Imperial College London

Expression Transfer

Author:
Martin Papanek

Supervisor:
Professor Duncan Gillies

August 11, 2010

Abstract

Acknowledgments

Contents

Chapter 1

Introduction

1.1 Motivation

Allowing computers to understand the world around them is one of the most intriguing goals of computer science. In order to aid humans in day-to-day tasks, the ideal computer should be able to perceive his surroundings, correctly identify the objects and beings around him and act based on this information. Achieving this level of sophisticated, environment-aware behavior is the focus of popular computer science fields such as machine learning, computer vision and logic.

The problem of understanding the surrounding world can be broken down into a number of sub-problems. First the machine must obtain and process the information on its sensors. Then it has to process this data in order to find objects in the sensory input. Finally, the machine has to assign meaning to the scene it perceived based on the the configuration and the properties of the objects it found. This allows the machine to understand what state the environment is in and it may then utilize this information using simple if-then rules.

For humans, all of the aforementioned sub-problems seem simple. However, programming machines to do the same is quite difficult. Computers often do possess better sensors than most humans and thus are readily able to obtain data from sensors. Yet, they are sorely lacking when it comes to locating objects in this sensory input and correctly assessing the properties and configuration of these objects. While it is possible to locate circles and lines, joining these to locate a face or a tree can only be done if the machine knows what a face or a tree should look like. Thus, the machine needs to have prior information about the objects it can expect. This prior information can be encoded in a *model.* The model describes the structure of the object. This, in turn, allows the machine to explain aspects of its sensory input as the occurrence of that object. Finding objects by means of locating an instance of a model in the input is called *model-based recognition.*

Certain objects may also change their shape or appearance. For example a face may transition from a closed eyed state to an open eyed state. An even better example is the body of a human, which is also highly dynamic. These deformable objects are often of special interest to us in our everyday life. A machine should therefore be able to recognize an arbitrarily deformed object and also correctly identify the the degree of deformation, since the amount of deformation may be crucial for the understanding of the scene. The challenge thus lies in constructing an appropriate *deformable model.*

Given a deformable model, the machine then has to process the sensor input and locate instances of the model by adjusting the model's parameters. This task is known as *model-based object recognition*. The parameters of the model then allow the machine to interpret the scene. In addition to locating an instance of the model in one sensory input the machine should also be able to track its movement given a sequence of snapshots of the environment. In *model-based tracking* an instance of the model is identified and tracked from snapshot to snapshot, in spite of deformations of shape and changes in position.

In this paper we describe convenient deformable shape and appearance models and effective algorithms for locating these models. For our sensory snapshots, we will focus exclusively on images. As we will be dealing with images we will investigate motion tracking and feature detection algorithms. These are necessary to obtain cues from our input image which allow us to first locate the model in the image and then track its movements through successive images.

1.2 Contributions

The area where deformable models are applicable is quite large. This paper will focus on one very interesting application of dynamical models and model extraction which is *expression transfer*. The purpose of expression transfer is to capture the expressions and visemes - speech related mouth articulations - from a video recording of one individual and generate a video of another individual mimicking these expressions and visemes. The alternative to transferring these expressions would be to construct a physical model of the face and simulate the observed expressions and visemes using the model. However, transferring the dynamics of a subjects face to that of another enables us to create very realistic animations without much difficulty. On the other hand, it is quite difficult to generate realistic expressions by setting the appropriate parameters of a physical model simply because of the complexity of the physics behind the movement that is responsible for expressions.

Chapter 2

Background

Considerable amount of research has been done on using models to characterize a deformable object as well as on model-based object recognition and tracking. In this chapter we will present an overview of various modeling techniques. Likewise, we will discuss computer vision algorithms which will allow us to locate the features of interest in the image. These features then make it possible to fit an instance of the model to the image.

2.1 Deformable Models

Models give the computer prior information about the structure of a class of objects. Most real world objects have a dynamic structure. To allow the computer to locate such dynamic objects it is necessary to account for this variability by giving the computer prior information about the possible variations. A deformable model describes the expected structure of a class of objects while at the same time allowing for variations from the expected structure. Thus, if a somewhat deformed instance of this class of objects is present in the image, the deformable model will be able to explain this is a deformation from the expected shape.

The quality of a deformable model can be assessed based on two important characteristics. First, the model should be *general* enough so that it is capable of representing any realistic deformation of the object. On the other hand, the computer should only to be able to find an instance of this model in its surroundings if and only if this object is present. The deformable model therefore has to be *specific* so that it does not locate non-existent instances of a model in the input.Clearly, a model that is too general will inadvertently fit objects from a different class and vice versa a model that is too specific will not be able to explain all the variations of the object structure and thus fail to locate instances of the correct class. The optimal deformable model should balance these two attributes.

The variance in shape or appearance of a model may be due to combination of the following factors:

- Variations of shape due to deformations of the object.
- Arbitrary scaling of the object, possibly due to distance from the observer.
- Arbitrary rotations of the object which may cause occlusions.

- Some measure of Gaussian input noise.
- Differences in color or intensity caused by a change in lighting conditions.

Some of these sources of variation may be explained away using standard machine vision techniques. Noisy input can be explained by means of Gaussian filtering. Other sources of variation such as those due to deformations of the object need to be represented by appropriate parameters of the model. The deformable model simulates the deformations based on the parameter values. The model thus needs to learn the mapping from parameters to deformations. The two most prevalent approaches to discovering this mapping are using either *physical models* or *statistical models*.

Physical models construct deformable models which mimic the elastic deformations of the class of object they model. The parameters of a physical model control the amount of actual physical deformation.

Statistical deformable models are trained on a set of examples of a class of objects. Through statistical analysis a basis for the deformations observed in this set of examples is constructed. This basis then allows the model to predict probable instances of the objects of this class.

Finally, given that a deformable model is uniquely defined by its parameters, the goal of model-based object recognition is then to fit an instance of a model to an the appropriate object in the image. The problem of fitting the model to the image is in essence an optimization problem. In order to fit the model we need to correctly adjust parameters that control the model. In addition to determining the intrinsic parameters of the model it is necessary to also find how the object is rotated, moved and scaled to explain the variation in structure which is not caused by the deformations. The recognition algorithm also needs to be robust in order to deal with variations caused by noise.

2.1.1 Physics Based Models

Natural objects all obey physical laws. Human and animal bodies change shape when their muscles contract and loosen, which in turn alters the shape of the elastic soft tissues which surround the muscles. Movement is further constrained by the skeletons and gravity. Due to the dynamic nature of such objects it is quite impractical to model the structure of these objects as consisting of only rigid components. To cope with these highly dynamic bodies researchers have turned to physics to describe the rules governing these dynamics in form of a set of equations. Pentland and Sclaroff in [11] give a closed form solution for extracting a physical model from images. Terzopoulos and Metaxas combine physics-based deformable models with kalman filtering theory in [8].

As the name suggests, physical models emulate the physical laws that govern deformations. Physical models are predominantly formulated using the finite element analysis. An in depth discussion of finite element analysis can be found in [1].

Finite Element Method (FEM)

The FEM method is a numerical engineering technique for the simulation of dynamic behavior of solids and structures. With the finite element method the assumption is made that the object alters shape as if it were made of a elastic material. In physics, *strain* is the measure of deformation or displacement from a rigid body state. The stiffness of the material determines responses to strain and stress and thus describes the degrees of freedom of the material.

The underlying idea behind FEM is that the object is approximated as an assemblage of elements of finite size. These elements are interconnected with each other by nodal points throughout the object. The structure of an object is thus discretized into a mesh of N finite elements. The elements are placed next to each other so that no gaps remain in between. When the object undergoes a deformation, this in turn propagates through the mesh of finite elements which themselves deform accordingly. The displacements within these elements are assumed to be a function of the displacements measured at the nodal points. This assumption is fundamental for the FEM and can be formalized as

$$u^{(m)}(x,y,z) = \mathbf{H}^{(m)}(x,y,z) * \mathbf{U} \tag{2.1}$$

where $u^{(m)}$ is the displacement at x,y, and z coordinate within the element m, $\mathbf{H}^{(m)}$ the displacement interpolation matrix and $\mathbf{U}$ the global displacement measured at every nodal point. With equation 2.1 the displacements at any point in the object can be calculated. The values of the displacement interpolation matrix $\mathbf{H}^{(m)}$ depend on the choice of the finite elements that make up the mesh.

From the displacement $u^{(m)}(x,y,z)$, the strain can be calculated as the derivative of the displacement with respect to x,y, and z. Thus, the strain $\epsilon^{(m)}(x,y,z)$ at the element m is given by

$$\epsilon^{(m)}(x,y,z) = \mathbf{B}^{(m)}(x,y,z) * \mathbf{U} \tag{2.2}$$

where $\mathbf{B}^{(m)}(x,y,z)$ is the strain-displacement matrix. This matrix can be calculated by differentiating the the displacement interpolation matrix $\mathbf{H}^{(m)}$. With equations 2.1 and 2.2 the behavior of the object structure given a global nodal point displacement $\mathbf{U}$ is defined.

The goal of displacement-based finite element analysis is to calculate unknown nodal point displacements from a known force or load acting on the object. When a load is applied to the structure of the object it will cause a deformation of the mesh. The nodal points in the mesh will bounce and move until they reach a state of equilibrium. It is important to stress that in this equilibrium the shape of the object is still deformed due the applied force. However, the nodal points may have assumed new stable positions hence we refer to it as an equilibrium. The equation governing the equilibrium is derived using equations 2.1, 2.2 and the Principle of Virtual Work [1]. It relates the stiffness matrix $\mathbf{K}$ and the unknown nodal displacements $\mathbf{U}$ to the loads $\mathbf{R}$ as follows

$$\mathbf{KU} = \mathbf{R} \tag{2.3}$$

The stiffness matrix $\mathbf{K}$ is calculated as the sum of the stiffness matrices $\mathbf{K}^{(m)}$ of the individual finite elements. The $\mathbf{K}^{(m)}$ are computed from the strain-displacement matrices $\mathbf{B}^{(m)}$ and the elasticity matrix $\mathbf{E}^{(m)}$ as

$$\mathbf{K} = \sum_m \mathbf{K}^{(m)} = \sum_m \int_{V^{(m)}} \mathbf{B}^{(m)T}\mathbf{E}^{(m)}\mathbf{B}^{(m)} dV^{(m)} \tag{2.4}$$

where the integral goes over the volume $V^{(m)}$ of the element m. This approach to computing the stiffness matrix is known as the *direct stiffness method*. The displacement interpolation matrix and strain-displacement interpolation matrix are constructed for each finite element. Their calculation depends on the displacement interpolation function seen in equation 2.1. Fortunately, these functions have a well defined formulation which depends on the degrees of freedom (nodal point connections) of the element. This formulation, along with numerical integration techinques for calculating the stiffness matrix can be found in chapter 5 of [1]. The elasticity matrix relates stress and strain in the material. Its form depends on the dimensionality of the element. In a one dimensional element the elasticity matrix is a scalar known as *Young's Modulus*. The value of Young's Modulus describes the elastic properties of materials and is computed as the constant ratio between stress and strain in the material.

Equation 2.3 describes a static equilibrium at a specific point in time. If the loads are applied rapidly then element inertial forces and energy damping forces must be considered as well. Equation 2.5 gives the dynamic form of the FEM equilibrium equation where $\dot{\mathbf{U}}$, $\ddot{\mathbf{U}}$ are the first and second time derivative of $\mathbf{U}$ respectively.

$$\mathbf{M}\ddot{\mathbf{U}} + \mathbf{C}\dot{\mathbf{U}} + \mathbf{K}\mathbf{U} = \mathbf{R} \tag{2.5}$$

Here $\mathbf{M}$ is the mass matrix and $\mathbf{C}$ the damping matrix. This differential equation is often referred to as the FEM governing equation. To solve for the unknown nodal displacements $\mathbf{U}$ standard techniques for solving differential equations can be used.

Modal Analysis

Solving the dynamic equilibrium equation 2.5 by direct integration is very costly [1]. However, it is possible to diagonalize the system of equations by changing the finite element displacement basis to a generalized displacement basis Φ as

$$\mathbf{U} = \Phi\tilde{\mathbf{U}} \tag{2.6}$$

The advantage of the new generalized basis can be seen when equation 2.5 is premultiplied from the left by Φ^T to give

$$\Phi^T\mathbf{M}\Phi\ddot{\tilde{\mathbf{U}}} + \Phi^T\mathbf{C}\Phi\dot{\tilde{\mathbf{U}}} + \Phi^T\mathbf{K}\Phi^T\tilde{\mathbf{U}} = \Phi^T\mathbf{R} \tag{2.7}$$

The above equation can be diagonalized if Φ is chosen as consisting of the n eigenvectors of the eigensolutions (ω_1^2, ϕ_1), ... , (ω_n^2, ϕ_n) which solve the eigenproblem

$$\mathbf{K}\Phi = \Omega^2\mathbf{M}\Phi \tag{2.8}$$

so that

$$\Phi^T\mathbf{K}\Phi = \Omega^2 \tag{2.9}$$

$$\Phi^T\mathbf{M}\Phi = \mathbf{I} \tag{2.10}$$

These eigenvectors are the free vibrational modes of the equilibrium equation. Choosing Φ in this manner transforms equation 2.7 into

$$\ddot{\tilde{\mathbf{U}}} + \tilde{\mathbf{C}}\dot{\tilde{\mathbf{U}}} + \Omega^2\tilde{\mathbf{U}} = \tilde{\mathbf{R}} \tag{2.11}$$

Under the assumption that the transformed damping matrix $\tilde{\mathbf{C}}$ is diagonal as well the above equation 2.11 is diagonalized since Ω^2 is a diagonal matrix. Diagonalizing decouples the individual components of $\mathbf{U}$. This means that we obtain a separate differential equation for each component of $\mathbf{U}$ and as such it is not necessary to compute the inverse of the $\mathbf{K}$. Computing this inverse is costly and may not always be possible if $\mathbf{U}$ is singular. Thus, the diagonalized equation 2.11 can be solved for the displacements $\tilde{\mathbf{U}}$ either in closed form or integrated numerically in fewer steps.

Recovering shape with FEM

Through Modal analysis it is possible to easily generate displacements of an arbitrary object given a stiffness matrix $\mathbf{K}$. These displacements deform the object. Since the matrix Φ encodes the free vibration modes, it is possible to generate a new face by taking a vector of weights $\mathbf{u}$ and calculating a deformed instance of the object class by adding the displacements to the mean shape $\bar{\mathbf{x}}$ using

$$\mathbf{x} = \bar{\mathbf{x}} + \Phi\mathbf{u} \tag{2.12}$$

Equation 2.12 synthesises a new object from a vector of parameters $\mathbf{u}$. It is therefore possible to fit a model of a class of objects with a known stiffness function to a canditate object in an image by solving an optimization problem which locates the parameters $\mathbf{u}$.

Alternatively, it is also possible to use the equilibrium equation 2.3 to recover the shape of the candidate object. Assuming that the positions of the nodal points are found in the image and that their 3D position is recovered then the load vector $\mathbf{R}$ can be calculated as

$$[r_{3k}, r_{3k+1}, r_{3k+2}] = [x_k^w, y_k^w, z_k^w] - [x_k, y_k, z_k] \tag{2.13}$$

where x_k^w, y_k^w and z_k^w is the position obtained from the image of the nodal point with index k. The rest position of this nodal point in the model is given by x_k, y_k and z_k. This approach to constructing the load vector $\mathbf{R}$ is analogous to attaching springs between the nodal point measurements and their corresponding points in the model. Given the load vector the goal is now to solve the previously seen equilibrium equation

$$\mathbf{KU} = \mathbf{R} \tag{2.14}$$

To solve for the displacements $\mathbf{U}$ the stiffness matrix $\mathbf{K}$ needs to be inverted. This is often a very costly operation which can be avoided by diagonalizing the equilibrium equation. This is done using modal analysis with the generalized basis from equation 2.8.

Combining Statistical and FEM modes

2.1.2 Satistical Models

A *shape model* describes the boundaries of an object. figure. For instance, a shape model of a face will denote the location and measures of the defining contours of a face. To locate an object in an image the shape model must be able to account for the following

PCA and Eigenfaces

In 1991, Turk and Pentland [15] pioneered a face recognition approach based on the mathematical technique called principal component analysis (PCA). Their face recognition scheme uses a data set of images to learn what they call *eigenfaces*. Expressed in mathematical terms - the eigenfaces are principal components of the 2D image space. They represent the vectors of the 2D data set that are responsible for any significant variation. As such these vectors are the eigenvectors of the covariance matrix of the face images. To obtain the eigenfaces it is necessary to compute the SVD decomposition of this covariance matrix. Any individual face from the training data set can then be exactly represented as a linear combination all the eigenfaces.

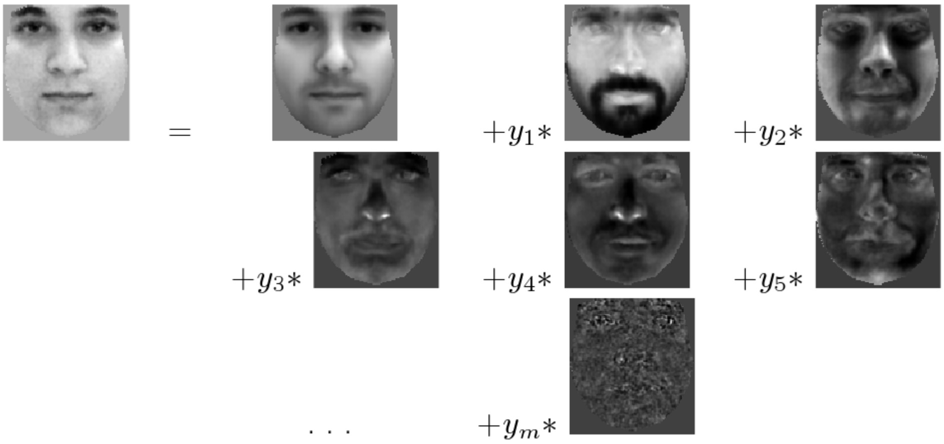


Figure 2.1: Using the eigenfaces we can represent an image as a linear combination of the eigenfaces. Taken from [13]

The weight of each eigenface in the linear combination is computed from the projection of the face image onto this eigenface. To recognize a face we first calculate these weights. Then we calculate the Euclidean distance of the vector of weights to the weights of faces from our training data set. The face from the data set with weights which are closest to the new image's weights is chosen as a match.

In an image of 256 by 256 we have a 65,536 dimensional vector that represents the face image. This means we would require 65,536 eigenfaces to be able to exactly represent every face. However, images of faces are very similar in their configuration which means that the underlying principal subspace of faces has a lower dimension than 65,536. Once the eigenfaces which span this principal subspace are found we can effectively encode a 65,536 dimensional vector using a vector of much smaller dimensions.

The main advantage of the eigenfaces is that we can approximate a face very well by a linear combination of only the eigenfaces that account for the largest variance. The set of M eigenfaces that represent the M largest variances is called the *face space*.

The drawback of a PCA based model is that the recognition rate drops significantly once independent sources of variation are introduced. Turk and Pentland noted that the eigenfaces approach has issues with variations in lighting, head size, head orientation or faces exhibiting expressions [15]. Likewise, when faces are partially occluded in images it causes difficulties to the technique.

The eigenfaces approach is based heavily information and coding theory [15]. With the PCA it is possible to encapsulate a face image using low dimensional vector. As such PCA and eigenfaces are often used to reduce dimensionality in more sophisticated modelling approaches.

2.1.3 Statistical Models of Appearance

In their paper Statistical Models of Appearance [5] Cootes and Taylor describe modelling approaches that consider texture variance, shape variance and the correlations between these two variances. The models they introduce are well suited for highly variables structures such as faces or internal organs. Using the shape and texture models, Cootes and Taylor show that it is possible to construct Active Shape and Active Appearance models that successfully locate shapes and even faces in images.

Cootes and Taylor separate the shape and the information of a target image into two distinct vectors - the shape parameter vector b_s and the texture represented by the grey-level vector b_g (here the image is cray-scaled). Thus texture can be manipulated and investigated independent of the shape. They then perform PCA on both vectors, which allows them to express grey-levels and shape as functions of a parameter vector c as follows

$$x = \bar{x} + Q_s c \tag{2.15}$$

$$g = \bar{g} + Q_g c \tag{2.16}$$

Where the shape vector x and the grey-level (texture) vector g are functions of the mean shape $\bar{x}$, the mean texture $\bar{g}$ and the matrices Q_s and Q_g which describe the modes of variation learned from the training data set [5]. To reconstruct a new face using the model we first determine the b_s and b_g parameters from the new face image. These parameters are then used to calculate the c. Finally, the reconstruction is obtained using the formulas 2.15 and 2.16.

Active Shape Model

The active shape model (ASM) proposed by Codee's and Taylor [5] is based on the statistical models of shape and texture. The ASM enables us to locate a shape in an image using an iterative procedure as shown in figure 2.2. To search for objects in an image using the ASM, we first place the shape model, which we obtained from the training data, into the centre of the image. In the next iterations a texture profile of k neighbouring points around each point of the shape model is taken and compared with the texture profile of this point obtained when training the model. Then the point of the shape model is moved to make the difference between the two profiles smaller. This process is repeated until convergence is reached.

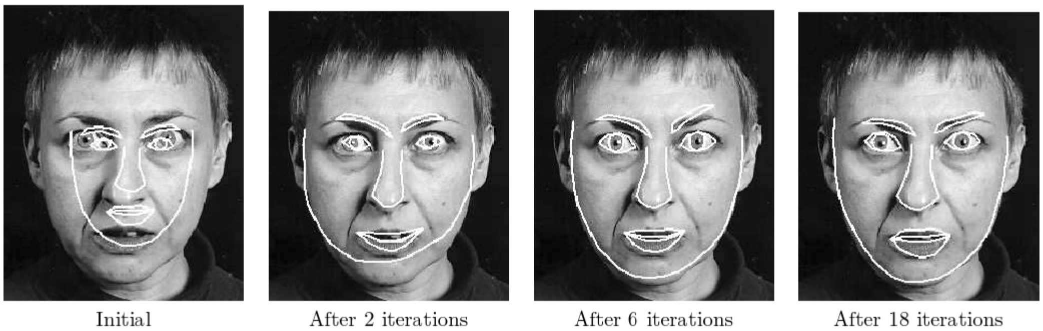

Initial After 2 iterations After 6 iterations After 18 iterations

Figure 2.2: Active Shape Model iterations shown on on finding the shape of a previously unseen face. Taken from [4]

The ASM moves the point only based on the texture profile of a local neighbourhood of this point. The obvious problem with this approach is that as it is a local search technique and thus it depends too much on the choice of the starting point. If the target image is not centered on the face and the algorithm begins in the centre of the image, then it can possibly converge on incorrect shapes such as the nose as can be seen in figure 2.3. This is due to the fact that as a local search method, the ASM cannot escape a local minimum.

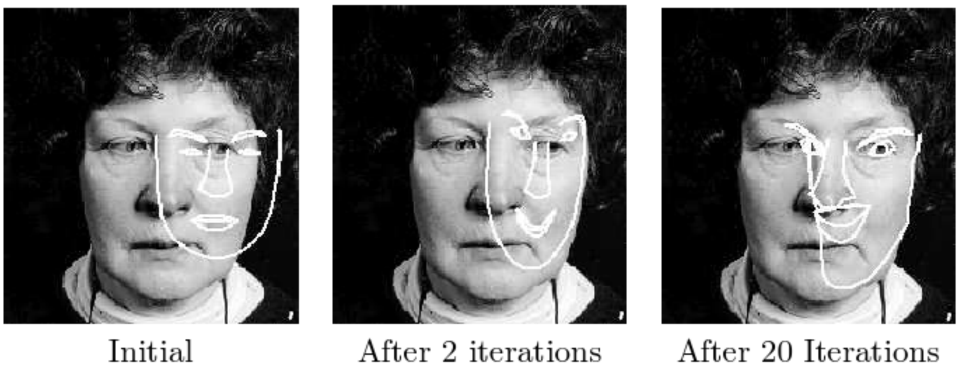


Figure 2.3: Incorrect convergence of the ASM when the search is initialised incorrectly. This can happen when the face is not centered in the image or when the search does not begin in the centre of the image. Taken from [4]

An interesting improvement of the technique is proposed by Cootes et al [5]. To improve the efficiency and the robustness of ASM, Cootes suggests to perform several searches at different resolutions. First a coarse search is performed on the image with a low resolution until the search converges. The resulting configuration of the shape model points from this coarse search then serves as the starting point for a search in the same image with a better resolution. The advantage of adopting this coarse to fine search approach is that it makes the search less susceptible to converging on incorrect shapes. When searching at a lower resolutions it will be easier to find the outlines of the face. Once the search is restarted with a higher resolution shapes like the mouth and the nose can be identified.

Active Appearance Model

Modelling faces using the Active Appearance Model (AAM) is a technique explored by Cootes and Taylor [5]. AMM takes advantage of the Active Shape Model which allows us to locate interesting shapes in a new image. The set of points defining this shape are called landmark or feature points. The AMM enhances the AMS search by also considering the texture information.

The AMM search attempts to minimise the difference between the texture (the grey-levels) of the image we are searching and the grey-levels of the image generated with the current model parameters. The difference vector that is minimised is defined as:

$$\delta I = I_i - I_m \tag{2.17}$$

where I_m is the vector of the grey-levels generated with the current model parameters and I_i is a vector containing the grey-levels of the image we are searching in.

To simplify this optimisation problem, Cootes and Taylor learn the relationship between the difference vector δI and the error in the model parameters. Learning this relationship makes it possible to correctly improve the model for a given measured error in an iterative procedure.

The AAM iterations are depicted in figure 2.4. The process attempts change model parameters to fit it to an unseen face. This fitting is done by minimising the differences between pixels at the landmark points.

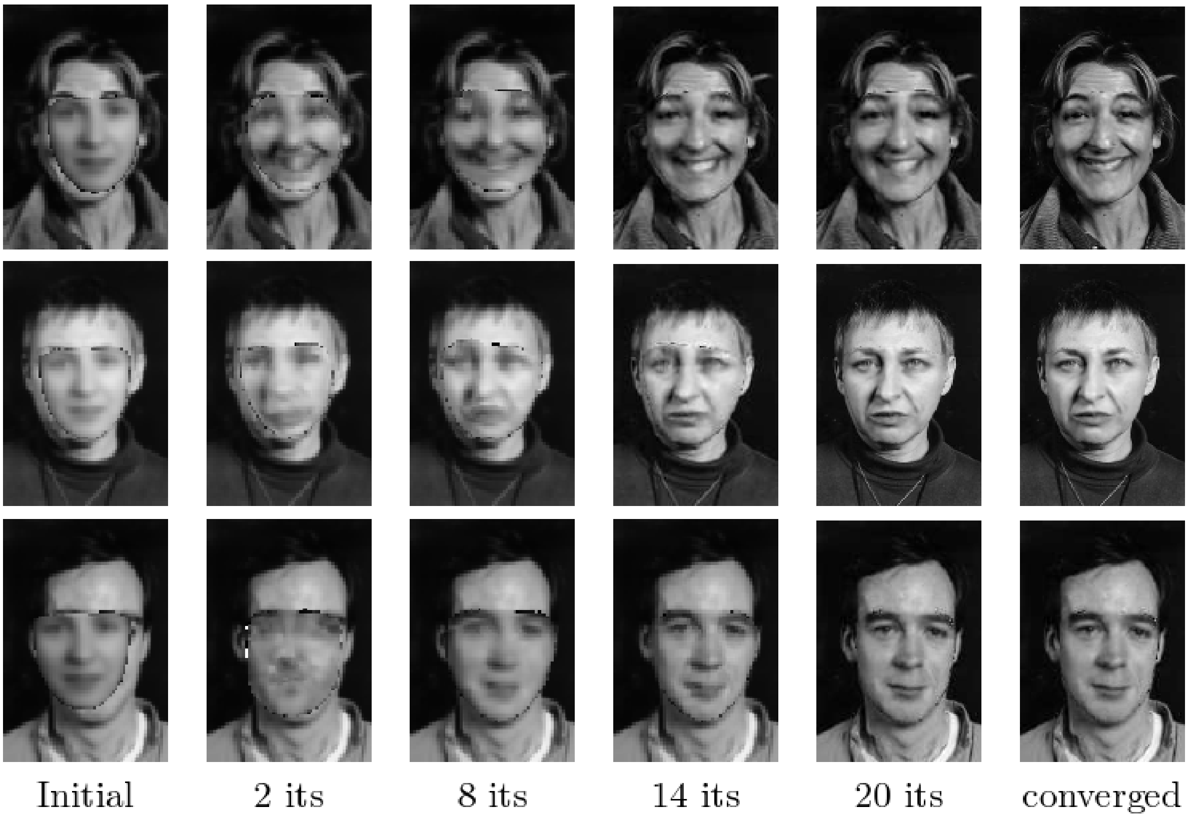


Figure 2.4: Active Appearance Model iterations shown when searching for previously unseen faces. Taken from [5]

The drawback of the AMM search is that it is required of the user to specify landmark points on the target image. This means the technique is not well suited for applications where the model would have to be matched to a large number of faces. To combat this problem, a number of automatic 2D and 3D landmark location solutions can be used [5].

Another problem of both the ASM and the AMM lies in the fact that it does not distinguish or model the different variation sources. The variation caused by identity and the variation caused by expression are not perceived as two fundamentally different sources. Thus, it is impossible to only alter one variational source while leaving the other source constant. This makes the models unsuitable for expression transfer.

2.1.4 Tensor-Based Model

The possibility of using a multilinear approach to construct face models for face transfer was explored by Vlasic, Brand, Pfister and Popovic [16]. The group successfully managed to implemented a face transfer application based on multilinear face model which allowed for expressions and even for speech related movements to be transferred between video-recordings of two different subjects. The multilinear model was estimated from geometric variations in 3D face scans that Vlasic and his group collected. Vlasic et al utilised two tensor models with different dimensionality. Their first model was a lower dimensional bilinear model that organised the data into three groups of vertexes, identity and expression. Their higher dimensional model organised faces into groups of expressions, vertexes, identities and visemes. The parameters for the multilinear face model which were used to generate new expressions were extracted from the video input using an optical flow algorithm.

To construct a bilinear face model it is necessary to separate the faces from the database into groups of expressions and identities in order to organise them into the tensor. The tensor based approach allows us to organise the data in a way that makes for easier manipulation with a transformation matrix. The groups are aligned in the tensor along the modes as shown in figure 2.5. In this example the vertexes change along the mode-1 space, the identity changes along the mode-2 space and the expressions change along the mode-3 space.

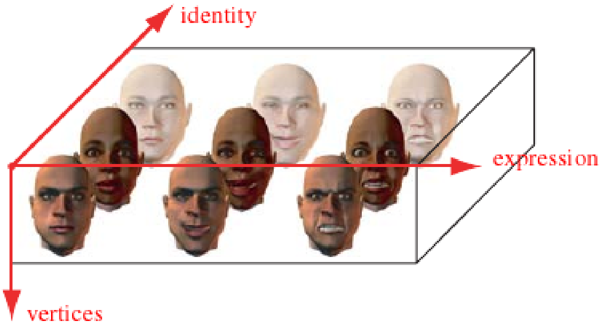


Figure 2.5: Multilinear face model showing how different attributes change along the modes. Taken from [16]

A mode spaces of a tensor can be altered independently of the other mode spaces by a linear transformations called *mode-n product*. The mode-n product is defined between a matrix $\mathbf{M}$ and a tensor $\mathscr{T}$ and is written as $\mathscr{T} \times_n \mathbf{M}$. The mode-n product transforms the vectors in the n-th mode of the tensor by the matrix $\mathbf{M}$. Thus, it is possible to separately transform the expressions in mode-3 and the identity in mode-2. This separability of the model is convenient for the face transfer application.

The *mode-n singular value decomposition (SVD)* is a linear transformation of a tensor that produces a *core tensor*. The core tensor is analogous to a diagonal matrix of eigenvalues used in PCA. Eigenvalues can be seen as measures of the variance along the corresponding eigenvector directions. In the core tensor the variance decreases from first element of the core tensor to the last, which makes it possible to reduce the dimensionality of the data set by truncating the core tensor. Therefore, it is possible to approximate the original tensor using a reduced core tensor. The approximation of a tensor $\mathscr{T}$ using a reduced core tensor $\mathscr{C}_{reduced}$ is derived from the mode-n SVD and thus defined as:

$$\mathscr{T} \approx \mathscr{C}_{reduced} \times_1 \hat{\mathbf{U}}_1 \times_2 \hat{\mathbf{U}}_2 \times_3 \hat{\mathbf{U}}_3 \cdots \times_N \hat{\mathbf{U}}_N \tag{2.18}$$

The matrices $\hat{\mathbf{U}}_i$ are truncated versions of the eigenvector matrices for the corresponding mode space.

Vlasic et al constructed a multilinear face model using mode-n SVD and decomposing the organised tensor into the matrices of eigenvectors of all modes except for the mode-1 space which holds the vertexes. Using equation 2.18 the multilinear face model then becomes:

$$\mathscr{T} \approx \mathscr{M} \times_2 \hat{\mathbf{U}}_2 \times_3 \hat{\mathbf{U}}_3 \cdots \times_N \hat{\mathbf{U}}_N \tag{2.19}$$

The tensor $\mathscr{M}$ is called the *multilinear model*. Multiplying the multilinear model $\mathscr{M}$ with a linear combination of rows from the truncated eigenvectors then gives us exactly one original face [16].

The described tensor-based model is similar to what we saw in the eigenfaces PCA approach in figure 2.1. However, the tensor-based approach has the advantage that we can manipulate the different sources of variance (i.e. the identity, the expressions) separately. This advantage outweighs the additional computational complexity carried by the high order SVD and makes this model suitable for the task of expression transfer.

The drawback of the method described by Vlasic et al is that it requires the tensor to be fully populated. This means that we require all expressions to be performed by all subjects to have a full tensor. This raises issues when some data is corrupted or lost during data gathering. There are two ways of coping with this problem. The first approach is to use a statistical method to model the data. Or the missing data can be estimated from the current data.

2.1.5 Tensor-based Statistical Discriminant Method

The tensor-based statistical discriminant methods (SDM) was successfully used by Minoi and Gillies [10, 9] to synthesise expressions and to neutralise faces displaying an expression. The SDM is based on Fischer's linear discriminant analysis (LDA) [6]. The LDA differs from the standard PCA eigenfaces approach in that it seeks to separate data into distinct classes. The way this is done is by projecting the data into a lower dimensional subspace that maximises the between class separability and minimises the within class variance.

In LDA we first separate the training data set into a number of groups. Then we look for the between class scatter matrix S_b and the within class scatter matrix S_w. The matrix Φ_{lda} that defines the projection onto the desired low dimensional space is defined as:

$$\Phi_{lda} = \arg\max_{\Phi} \frac{|\Phi^T S_b \Phi|}{|\Phi^T S_w \Phi|} \tag{2.20}$$

In equation 2.20 the ratio of the determinant of the between class separability and the determinant of the within class variability is maximised. The maximum of this equation is the optimal projection.

However, the LDA approach has difficulties when the training data set is small in comparison to the dimensions of the image. This is called the *small size problem* and it causes the scatter matrices to be singular due to not having a full rank.To overcome the small size problem the statistical discriminant method can be used.

The SDM approach consists of two stages. In the first stage the PCA and LDA are used to reduce the dimensionality and find the discriminant directions of the classes. The PCA helps to overcome the small size problem common to stand-alone LDA. In the second stage the most discriminant vectors of our classes are projected back into the original high dimensional space. This back projection will give us a vector for every class in the high dimensional space. These vectors allow us to synthesise an expression for a new face image by moving the the surface points of this new image in the direction of one of the vectors.

The SDM technique can be extended to tensor models by expanding the mode responsible for expressions into a number of modes representing individual expression classes, thereby effectively increasing the dimension of the tensor. The expanded tensor model still retains the quality of independence along the modes of variance. However, due to the expansion the SDM can be applied to the tensor if we flatten the sub-tensors into matrices.

2.1.6 Morphable Model

Blanz et al use PCA to construct a statistical model of 3D face shape and texture [2], [3]. The orthogonal matrix of basis vectors is extracted from a database of 3D faces with texture. This basis spans a vector space which Blanz refers to as the *Morhable Model*.

The faces in the database are encode as shape vectors with S_i to being the vector representing the 3D shape of a human face, stored as the x, y, z-coordinates of all the vertices of that face. So

$$\mathbf{S_i} = (x_1, y_1, z_1, x_2, ..., x_n, y_n, z_n)^T \tag{2.21}$$

Similarly, the texture vectors are defined as

$$\mathbf{T_i} = (R_1, G_1, B_1, R_2, ..., R_n, G_n, B_n)^T \tag{2.22}$$

The idea behind the morhable model is that any linear combination of the $\mathbf{S_i}$ and $\mathbf{T_i}$ constitues a realistic face. The full morphable model is the linear combination of the examples given by the formula

$$\mathbf{S} = \sum_{i=1}^{m} a_i \mathbf{S_i} \qquad \mathbf{T} = \sum_{i=1}^{m} b_i \mathbf{T_i} \tag{2.23}$$

To allow the morphable model to fit well to the source image, dense point-to-point correspondence must be ensured on the example images, so that important features such as the tip of the nose are correctly matched up in the examples. Then the fitting process matches the features in the source image to those in the examples, in order to calculate the correct combinations of the basis vectors.

Principal Component Analysis (PCA) is performed on the dataset of examples to produce orthogonal basis vectors that can be used in the morphable model instead of the examples directly.

$$\mathbf{S} = \bar{\mathbf{s}} + \sum_{i=1}^{m} \alpha_i.\mathbf{s_i} \qquad \mathbf{T} = \bar{\mathbf{t}} + \sum_{i=1}^{m} \beta_i.\mathbf{t_i} \tag{2.24}$$

This process identifies vectors which represent the most variance in the examples, and so contain the most information about the images. This means any image can now be represented by the linear combination of a smaller number of these components than would be required of the original examples. A limit can then be set on the number of components used in the model, via the value of m. This is generally set to 149, as the components beyond this contain mostly noise.

However, to to fit the Morphable Model of 3D faces to an image, the user has to manually select about 7 feature points.

3D Reconstruction algorithm

To be able to exchange the face we need the α_i, β_i parameters of the Morphable Model. We also need 22 rendering parameters ρ to be able to apply standard graphics procedures to correctly illuminate and position the 3D source face into the target image. The ρ parameters are:

- 3D rotation (3 angles)
- 3D translation (3 dimensions)
- focal length of the camera
- angle of directed light (2 parameters)
- intensity of directed light (3 colours)
- intensity of ambient light (3 colours)
- color contrast
- gain in each color channel (3 channels)
- offset in each color channel

We apply the reconstruction algorithm to estimate the paramters for both images and then use these parameters to compute the colour image $I_{model}(x, y)$ of the face we are reconstructing.

The parameters are all estimated in an analysis-by-synthesis stage in which we attempt to minimise the difference between the synthetic image I_{model} and the original colour image I_{input}. We measure the quality of this minimisation based on a sum of squared error

$$E_l = \sum_x \sum_y \sum_{c \in \{r,g,b\}} (I_{c,input}(x, y) - I_{c,model}(x, y))^2 \tag{2.25}$$

To this cost error we add another term E_F that measures the plausibility of the ρ parameters based on a user provided set of 2D feature points. The resulting cost function that we minimize during the fitting stage is

$$E = \frac{1}{\sigma_l^2} E_l + \frac{1}{\sigma_F^2} E_F + \sum_i \frac{\alpha_i^2}{\sigma_{S,i}^2} + \sum_i \frac{\beta_i^2}{\sigma_{T,i}^2} + \sum_i \frac{(\rho_i - \bar{\rho}_i)^2}{\sigma_{R,i}^2} \tag{2.26}$$

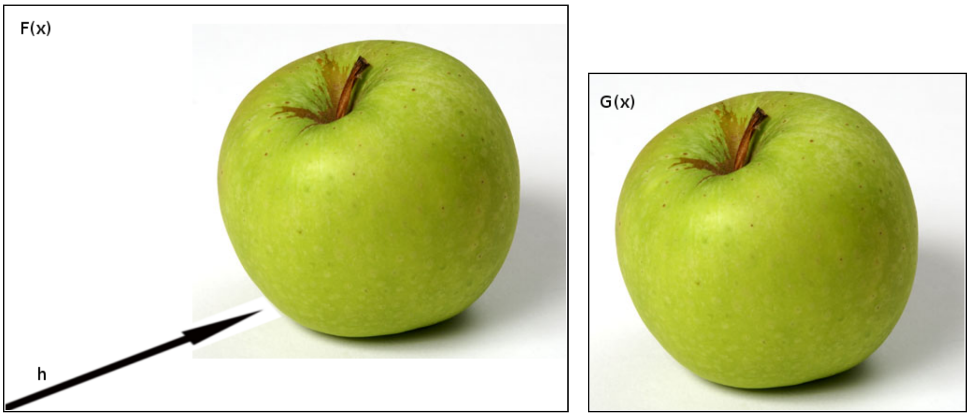
where our standard deviations σ are taken from the PCA. This cost function is minimised using a Stochastic Newton Algorithm to obtain a plausible result for the parameters.

However, there are skin features like moles or scars that we cannot compute from the linear combinations of the textures T_i. Therefore we additionally extract the texture of the person's face using an illumination-corrected texture extraction algorithm.

2.2 Model-based tracking

2.2.1 Kanade-Lucas-Tomasi Feature Tracker

The problem of registering a displaced image can be formalised by assuming we are given two functions $F(x)$ and $G(x)$. These function take a vector x as input and output the pixel intensity at x. Thus, we can think of think of these functions as representing images. The goal in image registration is to find a disparity vector h which minimises a measure of difference between $F(x+h)$ and $G(x)$ for a given region of interest R [7]. This region of interest R is the feature which we are tracking. The image registration problem is shown below:



(a) F(x) (b) G(x)

Figure 2.6: The image registration problem. The task is to find the disparity vector h.

The measure of difference most often used is the L_2 norm, which is defined as follows for the registration problem:

$$L_2\text{norm} = (\sum_{x \in R}[F(x+h) - G(x)]^2)^{\frac{1}{2}} \tag{2.27}$$

The naive way to locate the h that minimises the L_2 norm would be to iterate through all the possible values of h and measure the L_2 norm. A better approach was proposed by Kanade and Lucas [7]. Their algorithm specifies the order in which possible values of h will be explored. The procedure iteratively improves the guess for h by considering the spatial intensity gradients at each point x in the image. This means that the Kanade-Lucas-Tomas feature tracker locally searches for the best disparity vector h by using the information about the gradient at all points in the image. The estimation of the h and the convergence of the algorithm is further improved by weighing the gradients using a second derivation approximation of the image [7]. The weights are thus defined as:

$$w(x) = \frac{1}{|G'(x) - F'(x)|} \tag{2.28}$$

Where the w(x) was derived from an approximation to the second derivative with h factored out since it is a constant in all the weights.

The iterative scheme for the Kanade-Lucas-Tomasi algorithm is:

$$h_0 = 0 \tag{2.29}$$

$$h_{k+1} = h_k + \frac{\sum_x w(x)F'(x+h)[G(x) - F(x+h_k)]}{\sum_x w(x)F'(x+h_k)^2} \tag{2.30}$$

where w(x) is calculated using equation 2.28.

To further improve the technique Kanade et al suggest to perform the algorithm on several resolutions, using the result from the coarser resolution as starting h_0 of the algorithm with a finer resolution.

The algorithm can also successfully compute the h even if the region of interest has been rotated or scaled. If the image has been rotated or scaled we express the matching problem as $G(x) = F(xA + h)$ where A is a linear transformation matrix. This transformation allows us to apply the algorithm.

The Kanade-Lucas-Tomasi feature tracker has been successfully applied to tracking facial feature points [16]. Vlasic et al localised feature points in the initial frame manually and then used the Kanade-Lucas-Tomasi feature tracker to obtain the displacements between pairs of frames. From these displacements they were able to derive parameters for their tensor-based multilinear model. This allowed them to transfer expressions to another video performance.

As a local iterative search method, the Kanade-Lucas-Tomasi feature tracker's performance depends heavily on the initial guess of the h. If the initial guess is too far from the region of interest then the algorithm does not perform well, since the scheme was derived using local approximations of functions. These local approximations will not hold for points far from the region of interest.

2.3 Optimization

2.3.1 Nelder Mead Downhill Simplex

The downhill simplex is a local optimization method that does not require gradients to identify local extrema. With the downhill simplex method a local optimum is found by means of a sequence of fitness function evaluations.

The downhill simplex method was coined by Nelder and Mead in 1965 and has since proven itself to be comparable in efficiency to the more popular gradient based optimization methods. The method was designed for the optimization of multidimensional, unconstrained functions that have either no gradients or when the gradient exist only for portions of the search space sucha as in the case of discontinuous functions [14]. However, the drawback of the method is that many fitness function evaluations are required. Therefore, when the computational complexity of the fitness function is very high, other optimization methods need to be considered instead of the Nelder Mead downhill simplex [12].

The algorithm is based on the idea of isolating the minimum by geometrically transforming a *simplex*. The simplex is a convex hull of $N + 1$ vertices, where N is the underlying problem's dimension. In a two dimensional space this simplex would therefore be a triangle, in three dimensions a tetrahedron. The algorithm is initialized so that the simplex encloses a portion of the search space and the goal is to move the simplex along the search space surface and deform so that all of its vertices converge on the local optimum. This is achieved with the help of geometric transformations of the simplex.

The process of transforming a multidimensional simplex, in order to isolate the minimum, is somewhat analogous to bracketing a minimum in a one dimensional search space. The one dimensional search space will have peaks and valleys in places of local optima. The simplex, which is a line in one dimensional space, makes it's way downhill through this search space, in search of a minimum by means of shrinking and stretching. When the simplex finds a local minimum, it shrinks itself to contain only the minimum and the algorithm terminates.

The behavior of the simplex during the algorithm parallels the expanding and collapsing movements of the amoeba organism. The Nelder Mead downhill simplex is in some publications referred to as the *amoeba* method due to this similarity but also to distinguish it from Dantzig's simplex method for linear programming [12].

The Downhill Simplex Algorithm

To initialize the downhill simplex algorithm we need a nonlinear fitness function $f : \mathbb{R}^N \rightarrow \mathbb{R}$ and an initial point P_0. The simplex will be a $N + 1$ dimensional convex hull. The first vertex of the convex hull is the initial point. The remaining N vertices P_i are derived from the initial point. The shape of the simplex defines the way in which the points P_i are calculated[14].

The simplex shape can be one of the following:

- The simplex can have a regular shape where all sides are equally long. It is up to the user to pick this length.
- The simplex can be right angled in which case the vertices P_i are calculated according to formula 2.31.

$$P_i = P_0 + \lambda e_i \tag{2.31}$$

  where e_i are unit vectors for the N dimensions and λ is a constant. This constant influences the size of the simplex and represents a guess of the problem's characteristic scale length [12].

After the initialization phase, three crucial steps are repeated until the simplex has encountered the local minimum. These steps are based around moving the vertex of the simplex with the largest fitness function largest value to a new point where the value will be smaller.

1. The first step is to sort the vertexes x_i of the simplex from worst to best, where h is the index of the worst vertex, s the second worst index and l the best index.

2. Then the *centroid* of the best side is calculated according to formula 2.32.

$$c = \frac{1}{N} \sum_{j \neq h} x_j \tag{2.32}$$

3. In the final step, the centroid is used to geometrically transform the simplex in order to move the current worst vertex to a better position. To achieve this, the algorithm seeks a replacement point for x_h on the line that connects the worst index x_h and the centroid c. Three different points are then compared and the one with the best fitness value is chosen as the replacement point. These three candidates are obtained using reflection (formula 2.33), expansion (formula 2.34) and either inside or outside contraction (formula 2.35).

$$x_r = c + \alpha(c - x_h) \tag{2.33}$$

$$x_e = c + \gamma(x_r - c) \tag{2.34}$$

$$x_c = \begin{cases} c + \beta(x_r - c) & if\, x_h \leq x_r \\ c + \beta(x_h - c) & else \end{cases} \tag{2.35}$$

In case neither of these three new replacement point candidates has a better fitness value than the worst vertex x_h, then the entire simplex is shrunk towards the best vertex x_l. In this case N new vertices will be computed as follows:

$$x_j = x_l + \delta(x_j - x_l) \;\; j = 0, \ldots, N \, \wedge \, j \neq l \tag{2.36}$$

The geometric implications of the transformations reflection, expansion, contraction and shrinking are depicted in figure 2.7.

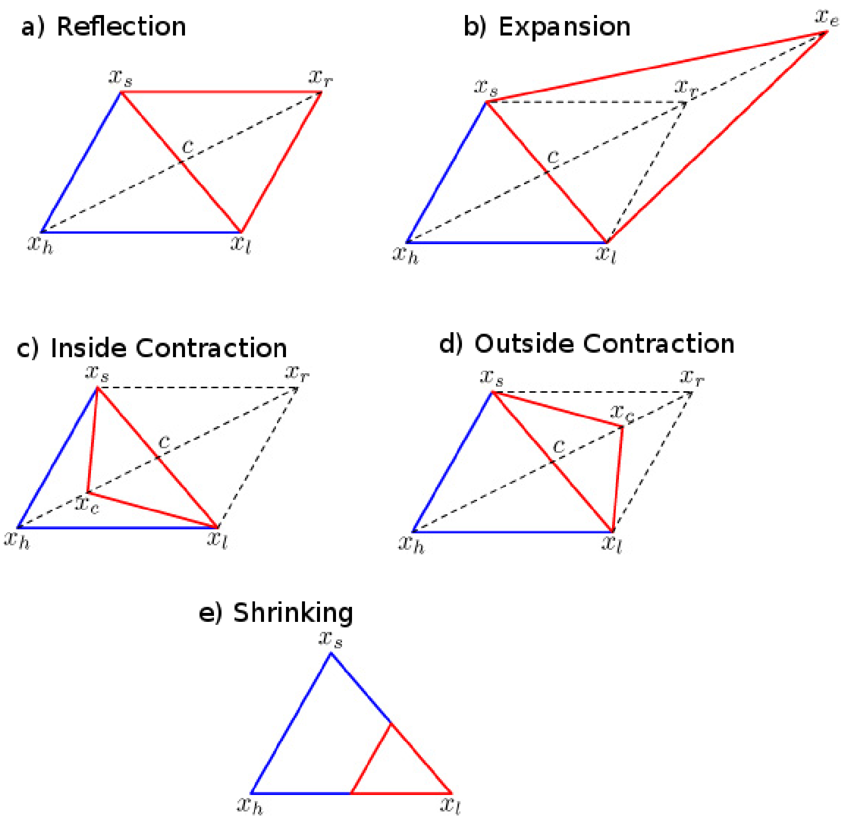


Figure 2.7: Geometric interpretations of the simplex transformations. Taken from [14]

The transformations are controlled by four parameters α for reflection, β for contraction, γ for expansion and δ for shrinking. Most implementations use the standard values $\alpha = 1$, $\beta = \frac{1}{2}$, $\gamma = 2$ and a shrinking by a half with $\delta = \frac{1}{2}$ [12, 14].

Finally, since the algorithm should terminate in finite time, it is necessary to establish a termination criterion. If the execution of the three aforementioned steps is considered one cycle of the algorithm, then possible termination criteria include terminating when the vector distance moved in the cycle was smaller than a constant tolerance *tol*, or when the difference between the fitness value of the newly obtained best and the old best is no larger than a tolerance $ftol$. Since either of these criteria could occur in a single anomalous step, restarts of the downhill algorithm are also sometimes utilized [12].

Bibliography

[1] Klaus-Juergen Bathe. *Finite Element Procedures.* Prentice Hall, 1996.

[2] V. Blanz, C. Basso, T. Poggio, and T. Vetter. Reanimating faces in images and video. *EUROGRAPHICS 2003*, 22(3), 2003.

[3] V. Blanz, K. Scherbaum, T. Vetter, and H.P Seidel. Exchanging faces in images. *EUROGRAPHICS 2004*, 23(3), 2004.

[4] T.F. Cootes and C.J. Taylor. Statistical models of appearance for medical image analysis and computer vision. *Proc. SPIE Medical Imaging*, 2001.

[5] T.F. Cootes and C.J. Taylor. Statistical models of appearance for computer vision. mar 2004.

[6] D.F. Gillies. Intelligent data and probabalistic inference lecture notes. *Imperial College London*, 2009.

[7] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[8] Dimitri Metaxas and Demetri Terzopoulos. Shape and nonrigid motion estimation through physics-based synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):580–591, 1993.

[9] Jacey-Lynn Minoi. *Geometric Expression Invariant 3D Face Recognition using Statistical Discriminant Models.* Phd dissertation, Imperial College London.

[10] J.L. Minoi and D.F. Gillies. 3d face and facial expression recognition using tensor-based discriminant analysis methods.

[11] Alex Pentland and Stan Sclaroff. Closed-form solutions for physically based shape modelling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):715–726, 1991.

[12] W.H. Press, S.A. Teukolsky, W.T. Vetterlin, and B.P. Flannery. *Numerical Recepies in C: The Art of Scientific Computing 2nd Edition.* Cambridge University Press, 1992.

[13] M. Reiter. Neural computation ws 2006/2007 lecture notes. *Vienna University of Technology*, 2006.

[14] J. Nelder S. Singer. Nelder-Mead algorithm. 2009. [http://www.scholarpedia.org/article/Nelder-Mead_algorithm].

[15] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[16] D. Vlasic, M. Brand, H. Pfister, and J. Popovic. Face transfer with multilinear models. *ACM Transactions on Graphics*, 24(3), 2005.