Individual Project
Expression Transfer

Martin Papanek (mp5309)

Abstract

Acknowledgements

Contents

1 Introduction **9**
1.1 Motivation 9
1.2 Contributions 10
1.3 Report Overview 10

2 Background **11**
2.1 Model-based object recognition 11
2.1.1 Shape Model 11
2.1.2 PCA and Eigenfaces 11
2.2 Model-based tracking 13
2.2.1 Nelder Mead Downhill Simplex 13

List of Tables

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 perceive his surroundings, correctly identify the objects and beings around him and act based on this information. Achieving this level of sophisticated, environment aware behaviour 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 utilize the data gleaned from its sensors to find objects in the image. Finally, the machine has to obtain contextual information from the configuration of these objects. Possessing this contextual information can be seen as equivalent to understanding the scene. The computer uses the contextual information to understand what state the environment is in and can then act based on 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. To instruct machines about the properties of the various objects, which could potentially be present in the surroundings, we use *models*. A model describes the expected structure of the object. This, in turn, allows the machine to explain aspects of its sensory input as the occurence of that object. Finding objects by means of locating an instance of a model in the input is called *model-based tracking*.

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 shape or appearence of this dynamic object. The challenge thus lies in constructing an appropriate *deformable model*.

Furthermore, the computer has to be able to find an instance of this model in its surroundings if and only if this object is present. This task is known as *model extraction*.

When extracting deformable models, it is necessary to also estimate what state the model is in. This means we have to not only locate the object but also estimate the values of parameters which govern how this object changes.

Hence, in this paper we are interested in describing convenient dynamical shape and appearence models and effective algorithms for locating these models. For our sensory snapshots, we will focus exclusively on images. Since 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 locate the model in the image.

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.

1.2 Contributions

1.3 Report Overview

2. Background

Using models to characterise a deformable object and locate it in an image remains a very fundamental challenge of computer vision and graphics. In this chapter we will examine various model extraction techniques. Likewise, we will discuss computer vision algorithms that will allow us to locate the parts that make up our objects and observe how the objects change between several images.

2.1 Model-based object recognition

The goal of model-based object recognition is 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. The parameters which determine the orientation are called *pose parameters*.

2.1.1 Shape Model

A *shape model* describes the boundaries of an object. figure 111 For example a shape model of a face which denotes 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

- variations of shape due to deformations of the object
- arbitrary scaling of the object
- arbitrary rotations of the object
- some measure of gaussian input noise tolerance

2.1.2 PCA and Eigenfaces

In 1991, Turk and Pentland [4] 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.

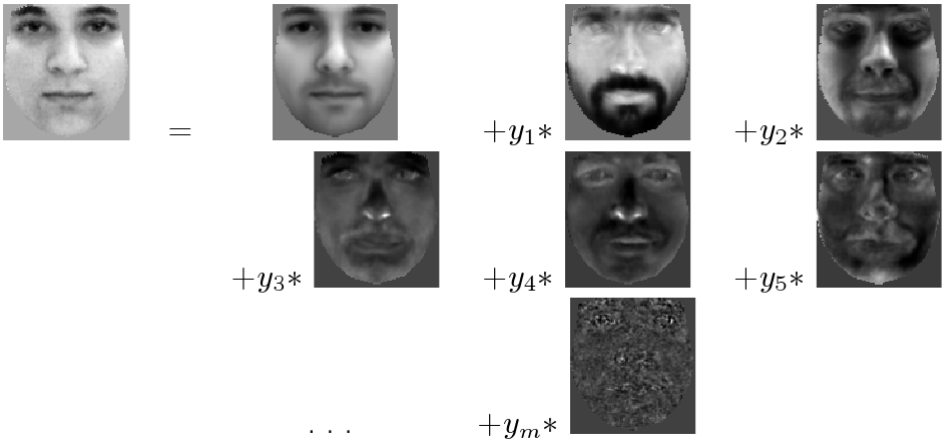

Figure 2.1: Using the eigenfaces we can represent an image as a linear combination of the eigenfaces. Taken from [2]

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 [4]. Likewise, when faces are partially occluded in images it causes difficulties to the technique.

The eigenfaces approach is based heavily information and coding theory [4]. 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.

2.2 Model-based tracking

2.2.1 Nelder Mead Downhill Simplex

The downhill simplex is a local optimization method that does not require gradients and makes use of only fitness function evaluations in finding a local optimum. It was created by Nelder and Mead in 1965 and has since proven itself to display efficiency comparable to that of gradient based optimization methods. The method was engineered for the optimization of multidimensional, unconstrained functions that have either no gradients or when the derivatives exist only for portions of the search space like in the case of discontinuous function [3]. The downhill simplex is also fairly simple to implement, however the drawback is that many function evaluations are required. Therefore, when the computational difficulty of the fitness function is very high, other optimization methods need to be considered instead of the Nelder Mead downhill simplex [1].

The downhill simplex method 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 be a triangle, in three dimensions a tetrahedron. This simplex encloses a portion of the search space and the goal is to move and deform the simplex to have it enclose the minimum only. This is achieved with the help of geometric transformations of the simplex. These transformations, will be discussed more in depth later.

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 simplex makes it's way downhill through the search space, in search of a minimum and when it finds such a local minimum, it shrinks itself to contain only the minimum which is then the result of our local optimization. The behavior of the simplex during the algorithm parallels the movements of the amoeba organism. Due to this and also to distinguish it from the Dantzig's simplex method for linear programming, the downhill simplex is in some publications referred to as the *amoeba* method [1].

The Downhill Simplex Algorithm

Given a nonlinear fitness function $f : \mathbb{R}^N \to \mathbb{R}$ and an initial point P_0, a simplex is constructed from the initial point and N derived points P_i. There are generally two kinds of shapes of simplex that can be picked from and it is this shape that defines the way in which the points P_i are obtained [3]. The simplex can either have a regular shape where all sides are equally long, or the simplex can be right angled in which case the vertices P_i are calculated according to formula 5.1.

$$P_i = P_0 + \lambda e_i \tag{2.1}$$

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 [1].

After the simplex has been initialized, three crucial steps are repeated until the simplex has encountered the local minimum. These steps are based around moving the vertex of the simplex from where the fitness function has the largest value to a point where the value will be smaller. To achieve this, the first step is to order 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. Then the *centroid* of the best side is calculated according to formula 5.2.

$$c = \frac{1}{N} \sum_{j \neq h} x_j \tag{2.2}$$

This centroid is then used in the final step, wherein the simplex is geometrically transformed in order to move the current worst to a better position. At first, a replacement point for x_h is sought on the line that connects the worst index x_h and the centroid c. Three different points are considered as a replacement point and they are obtained using reflection (formula 5.3), expansion (formula 5.4) and either inside or outside contraction (formula 5.5).

$$x_r = c + \alpha(c - x_h) \tag{2.3}$$

$$x_e = c + \gamma(x_r - c) \tag{2.4}$$

$$x_c = \begin{cases} c + \beta(x_r - c) & if\, x_h \leq x_r \\ c + \beta(x_h - c) & else \end{cases} \tag{2.5}$$

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. Meaning, N new vertices will be computed as follows:

$$x_j = x_l + \delta(x_j - x_l) \;\; j = 0, \ldots, N \wedge j \neq l \tag{2.6}$$

The geometric implications of the transformations reflection, expansion, contraction and shrinking are depicted in figure 7.

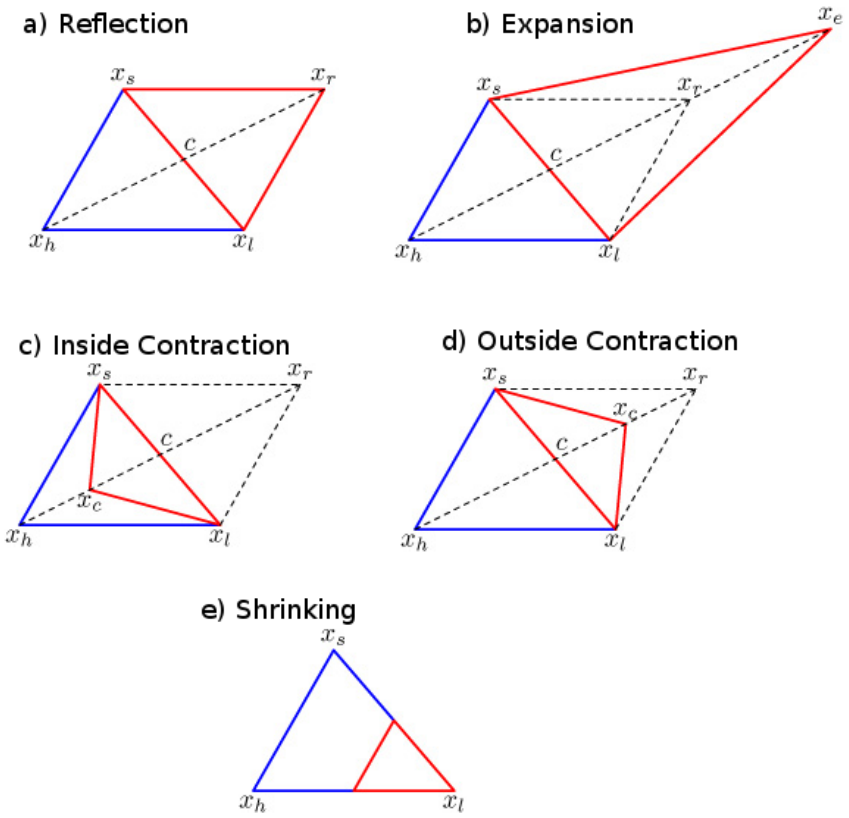

Figure 2.2: Geometric interpretations of the simplex transformations. Taken from [3]

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 $\delta = \frac{1}{2}$ [1, 3].

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 [1].

Bibliography

[1] 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.

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

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

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