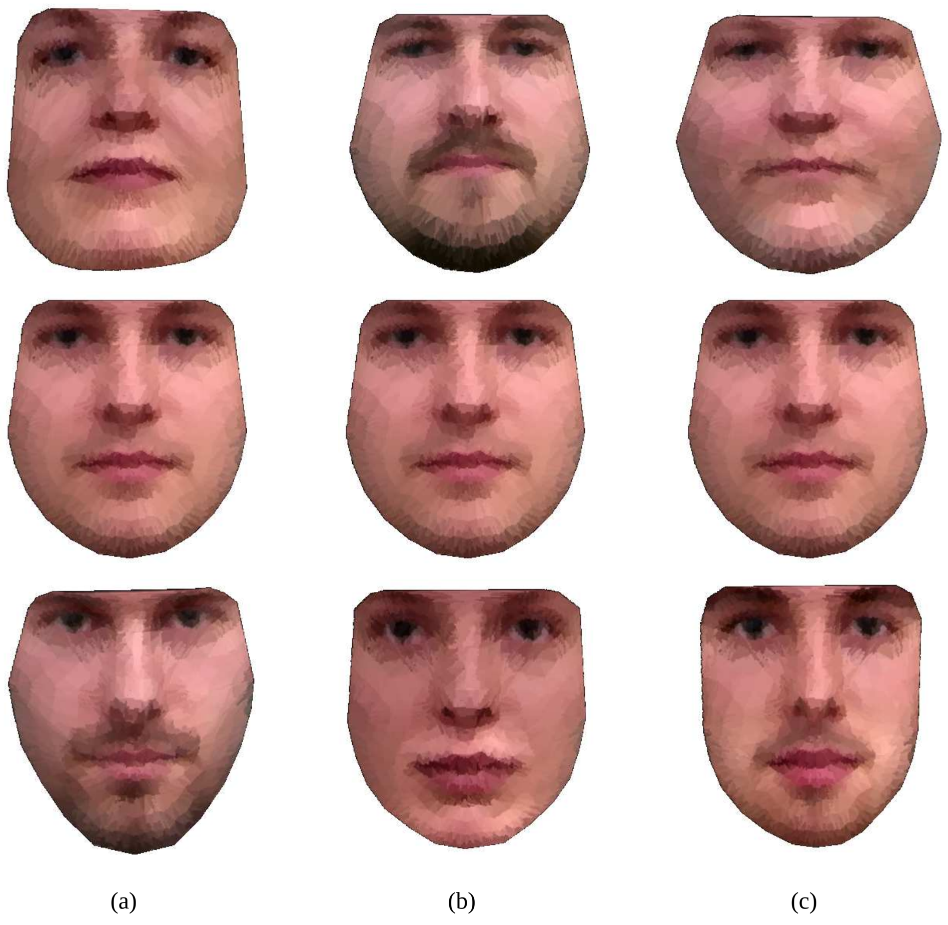


Figure C.3: (a),(b), (c) shows the first three combined principal components +- 3 std. and the mean shape ratio 1:20.

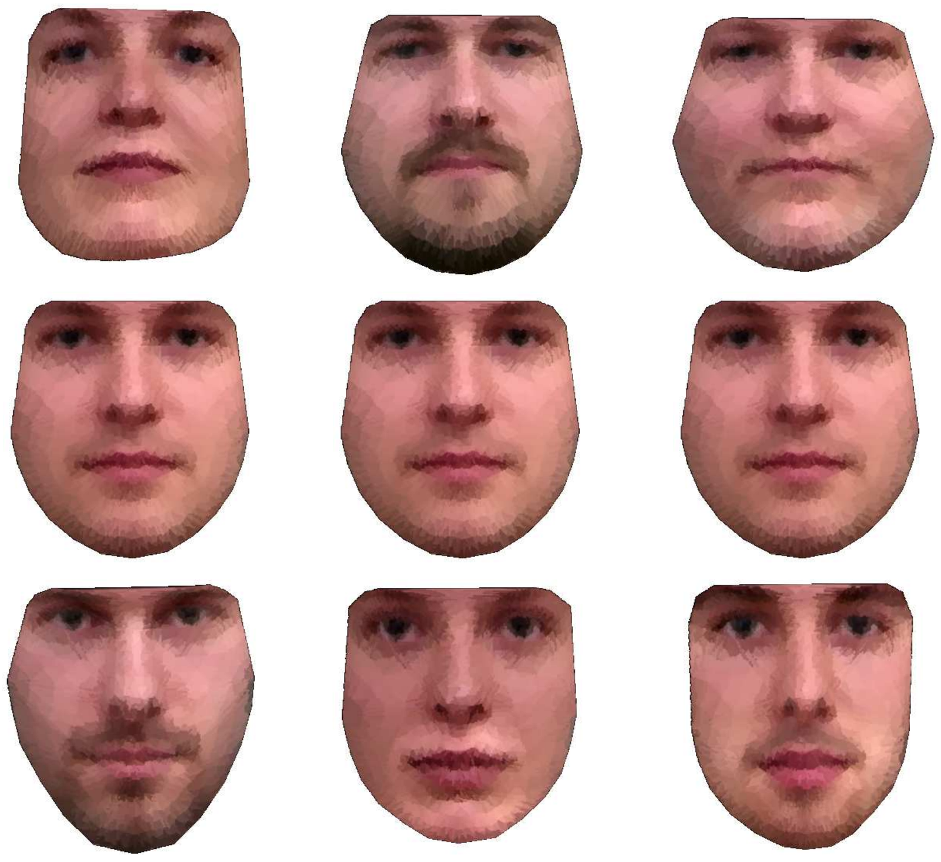


Figure C.4: (a),(b), (c) shows the first three combined principal components +- 3 std. and the mean shape ratio 1:16.

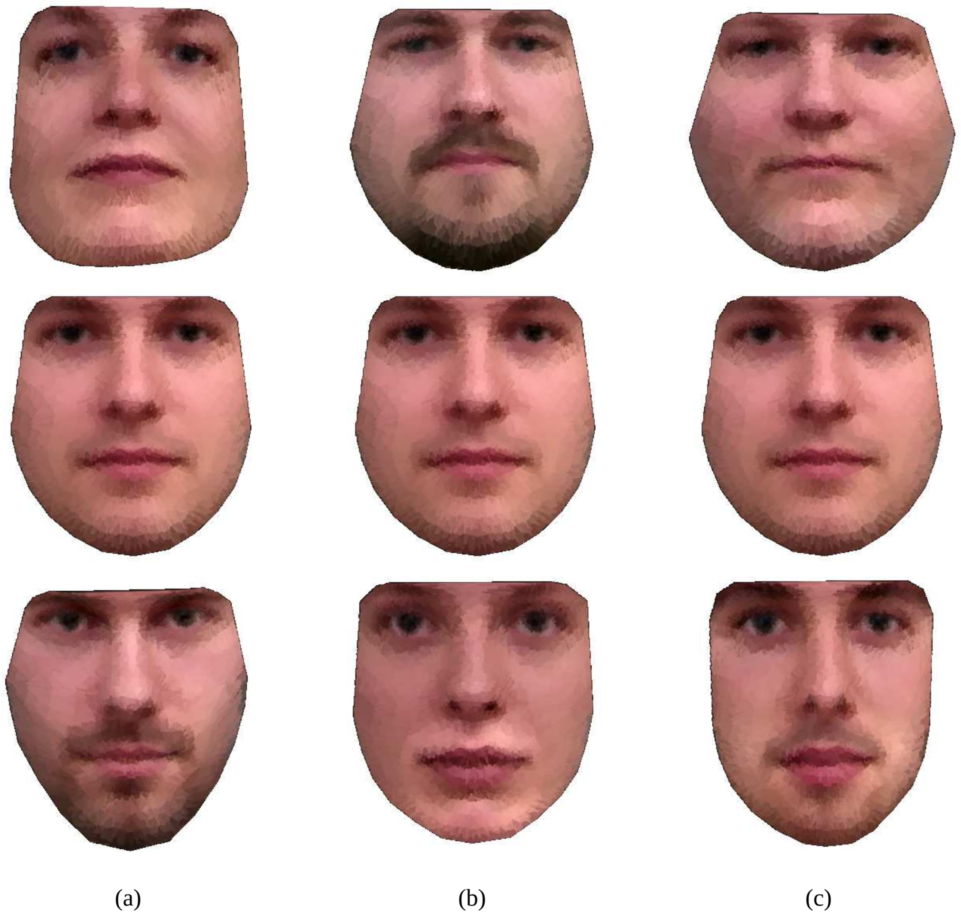


Figure C.5: (a),(b), (c) shows the first three combined principal components +- 3 std. and the mean shape ratio 1:11.

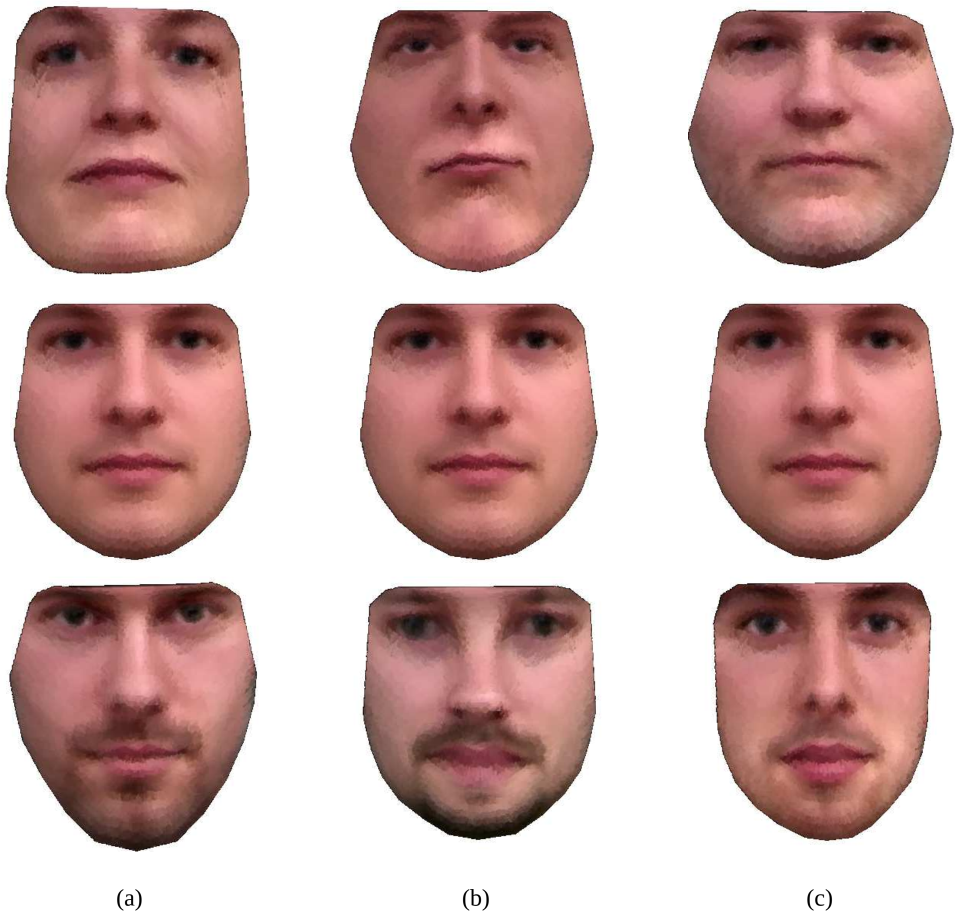


Figure C.6: (a),(b), (c) shows the first three combined principal components +- 3 std. and the mean shape ratio 1:7.

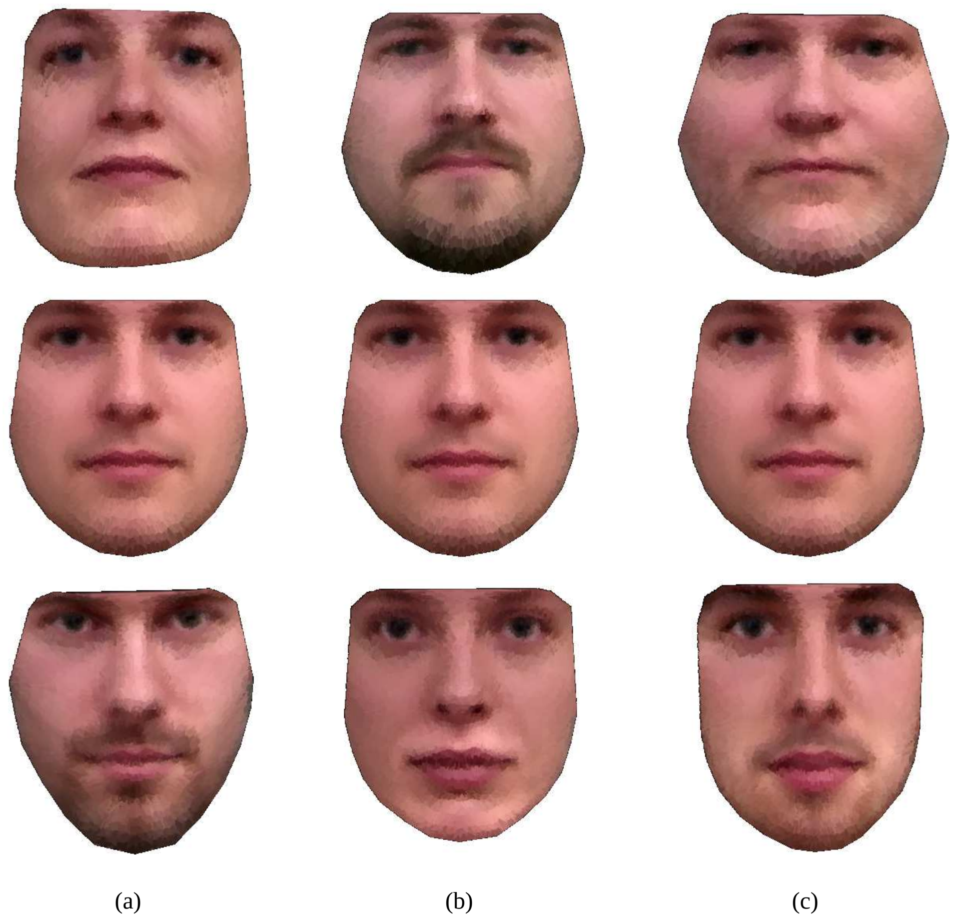
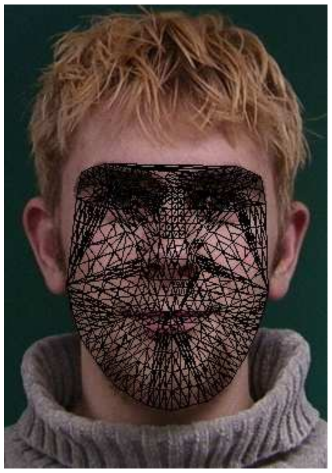
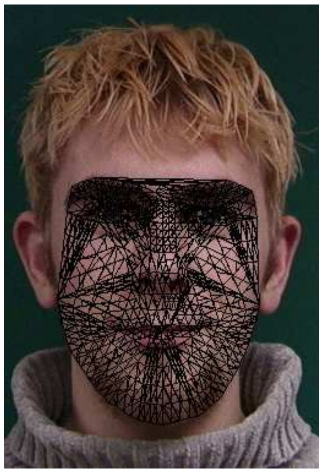


Figure C.7: (a),(b), (c) shows the first three combined principal components +- 3 std. and the mean shape ratio 1:5.

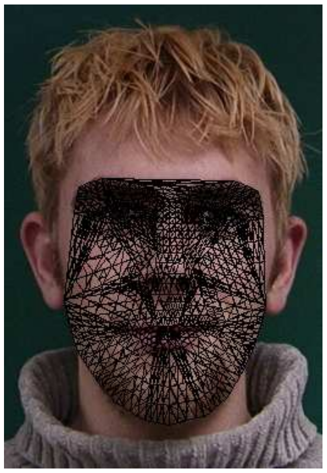


(a)

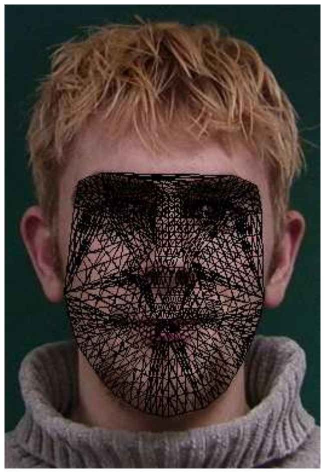


(b)

Figure C.8: Images at ratio 1:16(a) without the 17th image, (b) without the 16th image



(a)



(b)

Figure C.9: Images at ratio 1:11(a) without the 17th image, (b) without the 16th image

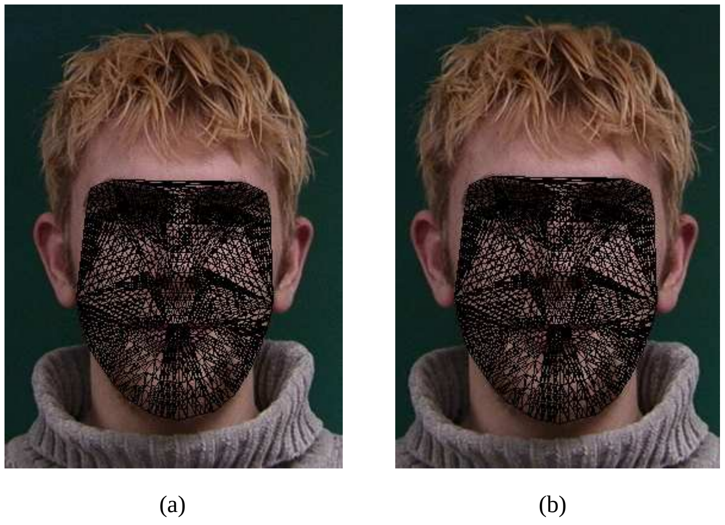


Figure C.10: Images at ratio 1:5(a) without the 17th image, (b) without the 16th image

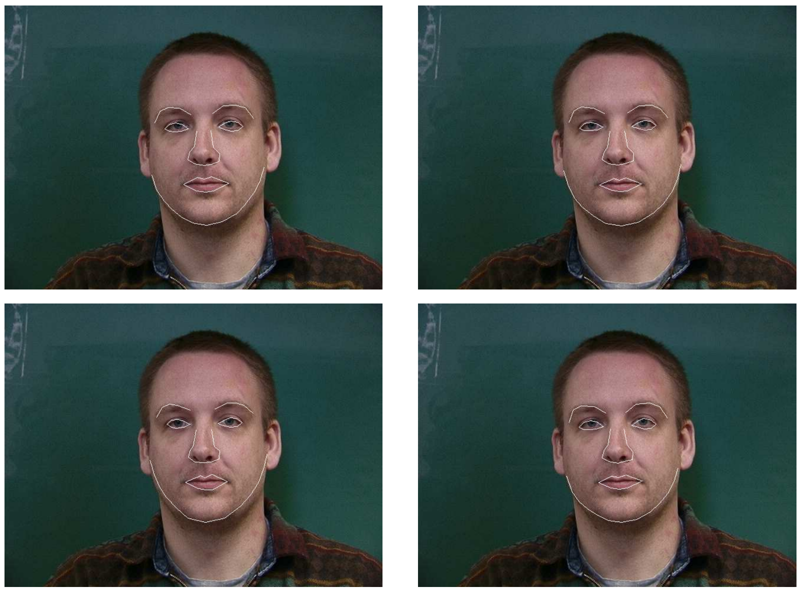


Figure C.11: The fit of the AAM at $1 : 44$ ratio.

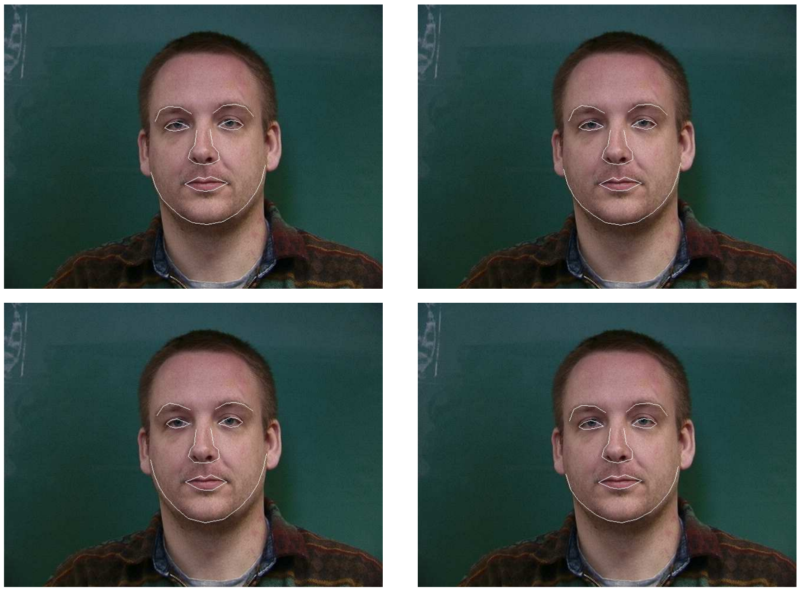


Figure C.12: The fit of the AAM at 1 : 11 ratio.

List of Figures

Bibliography

[1] T. F. Cootes, G. J. Edwards, and C. J. Taylor, “Active appearance models,” *IEEE Trans. on Pattern Recognition and Machine Intelligence*, vol. 23, no. 6, pp. 681–685, 2001.

[2] 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*, vol. 1, SPIE, 2000.

[3] S. C. Mitchell, B. P. F. Lelieveldt, R. J. van der Geest, H. G. Bosch, J. H. C. Reiber, and M. Sonka, “Multistage hybrid active appearance model matching: Segmentation of left and right ventricles in cardiac MR images,” *IEEE Transactions on Medical Imaging*, vol. 20, pp. 415–423, May 2001.

[4] M. B. Stegmann, B. K. Ersbøll, and R. Larsen, “FAME - a flexible appearance modelling environment,” *IEEE Transactions on Medical Imaging*, vol. 22, pp. 1319–1331, may 2003.

[5] 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.*, vol. 2, pp. 680–689, Univ. Southampton, 1998.

[6] C. B. H. Wolstenholme and C. J. Taylor, “Wavelet compression of active appearance models,” in *Medical Image Computing and Computer-Assisted Intervention, MICCAI*, pp. 544–554, 1999.

[7] M. B. Stegmann, S. Forchhammer, and T. F. Cootes, “Wavelet enhanced appearance modelling,” in *International Symposium on Medical Imaging 2004, San Diego CA, SPIE (in press)*, SPIE, 2004.

[8] A. Cohen, I. Daubechies, and J.-C. Feauveau, “Biorthogonal bases of compactly supported wavelets,” *Comm. Pure and Applied Mathematics*, vol. 45, pp. 485–560, 1992.

[9] D. Donoho, “Wedgelets: Nearly minimax estimation of edges,” *Annals of Statistics*, vol. 27, pp. 859–897, 1999.

[10] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and regression trees*. Monterey, California: Wadsworth & Brooks/Cole advanced books & software, 1984. 358 pp.

[11] R. A. Finkel and J. L. Bentley, "Quad trees: A data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, pp. 1–9, 1974.

[12] M. W. Bern, D. Eppstein, and J. R. Gilbert, "Provably good mesh generation," in *Proc. 31st Symp. Foundations of Computer Science*, vol. I, pp. 231–241, IEEE, Oct. 1990.

[13] S. Z. Li, *Markov Random Field Modeling in Computer Vision*. Computer Science Workbench, Springer-Verlag, 1995. 264 pp.

[14] G. Winkler, *Imige Analysis, Random Fields and Markov Chain Monte Carlo Methods*. Springer, 2003.

[15] J. M. Carstensen, *Description and Simulation of Visual Texture*. PhD thesis, Institute of Mathematical Statistics and Operations Research, Technical University of Denmark, Lyngby, 1992. 234 pp.

[16] R. Potts, "Some generalized order-disorder transformations," *Proc. Camb. Phil.*, vol. 48, pp. 106–109, 1952.

[17] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," in *Proceedings of the European Conf. On Computer Vision*, pp. 484–498, Springer, 1998.

[18] M. B. Stegmann and R. Larsen, "Multi-band modelling of appearance," in *First International Workshop on Generative-Model-Based Vision - GMBV* (A. Pece, ed.), (Copenhagen, Denmark), pp. 101–106, DIKU, jun 2002.

[19] M. B. Stegmann, "Analysis of 4D cardiac magnetic resonance images," *Journal of The Danish Optical Society, DOPS-NYT*, pp. 38–39, dec 2001.

[20] I. L. Dryden and K. Mardia, *Statistical Shape Analysis*. Chichester: John Wiley & Sons, 1998. xx + 347 pp.

[21] J. B. Lee, S. Woodyatt, and M. Berman, "Enhancement of high spectral resolution remote sensing data by a noise-adjusted principal components transform," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 28, pp. 295–304, May 1990.

[22] T. Cootes and C. Taylor, "Statistical models of appearance for computer vision," tech. rep., Univerity of Manchester, Manchester M13 9PT, UK, Oct. 2001.

[23] T. Hastie, R. Tibshirani, and J. Friedman, *Elements of Statistical Learning: data mining, inference and prediction*. Springer, 2001.

[24] C. E. L. Thomas H. Cormen and R. L. Rivest, *Introduction to Algorithms*. MIT Press, 1990. 435 pp.

[25] S. Mallat, *A wavelet tour of signal processing*. Academic Press, 1998.

[26] A. Graps, “An introduction to wavelets,” *IEEE Computational Sciences and Engineering*, vol. 2, no. 2, pp. 50–61, 1995.

[27] J. Romberg, M. Wakin, and R. Baraniuk, “Multiscale wedgelet image analysis: fast decompositions and modeling,” 2002.

[28] A. F. Möbius, ed., *Der barycentrische Calcul, ein neues Hülfsmittel zur analytischen Behandlung der Geometrie / dargestellt und angewendet*. Lpz, 1827.

[29] E. Ising, “Beitrag zür Theorie des Ferromagnetismus,” *Zeitschrift für Physik*, vol. 31, pp. 253–258, 1925.