# OBJECT TRACKING USING STATISTICAL MODELS OF APPEARANCE

*Mikkel B. Stegmann*

Informatics and Mathematical Modelling, Technical University of Denmark
Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, Denmark

## ABSTRACT

This paper demonstrates that (near) real-time object tracking can be accomplished by the deformable template model; the Active Appearance Model (AAM) using only low-cost consumer electronics such as a PC and a web-camera. Successful object tracking of perspective, rotational and translational transformations was carried out using a training set of five images. The tracker was automatically initialised by a described multi-scale initialisation method and achieved a performance in the range of 7-10 frames per second.

**Keywords:** Deformable Models, Model Initialisation, Real-time tracking, Motion, Active Appearance Models.

## 1. INTRODUCTION

Traditionally, object tracking required special-tailored segmentation algorithms in order to meet the real-time constraints of a motion capture system. However, as the computational performance has increased, models of increasing complexity have been applied for tracking purposes.

In this paper we apply a generative model capable of synthesizing near-photo realistic images of the object class being tracked – i.e. faces, hands et cetera. The basis for this is sampling of shape and texture in a training set. This is the Active Appearance Model (AAM) [1, 2], which is generally considered a fairly complex deformable template model and has previously been applied to still images of faces [1, 6], brain MRI [4], cardiac MRI [18, 16], bones [17, 16] et cetera. However, as we shall see in the following the AAM is also suitable for live tracking in video sequences.

The paper is organised as follows: section 2 gives a short introduction to the theory of Active Appearance Models, section 3 describes a developed multi-scale initialisation scheme, section 4 shows experimental results, section 5 gives pointers for future work and section 6 discuss the results.

Project web-site: http://www.imm.dtu.dk/∼aam/

## 2. ACTIVE APPEARANCE MODELS

Below is presented the outline of the Active Appearance Model approach. AAMs distinguish themselves from many other segmentation methods in the sense that segmentation can be carried out using the approach as a black box. The user only needs to provide a training set of annotated shapes and a set of corresponding images. For further details refer to [1, 3, 16].

### 2.1. Shape Formulation

AAMs handle planar shapes as a finite set of landmarks – i.e. corresponding points between and within populations. The representation used for a single $n$-point shape is:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n]^{\mathrm{T}} \tag{1}$$

For dealing with redundancy in multivariate data – such as shapes – AAMs utilise the linear orthogonal transformation; *principal component analysis* (PCA). In this setting a shape of $n$ points is thus considered one observation, $\mathbf{x}_i$, in a $2n$ dimensional space.

The shape PCA is essentially an eigen-analysis of the covariance matrix of the shapes aligned w.r.t. position, scale and rotation, i.e. after a Procrustes analysis. For details on PCA shape decomposition refer to appendix A.

New shape instances can thus be synthesised by deforming the mean shape, $\overline{\mathbf{x}}$, using a linear combination, $\mathbf{b}_s$, of the eigenvectors of the covariance matrix, $\mathbf{\Phi}_s$:

$$\mathbf{x} = \overline{\mathbf{x}} + \mathbf{\Phi}_s \mathbf{b}_s \tag{2}$$

Essentially, the points of the shape are transformed into a *modal representation* where modes are ordered according to the percentage of variation that they explain. To regularize our solution sapce and improve performance modes are included until the cumulated variation explained by the model is above a certain threshold (e.g. 95%).

### 2.2. Texture Formulation

Contrary to the prevalent understanding of the term *texture* in the computer vision community, this concept will be used somewhat differently below. Here we define texture as *"The pixel intensities across the object in question (if necessary after a suitable normalisation)"*. For $m$ samples over the object surface, the texture is represented as:

$$\mathbf{g} = [g_1, g_2, \ldots, g_m]^{\mathrm{T}} \tag{3}$$

In the shape case, the data acquisition is straightforward because the landmarks in the shape vector constitute the data itself. In the texture-case one needs a consistent method for collecting the texture information between the landmarks, i.e. an image sampling function needs to be established. This can be done in several ways. Here, a piece-wise affine warp based on the Delaunay triangulation of the mean shape is applied.

Following the warp from an actual shape to the mean shape, a normalisation of the $\mathbf{g}$-vector set is performed to avoid the influence from global linear changes in pixel intensities. Hereafter, the analysis is identical to that of the shapes. Hence, a compact representation is derived to deform the texture in a manner similar to what is observed in the training set:

$$\mathbf{g} = \overline{\mathbf{g}} + \mathbf{\Phi}_g \mathbf{b}_g \tag{4}$$

Where $\overline{\mathbf{g}}$ is the mean texture; $\mathbf{\Phi}_g$ denotes the eigenvectors of the covariance matrix and finally $\mathbf{b}_g$ is the set of texture deformation parameters.

### 2.3. Combined Model Formulation

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

$$\mathbf{b} = \mathbf{Q}\mathbf{c} \tag{5}$$

The PCA scores are directly obtained due to the linear nature of the model:

$$\mathbf{b} = \begin{bmatrix} \mathbf{W}_s \mathbf{b}_s \\ \mathbf{b}_g \end{bmatrix} = \begin{bmatrix} \mathbf{W}_s \mathbf{\Phi}_s^{\mathrm{T}} (\mathbf{x} - \overline{\mathbf{x}}) \\ \mathbf{\Phi}_g^{\mathrm{T}} (\mathbf{g} - \overline{\mathbf{g}}) \end{bmatrix} \tag{6}$$

A suitable weighting between pixel distances and pixel intensities is obtained through the diagonal matrix $\mathbf{W}_s$. For details on choosing $\mathbf{W}_s$ refer to [3].

Now, a complete model instance including shape, $\mathbf{x}$ and texture, $\mathbf{g}$, is generated using the combined model parameters, $\mathbf{c}$.

$$\mathbf{x} = \overline{\mathbf{x}} + \mathbf{\Phi}_s \mathbf{W}_s^{-1} \mathbf{Q}_s \mathbf{c} \tag{7}$$

$$\mathbf{g} = \overline{\mathbf{g}} + \mathbf{\Phi}_g \mathbf{Q}_g \mathbf{c} \tag{8}$$

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

### 2.4. Optimisation

In AAMs, the search is treated as an optimisation problem in which the difference between the synthesized object delivered by the AAM and an actual image is to be minimised. By adjusting the AAM-parameters ($\mathbf{c}$ and pose) the model texture, $g_{model}$, can be deformed to fit the image, $g_{image}$, in the best possible way. In this case the quadratic error norm is applied as optimisation criterion [3]:

$$E = \sum_{i=1}^{m} (g_{model} - g_{image})^2 = \sum_{i=1}^{m} (\delta g_i)^2 = ||\delta \mathbf{g}||^2 \tag{9}$$

Though the parameterisation of the object class in question can be compressed markedly by the principal component analysis it is far from an easy task to optimise the system. This is not only computationally cumbersome but also theoretically challenging since it is most likely non-convex. AAMs handle these potential problems by assuming a linear relationship between parameter changes, $\delta\mathbf{c}$, and pixel differences, $\delta\mathbf{g}$.

$$\delta \mathbf{c} = \mathbf{R} \delta \mathbf{g} \tag{10}$$

Since the matrix $\mathbf{R}$ is estimated once at model building time, this is very run-time efficient. In practice $\mathbf{R}$ is estimated by a set of experiments on the training set, which are fed into a multivariate principal component regression framework. In the AAM optimisation, this prediction scheme is applied iteratively to both model- and pose- parameters. For details refer to [3, 1, 16].

Recently [2, 4] the above method have been superseded by a weighted estimation of a fixed $\frac{\partial(\delta \mathbf{g})}{\partial \mathbf{c}}$ done from e.g. the training set using numeric differentiation. In practice – according to Cootes – this works as well as the principal component regression, but with a far smaller computational burden. However, this has not been utilised in the current work.

### 2.5. Model Complexity

As pointed out by [2, 4] the complexity of an AAM is $O(m \cdot p)$ at each iteration of the optimisation. Here $m$ denotes the number of pixels in the model and $p$ the number of modes included into the combined appearance model – i.e. the length of $\mathbf{c}$. Usually the model would converge in 10-20 iterations, where the first few iterations would account for far the most of the decrease in our optimisation criterion.

So – in summary – to reduce the computational burden in the AAM optimisation one could limit the amount of 1) maximum iterations, 2) number pixels in the texture model and 3) number of modes.

## 3. MULTI-SCALE INITIALISATION

The basic AAM optimisation scheme is inherently dependent on good initialisation. To accommodate this, we devise the following search-based scheme thus rendering the use of AAMs fully automated. The technique is inspired by the work of Cootes et al. [5] who use a pixel difference evaluation criteria and a threshold estimation for detecting multiple object instances.

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

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

This has proven to be feasible, fast and robust. A set of relevant search configuration ranges is established and the sampling within this set is done as sparsely as possible. Further, this is done in scale-pyramid to increase speed. Any available prior knowledge about pose is utilised when determining search ranges.

The crucial part of this algorithm is somewhat inspired from the class of Genetic Algorithms.[1] The total set of search configurations constitutes the initial population of candidates. From this we let the $n$ fittest survive. These are then reproduced into more evolved guesses. From these the best is drawn and deemed the initial configuration. In pseudo-code, the initialisation scheme for detecting one object per image is:

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

**Algorithm 1** Multi-scale AAM Initialisation

**Require:** $m$: max number of transient iterations
**Require:** $k$: max number of final iterations ($k > m$)
**Require:** $n$: number of initialisation candidates
**Require:** $\{\mathbf{\Omega}\}$: An empty candidate set with room for $n$ result entries, $\{\omega_i\}_{i=1}^{n} = \{\boldsymbol{\theta}_i, E_i\}_{i=1}^{n}$
**Require:** $\{\mathbf{\Psi}\}$: A set of application specific search ranges for each model parameter (e.g. $-\sigma_1 \leq c_1 \leq \sigma_1$, $x_{min} \leq x \leq x_{max}$ et cetera)

```
Populate the space spanned by {Ψ} – as sparsely as the linear regression allows – by a set of search configurations Θ = {θ_1, ..., θ_n}.
for each vector in Θ do
    Run AAM optimisation at θ_i (max m iterations)
    Calculate the fit, E ← ||δg||^2
    ω_max ← max_E {Ω}
    if E < E_max then
        if the number of elements in {Ω} == n then
            then remove max_E {Ω}
        end if
        add (θ_i, E) to {Ω}
    end if
end for
for each element in {Ω} do
    Run AAM optimisation at θ_i (max k iterations)
    Calculate and update the fit, E_i ← ||δg||^2
end for
ω_initial ← min_E {Ω}
for each scale level do
    Run AAM optimisation at θ_initial (max k iterations)
end for
return θ_initial
```

We stress that the application specific search ranges mentioned in the fifth requirement are merely a help to increase initialisation speed and robustness rather than a requirement. If no prior is known, an exhaustive search is performed.

This scheme is readily extended into more than one object per image by a clustering of the candidate set using overlap tests. For a detailed treatment of initialisation of deformable template models refer to [8].

## 4. EXPERIMENTAL RESULTS

To test the real-time capabilities of Active Appearance Models a very simple model was built from the five colour training images in the CIF format (352×288). As seen in fig. 1 the training set consists of five perspective views of a planar object – namely a DAT cassette – where the object has undergone a rotation around the vertical axis. The images were sub sampled to QCIF (174×144) and converted

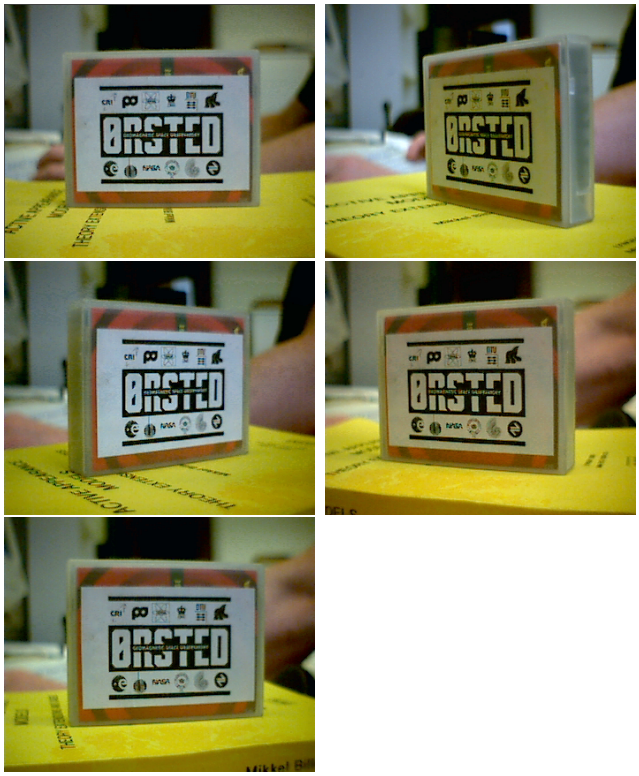


**Fig. 1**. Training set.

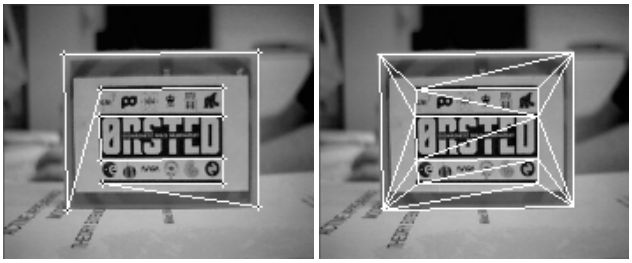


**Fig. 2**. Annotated image (left). Delaunay triangulation (right).

to greyscale prior to any processing by the AAM, though the extension to colour AAMs is fairly simple [5]. Each training image was subsequently annotated using 12 landmark points as shown in fig. 2. To capture the object we let the convex hull of the landmarks denote the model surface.

Upon the training set a two-level multi-scale AAM was built using the AAM-API[2]. The texture model consisted of 9100 pixel at scale level 0 and 2261 pixels at level 1. The variance explained by the first three eigenvalues in the combined shape and texture model were approximately 69%, 22% and 7%. In fig. 3 the three most significant modes of variation in the shape model is shown. Due to rather controlled lightning conditions, the texture model is fairly constant. Major contribution herein was instead the remaining variation stemming from the piece-wise affine warps inability to represent the inherently perspective transformation the object has undergone in the training set.

[2]The AAM-API is an open source AAM implementation done in C++ by the author. The AAM-API is further described in [17, 16] and can be acquired from the web-site http://www.imm.dtu.dk/∼aam/

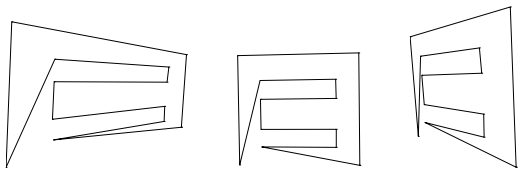


**Fig. 3**. The first shape mode of the AAM.

**Fig. 4**. The first three shape modes of the AAM.

Initial position of the object was found by the multi-scale initialisation described previously. No temporal filtering was applied to produce the results. The result from the current frame was simply propagated directly to the next frame in the video stream. To increase the frame rate the AAM search was fixed to a maximum of three iterations. We anticipate that even simple temporal filtering would increase the robustness of the tracking markedly in a real-life situation. In that case, robustness to occlusions and dramatic lightning effects (such as shadows, specular highlights etc.) should also be addressed. This is a critical issue since the AAM search is based on strong prior assumptions and when these don't hold the prediction gets highly erroneous. Cootes et al. [5] addresses the problem by applying a pre-learnt threshold on the texture vectors sampled from the image. Sclaroff and Isidoro [15] handles the problem by means of the robust Lorentzian norm in a prediction scheme similar to that of AAMs.

The tracker performed successful tracking in the range of 7-10 frames/sec at QCIF resolution (174x144). Examples are given in fig. 5.

### 4.1. Implementation and Hardware

The AAM-API was embedded in a C++ Win32 application framework providing real-time video input. Parts of this was provided by the Microsoft Vision SDK [13].

In order to speed up matrix computations in the AAM-API, matrix classes was augmented with processor optimised BLAS support by the Intel Math Kernel Library – MKL [9]. All tests were performed on an 1100 MHz Athlon PC.

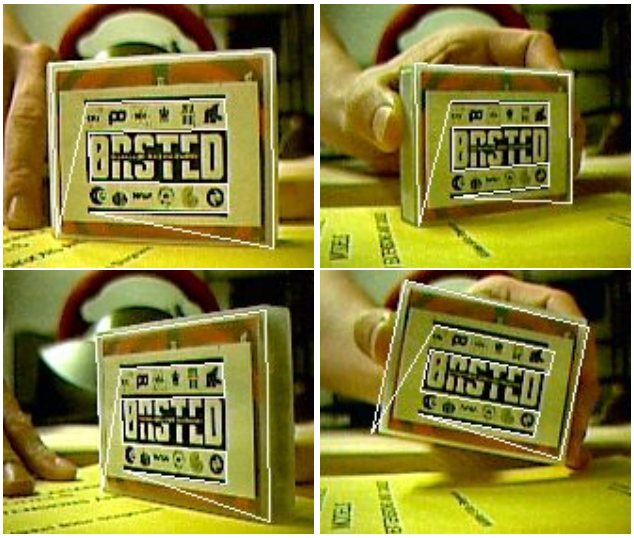


**Fig. 5**. Tracking results computed at 7-10 frames/sec.

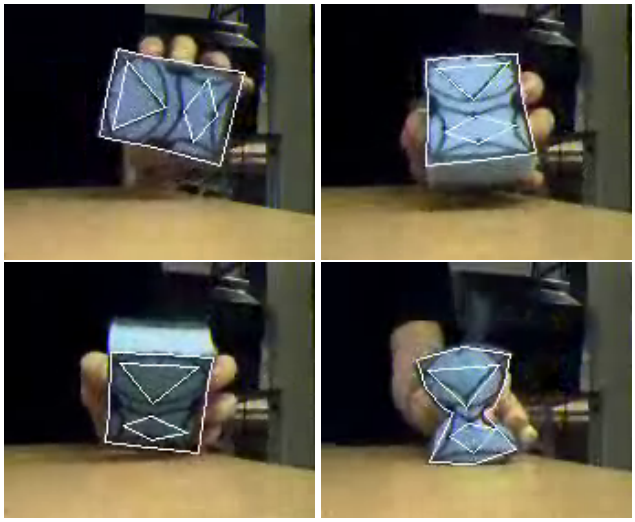


**Fig. 6**. AAM tracking of a deformable object by Nielsen, Lehn-Schiøler & Wrobel [14].

### 4.2. Tracking of Deformable Objects

In conjunction with the above experiment – and in resemblance to the Active Voodoo Dolls [11] – Nielsen, Lehn-Schiøler & Wrobel [14] have shown that a simple linear model parameter prediction markedly improved tracking quality when applied on a deformable object (a sponge). This was accomplished using the AAM-API in an off-line setting on basis of a training set of 11 images each annotated with 21 points. The texture model consisted of 5851 pixels and the combined model used four parameters. In each frame, a maximum of 20 iterations was allowed in the optimisation procedure. Examples are given in fig. 6. Notice the change of lightning conditions in each of the frames.

## 5. FUTURE WORK

A continuation of this work would naturally fall into one of the following categories (in no particular order):

- Temporal filtering to increase robustness by using the much celebrated linear predictor by Rudolf Kalman [12] (used on ASMs [7]) or the more recent CONDENSATION algorithm by Isard and Blake [10].
- Robustness to occlusions and lightning changes inspired by [5, 15, 11].
- Inclusion of colour into the model, either directly as RGB [5] or under some mapping, e.g. HSV etc.
- Speeding up the current implementation by intensive usage of dynamic programming and hardware accelerated libraries such as an OpenGL and Intel Image Processing Library. Further, explicit usage of MMX / SSE / 3DNow! optimised code for parallel operations would presumably also give a substantial speed up.

We anticipate that a continuation following the lines of the above would enable us to perform real-time (>25Hz) tracking of facial features. Applications would be assisted speech recognition, virtual characters, perceptual user interfaces etc.

## 6. DISCUSSION

Though Active Appearance Models is considered as one the more complex deformable template models, this paper has emphasised the fact that it is the off-line model building phase, which is computationally expensive. AAM optimisation on contrary is well suited for real-time implementations. In this context, we have achieved online tracking at 7-10 Hz with a non-optimised software implementation.

The model-specific data was minimised by using only five training images. This enabled us to set up a complete model (annotation, model building etc.) within fifteen minutes. From this training set robustness to three-dimensional rotation around one axis was obtained. Robustness to planar similarity transforms (Euclidian transformations) was obtained automatically as a part of model building phase.

Further, though the presented problem – tracking of a planar object in a perspective projection – is far form optimally represented in the AAM framework (due to the 2D PCA decomposition and 2D piece-wise affine warp) this experiment has shown that the crude approximations holds "well enough" to enable an acceptable tracking quality. This underlines the general nature of AAMs even in cases of extremely small training sets.

## 7. ACKNOWLEDGEMENTS

Hans P. Palbøl is gratefully acknowledged for providing the all-important Athlon PC used for the experimental results.

## 8. APPENDIX

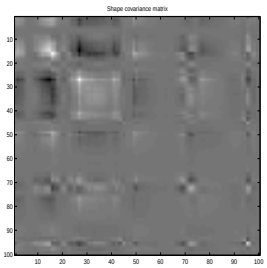
### A – PCA SHAPE DECOMPOSITION

This appendix contains a self-contained presentation of how the Principal Component Analysis (PCA) can be derived by means of simple linear algebra.

Consider the case of having $N$ planar shapes[3] consisting of $n$ points, where each shape is represented as:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n]^{\mathrm{T}} \tag{11}$$

Looking at the covariance matrix of these shapes we typically observe a pattern similar to the one shown below:




This suggest that there might exist a shape representation accounting for the obvious correlation between points. If some point movements were to be (numerically) much correlated, this could be exploited to reduce dimensionality. In this case, we will seek a linear transformation of our data:

$$\mathbf{y} = \mathbf{M}\mathbf{x} \tag{12}$$

First, consider the mean shape $\overline{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$ and the estimate of the shape covariance matrix:

$$\mathbf{\Sigma}_{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^{\mathrm{T}} \tag{13}$$

The mean of the $\mathbf{y}$-variables can then be expressed as:

$$\overline{\mathbf{y}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{y}_i = \frac{1}{N}\sum_{i=1}^{N}\mathbf{M}\mathbf{x}_i = \mathbf{M}\overline{\mathbf{x}} \tag{14}$$

[3]Due to the context the data vectors represent shapes, but we emphasise that the PCA can be applied to any type of data as long as a the data forms hyper ellipsoid in the original space.

And consequently the estimate of the covariance of the $\mathbf{y}$'s:

$$\begin{aligned}
\mathbf{\Sigma}_{\mathbf{y}} &= \frac{1}{N}\sum_{i=1}^{N}(\mathbf{y}_i - \overline{\mathbf{y}})(\mathbf{y}_i - \overline{\mathbf{y}})^{\mathrm{T}} \\
&= \frac{1}{N}\sum_{i=1}^{N}(\mathbf{M}\mathbf{x}_i - \mathbf{M}\overline{\mathbf{x}})(\mathbf{M}\mathbf{x}_i - \mathbf{M}\overline{\mathbf{x}})^{\mathrm{T}} \\
&= \frac{1}{N}\sum_{i=1}^{N}\mathbf{M}(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{M}(\mathbf{x}_i - \overline{\mathbf{x}}))^{\mathrm{T}} \\
&= \frac{1}{N}\sum_{i=1}^{N}\mathbf{M}(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^{\mathrm{T}}\mathbf{M}^{\mathrm{T}} \\
&= \mathbf{M}\left(\frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^{\mathrm{T}}\right)\mathbf{M}^{\mathrm{T}} \\
&= \mathbf{M}\mathbf{\Sigma}_{\mathbf{x}}\mathbf{M}^{\mathrm{T}}
\end{aligned} \tag{15}$$

Then, if we limit ourselves to orthogonal transformations (i.e. $\mathbf{M}^{-1} = \mathbf{M}^{\mathrm{T}}$) left-multiplication by $\mathbf{M}^{\mathrm{T}}$ in (15) yields:

$$\mathbf{M}^{\mathrm{T}}\mathbf{\Sigma}_{\mathbf{y}} = \mathbf{\Sigma}_{\mathbf{x}}\mathbf{M}^{\mathrm{T}} \tag{16}$$

Substitution of $\mathbf{M}^{\mathrm{T}}$ by $\mathbf{\Phi}$ yields:

$$\mathbf{\Sigma}_{\mathbf{x}}\mathbf{\Phi} = \mathbf{\Phi}\mathbf{\Sigma}_{\mathbf{y}} \tag{17}$$

From (17) it is seen that if $\mathbf{\Phi}$ is chosen as the (column) eigenvectors of the symmetric matrix $\mathbf{\Sigma}_{\mathbf{x}}$, then the covariance of the transformed shapes, $\mathbf{\Sigma}_{\mathbf{y}}$, becomes a diagonal matrix of eigenvalues. In the case of correlated points the smallest eigenvalues will be (close to) zero and the corresponding eigenvectors could be omitted from $\mathbf{\Phi}$, thus reducing the length of $\mathbf{y}$.

In conclusion, to establish an orthogonal linear transform that de-correlate data vectors, the transformation matrix must be the eigenvectors of the covariance matrix of the original data. In order to back transform from the new set of variables, $\mathbf{y}$, we invert (12), remembering that $\mathbf{M}$ is orthogonal:

$$\mathbf{x} = \mathbf{M}^{-1}\mathbf{y} = \mathbf{M}^{\mathrm{T}}\mathbf{y} = \mathbf{\Phi}\mathbf{y} \tag{18}$$

As a final comment; one would typically (as in the AAM case) apply PCA on variables with zero mean:

$$\mathbf{y} = \mathbf{M}(\mathbf{x} - \overline{\mathbf{x}}) \quad , \quad \mathbf{x} = \overline{\mathbf{x}} + \mathbf{\Phi}\mathbf{y} \tag{19}$$

This method of dealing with redundancy in multivariate data is known as *Principal Component Analysis* (PCA) or the *Karhunen-Loéve Transform* (KLT).

## 9. REFERENCES


[1] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In *Proc. European Conf. on Computer Vision*, volume 2, pages 484–498. Springer, 1998.

[2] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Trans. on Pattern Recognition and Machine Intelligence*, 23(6):681–685, 2001.

[3] T. F. Cootes and C. J Taylor. *Statistical Models of Appearance for Computer Vision*. Tech. Report , University of Manchester, http://www.isbe.man.ac.uk/∼bim/, Feb. 2000.

[4] T. F. Cootes and C. J. Taylor. Statistical models of appearance for medical image analysis and computer vision. In *Proc. SPIE Medical Imaging 2001*, volume 1. SPIE, 2001.

[5] G. J. Edwards, T.F. Cootes, and C. J. Taylor. Advances in active appearance models. In *Proc. Int. Conf. on Computer Vision*, pages 137–142, 1999.

[6] G.J. Edwards, T. F. Cootes, and C. J. Taylor. Face recognition using active appearance models. In *ECCV'98. 5th European Conf. on Computer Vision. Proc.*, volume 2, pages 581–95. Springer-Verlag, 1998.

[7] G.J. Edwards, C. J. Taylor, and T. F. Cootes. Learning to identify and track faces in image sequences. In *6th Int. Conf. on Computer Vision*, pages 317–22. Narosa Publishing House, 1998.

[8] R. Fisker. *Making Deformable Template Models Operational*. PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, 2000.

[9] Intel. Math kernel library, 2000. Freeware, Web page: http://developer.intel.com/software/products/mkl/.

[10] M. Isard and A. Blake. CONDENSATION-conditional density propagation for visual tracking. *Int. Jour. Computer Vision*, 29:5–28, 1998.

[11] J. Isidoro and S. Sclaroff. Active voodoo dolls: a vision based input device for nonrigid control. In *Proc. Computer Animation '98*, pages 137–143. IEEE Comput. Soc, 1998.

[12] R. Kalman. A new approach to linear filtering and prediction problems. *ASME Basic Engineering Journal*, 82:35–45, March 1960.

[13] Microsoft. Vision SDK, 2000. Freeware, Web page: http://research.microsoft.com/projects/VisSDK/.

[14] M. E. Nielsen, T. Lehn-Schiøler, and M. Wrobel. Tracking of a deformable object. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, February 2001. (internal report in danish).

[15] S. Sclaroff and J. Isidoro. Active blobs. *Proc. of the Int. Conf. on Comput. Vision*, pages 1146–1153, 1998.

[16] M. B. Stegmann. Active appearance models: Theory, extensions and cases. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, 2000. http://www.imm.dtu.dk/∼aam/.

[17] M. B. Stegmann, R. Fisker, and B. K. Ersbøll. Extending and applying active appearance models for automated, high precision segmentation in different image modalities. In *Proc. 12th Scandinavian Conference on Image Analysis - SCIA 2001*, volume 1, pages 90–97, 2001.

[18] M. B. Stegmann, J. C. Nilsson, and B. A. Grønning. Automated segmentation of cardiac magnetic resonance images. In *Proc. International Society of Magnetic Resonance In Medicine - ISMRM 2001*, volume 2, page 827. ISMRM, 2001.