Comparing Variations on the Active Appearance Model Algorithm

T.F. Cootes and P.Kittipanya-ngam
Dept. Imaging Science and Biomedical Engineering
University of Manchester, Manchester M13 9PT U.K.
t.cootes@man.ac.uk

Abstract

The Active Appearance Model (AAM) algorithm has proved to be a successful method for matching statistical models of appearance to new images. Since the original algorithm was described there have been a variety of suggested modifications to the basic algorithm, each typically claiming to be in some way superior. We review these algorithms and report the results of experiments comparing their performance. We also investigate the effects of different methods of estimating the update matrix used in the algorithm. We find that careful choice of the latter has at least as much effect as the choice of updating technique.

1 Introduction

Since its introduction by Edwards *et.al.* [7] the Active Appearance Model (AAM) algorithm has been widely used for matching statistical models of appearance to image data. The models, trained on suitably annotated examples, can synthesize new images of the objects of interest. To match them to new images one seeks the model parameters which minimise the difference between the target image and the synthesized image. The AAM essentially learns the relationship between the residual differences in such a match and the parameter displacements required to correct the current offset from the optimal position.

A number of improvements and alternatives to the original algorithm have been proposed, including different training methods [5], non-linear models [15], novel image representations [18] and a variety of different methods of updating the model during search [4, 14, 9, 17]. Each has usually been tested against an implementation of the original algorithm (and is demonstrated to be superior on the data used), but the different methods have not yet been systematically tested against one another.

In this document we seek to compare and contrast some of these alternatives, and to test their performance quantatively. In addition we consider how to estimate the matrix used to predict the update step in the AAM algorithm, and demonstrate that the method used can considerably influence the performance of the search algorithm. We suggest a novel method of estimating the update matrix which gives significant improvements in performance.

2 Background

The AAM algorithm has been widely used for face and medical image processing. Here we mention some of the more notable applications and extensions to the original algorithm. Romdhani *et.al.* [15] described a version of the AAM which used a non-linear regression (SVM regression) to predict the parameter updates from the texture errors. Mitchell *et.al.* [13] describe building a 2D+time Appearance model of the time varying appearance of the heart in complete cardiac MR sequences. Bosch *et.al.* [2] describe building a 2D+time Appearance model and using it to segment parts of the heart boundary in echocardiogram sequences. The paper describes an elegant method of dealing with the strongly non-gaussian nature of the noise in ultrasound images. Stegmann and Fisker [8, 10, 16] have shown that applying a general purpose optimiser can improve the final match obtained by an AAM. Cootes and Taylor [18] describe how using non-linearly normalised gradient information in the appearance model can improve the matching performance, particularly on unseen data sets with different imaging conditions. Stegmann and Larsen [11] demonstrate that using multiple features at each pixel (eg intensity, hue and edge strength) leads to more accurate face location. Alternative updating schemes proposed by Hou *et.al.* [9] and Baker and Matthews [14] are described in more detail below.

More recently Yan *et.al.* [17] have described how the prediction of shape from the texture and that from an Active Shape Model style search can lead to more accurate model matching.

3 Statistical Models of Appearance

An appearance model can represent both the shape and texture variability seen in a training set. The training set consists of labelled images, where key landmark points are marked on each example object. Using the notation of Cootes *et.al.* [3], the shape of an object can be represented as a vector $\mathbf{x}$ and the texture (grey-levels or colour values) represented as a vector $\mathbf{g}$.

The shape and texture are controlled by a statistical models of the form

$$\begin{aligned} \mathbf{x} &= \bar{\mathbf{x}} + \mathbf{P}_s\mathbf{b}_s \\ \mathbf{g} &= \bar{\mathbf{g}} + \mathbf{P}_g\mathbf{b}_g \end{aligned} \tag{1}$$

Where $\mathbf{b}_s$ are shape parameters, $\mathbf{b}_g$ are texture parameters.

Since often shape and texture are correllated, we can take this into account in a combined statistical model of the form

$$\begin{aligned} \mathbf{b}_s &= \mathbf{C}_s\mathbf{c} \\ \mathbf{b}_g &= \mathbf{C}_g\mathbf{c} \end{aligned} \tag{2}$$

The combined appearance model has parameters, $\mathbf{c}$, controlling the shape and texture at the same time. Combining (1) and (2) gives

$$\begin{aligned} \mathbf{x} &= \bar{\mathbf{x}} + \mathbf{Q}_s\mathbf{c} \\ \mathbf{g} &= \bar{\mathbf{g}} + \mathbf{Q}_g\mathbf{c} \end{aligned} \tag{3}$$

where $\bar{\mathbf{x}}$ is the mean shape (in a normalised frame), $\bar{\mathbf{g}}$ the mean texture (suitably normalised) and $\mathbf{Q}_s$,$\mathbf{Q}_g$ are matrices describing the modes of variation derived from the training set.

To generate the positions of points in an image we use

$$\mathbf{X} = T_{\mathbf{t}}(\mathbf{x}) \tag{4}$$

where $\mathbf{x}$ are the points in the model frame, $\mathbf{X}$ are the points in the image, and $T_{\mathbf{t}}(\mathbf{x})$ applies a global transformation with parameters $\mathbf{t}$. For instance, in 2D, $T_{\mathbf{t}}(\mathbf{x})$ is commonly a similarity transform with four parameters describing the translation, rotation and scale.

The texture in the image frame is generated by applying a scaling and offset to the intensities, $\mathbf{g}_{im} = T_{\mathbf{u}}(\mathbf{g})$ where $\mathbf{u}$ is the vector of transformation parameters.

4 Active Appearance Models

The AAM algorithm is a method of rapidly matching an appearance model to an image. It is a form of gradient descent algorithm, in which the gradient is assumed to be fixed at all iterations, and can thus be estimated in advance from a training set. This allows efficient matching to take place, even when there are many model parameters.

The original matching algorithm was first described by Edwards *et.al.* [7] and expanded and refined by Cootes *et.al.* [3, 5]. In the following we will describe the various approaches in enough detail to establish notation.

4.1 Basic algorithm

The appearance model parameters, $\mathbf{c}$, and shape transformation parameters, $\mathbf{t}$, define the position of the model points in the image frame, $\mathbf{X}$, which gives the shape of the image patch to be represented by the model. During matching the pixels in this region of the image, $\mathbf{g}_{im}$, are sampled and projected into the texture model frame, $\mathbf{g}_s = T^{-1}(\mathbf{g}_{im})$. The current model texture is given by $\mathbf{g}_m = \bar{\mathbf{g}} + \mathbf{Q}_g\mathbf{c}$. The difference between model and image (measured in the normalized texture frame) is then

$$\mathbf{r}(\mathbf{p}) = \mathbf{g}_s - \mathbf{g}_m \tag{5}$$

where $\mathbf{p}$ are the parameters of the model, $\mathbf{p}^T = (\mathbf{c}^T|\mathbf{t}^T|\mathbf{u}^T)$.

A simple scalar measure of difference is the sum of squares of elements of $\mathbf{r}$, $E(\mathbf{p}) = \mathbf{r}^T\mathbf{r}$. It can be shown [5] that to minimise this we modify the parameters by

$$\delta\mathbf{p} = -\mathbf{R}\mathbf{r}(\mathbf{p}) \quad where \quad \mathbf{R} = (\frac{\partial\mathbf{r}}{\partial\mathbf{p}}^T \frac{\partial\mathbf{r}}{\partial\mathbf{p}})^{-1} \frac{\partial\mathbf{r}}{\partial\mathbf{p}}^T \tag{6}$$

$\frac{\partial\mathbf{r}}{\partial\mathbf{p}}$ is estimated by numeric differentiation, systematically displacing each parameter from the known optimal value on typical images and computing an average over the training set. $\mathbf{R}$ is computed and used in all subsequent searches with the model.

4.1.1 Basic AAM Search

Equation (6) can be used to suggest a correction to make in the model parameters based on a measured residual $\mathbf{r}$.

Given a current estimate of the model parameters, $\mathbf{c}$, the pose $\mathbf{t}$, the texture transformation $\mathbf{u}$, and the image sample at the current estimate, $\mathbf{g}_{im}$, one step of the iterative matching procedure is as follows:

1. Project the texture sample into the texture model frame using $\mathbf{g}_s = T_{\mathbf{u}}^{-1}(\mathbf{g}_{im})$
2. evaluate the error vector, $\mathbf{r} = \mathbf{g}_s - \mathbf{g}_m$, and the current error, $E = |\mathbf{r}|^2$
3. compute the predicted displacements, $\delta\mathbf{p} = -\mathbf{R}\mathbf{r}(\mathbf{p})$
4. update the model parameters $\mathbf{p} \to \mathbf{p} + k\delta\mathbf{p}$, where initially $k = 1$,
5. calculate the new points, $\mathbf{X}'$ and model frame texture $\mathbf{g}'_m$
6. sample the image at the new points to obtain $\mathbf{g}'_{im}$
7. calculate a new error vector, $\mathbf{r}' = T_{\mathbf{u}'}^{-1}(\mathbf{g}'_{im}) - \mathbf{g}'_m$
8. if $|\mathbf{r}'|^2 < E$ then accept the new estimate, otherwise try at $k = 0.5$, $k = 0.25$ etc.

This procedure is repeated until no improvement is made to the error, $|\mathbf{r}|^2$, and convergence is declared (or a maximum number of iterations is reached). In practice a multi-resolution implementation is used, in which the search is started at a coarse resolution and iterated to convergence at each level.

4.2 Shape AAM

Cootes *et.al.* [4] proposed a variant on the AAM in which instead of the residuals driving the appearance model parameters,$\mathbf{c}$, they could be used to drive the pose,$\mathbf{t}$, and shape model parameters, $\mathbf{b}_s$, alone. The texture model parameters could then be directly estimated by fitting to the current texture.

In a training phase one learns the relationships

$$\begin{array}{rcl} d\mathbf{t} & = & \mathbf{R}_t\mathbf{r} \\ d\mathbf{b}_s & = & \mathbf{R}_s\mathbf{r} \end{array} \tag{7}$$

During the search the update step is modified as follows

1. Normalise the current texture sample to obtain $\mathbf{g}_s$
2. Fit the texture model to the sample using $\mathbf{b}_g = \mathbf{P}_g^T(\mathbf{g}_s - \bar{\mathbf{g}})$
3. Compute the residual as $\mathbf{r} = \mathbf{g}_s - \mathbf{g}_m$
4. Predict the pose and shape parameter updates using (7)
5. Apply and test the updates as for the basic algorithm

As was described in [4] the linear nature of the relationships allows one to predict the shape and pose updates from the normalised texture sample, avoiding fitting the texture model. This can improve the efficiency of the process. However, one must fit the texture model if the total error is to be computed (for instance, to test for improvements).

If required, a combined model of shape and texture can be used to apply constraints to the relative shape and texture parameter vectors.

4.3 Compositional Approach

Baker and Matthews [14] point out that the essentially additive method of updating the parameters in the basic framework can be problematic, and propose an alternative compositional updating scheme. They consider the case in which separate (independent) shape and texture models are used (essentially the Shape AAM above). The shape model equation (1) can be thought of as a parameterised transformation of the mean shape,

$$\mathbf{x} = T_{\mathbf{b}_s}(\hat{\mathbf{x}}) \tag{8}$$

Using an additive update of the form $\mathbf{b}_s \rightarrow \mathbf{b}_s + \delta$, leads to a new transformation

$$\mathbf{x}_{new} = T_{\mathbf{b}_s+\delta}(\hat{\mathbf{x}}) \tag{9}$$

However, it is more natural to think of the update itself as a transformation,

$$\mathbf{x}_{new} = T_{\mathbf{b}_s}(T_\delta(\hat{\mathbf{x}})) \tag{10}$$

In this case we must compute the new parameters, $\mathbf{b}'_s$ such that

$$T_{\mathbf{b}'_s}(\hat{\mathbf{x}}) = T_{\mathbf{b}_s}(T_\delta(\hat{\mathbf{x}})) \tag{11}$$

One way of achieving this is to approximate the transformation using thin-plate splines [1] as follows

- Compute the thin plate spline, $T_{tps}(.)$ which maps the points $\hat{\mathbf{x}}$ to $\mathbf{x}$
- Compute the modified mean points, $\mathbf{x}_\delta = \hat{\mathbf{x}} + \mathbf{P}_s\delta$
- Apply the transformation, $\mathbf{x}' = T_{tps}(\mathbf{x}_\delta)$
- Find the shape parameters which best match, $\mathbf{b}'_s = \mathbf{P}_s^T(\mathbf{x}' - \hat{\mathbf{x}})$

The sampling and updating is otherwise identical to that for the Shape AAM described above.

4.4 Direct AAMs

Hou *et.al.* [9] suggest that in some cases it is possible to predict the shape directly from the texture, when the two are sufficiently correllated. From eq. (2)

$$\mathbf{b}_s = \mathbf{S}\mathbf{b}_g = \mathbf{C}_s\mathbf{C}_g^{-1}\mathbf{b}_g \tag{12}$$

where $\mathbf{C}_g^{-1}$ is the pseudo-inverse of $\mathbf{C}_g$. Thus we can predict the shape parameters from the texture accurately if the rank of $\mathbf{C}_g$ is larger than the number of shape parameters. Hou *et.al.* demonstrated that this was the case for a face model. [1]

One iteration of the matching procedure is then as follows

1. Fit the texture model to the normalised sample using $\mathbf{b}_g = \mathbf{P}_g^T(\mathbf{g}_s - \bar{\mathbf{g}})$
2. Compute the residual as $\mathbf{r} = \mathbf{g}_s - \mathbf{g}_m$
3. Predict the pose using $d\mathbf{t} = \mathbf{R}_t\mathbf{r}$
4. Predict the shape using $\mathbf{b}_s = \mathbf{S}\mathbf{b}_g$
5. Apply and test the updates as for the basic algorithm

More recently Yan *et.al.* [17] have extended this approach, combining a shape prediction from an ASM like search with that from the texture model above. This is claimed to provide more accurate results than either. Unfortunately we did not have time to implement this new method.

5 Experiments

We wished to compare the performance of the four algorithms described above.

We constructed an appearance model from 102 face images, each annotated with 68 landmarks. Retaining 98% of the shape variation gave 49 shape modes. Retaining 98% of the texture variation (for a 3000 pixel model) gave 73 texture modes. Retaining 99.9% of the combined variation gave 92 combined modes. Note that if the shape were completely predictable from the texture, we would only 73 combined modes. We may have to retain more texture modes to get complete prediction, but we may then be just matching to the noise.

We adopted the following procedure to compare the performance of the different algorithms. We used a set of 155 test images of people (those who had no glasses or facial hair) taken from the XM2VTS [12] data set.

On each test image we start a search with the mean model shape at positions displaced from the known optimal centre by ± 10 pixels in x and y. After running a 3-level multi-resolution AAM (allowing at most 10 iterations per level) we compute the mean error between model points and hand labelled points (pt-pt), the mean distance of the model points to the appropriate hand labelled boundaries (pt-crv) and the RMS texture error (in the model frame).

Summary statistics of positional and texture errors for the four search methods (the results of 1395 individual searches) are shown in Table 5. [2]

This suggests that given the particular set of search parameters, the shape and composition AAMs significantly out-perform the basic and direct AAMs, and that the basic AAM is significantly worse than the direct AAM. The time given is the mean search time on a 550MHz PC running Linux.

[1]It shouldbe noted that there are many cases where shape is sufficiently independent of the texture for this to be an in-appropriate assumption (consider a rectangle of varying aspect ratio but fixed colour).This is particularly true in various medical image analysis problems.

[2]Note that face model is about 60 pixels across and faces are about 200 pixels across in image. Thus each model pixel is about 3 image pixels wide when fit.

AAM Type	Pt-Pt Error (Pixels)		Pt-Crv Error (Pixels)		Texture Error (Grey-scale)		Time ms
	Median	90%ile	Median	90%ile	Median	90%ile	mean
Basic	11.6	15.6	7.8	10.2	18.9	26.2	172
Direct	**9.4**	**13.6**	6.3	9.0	15.5	19.9	172
Shape	9.4	15.1	**5.0**	**8.9**	**12.8**	**17.2**	292
Composition	9.4	15.4	5.0	9.1	12.8	17.5	332

Table 1: Comparative performance of different AAMs (no forced iterations) (XM2VTS set images)

5.1 Forcing first iteration step

We have found that search performance can be improved by applying the predicted update without testing whether it improves the result or not - this appears to allow us to jump over local minima.

Here we apply one forced iteration at each resolution, then only accept subsequent steps which improve the results. Results summarised in Table 5.1.

This strategy slightly improves the performance of the shape and Composition AAMs, degrades the Direct AAM but significantly improves that of the Basic AAM, leading to significantly better location accuracy than any in the previous experiment.

AAM Type	Pt-Pt Error (Pixels)		Pt-Crv Error (Pixels)		Texture Error (Grey-scale)	
	Median	90%ile	Median	90%ile	Median	90%ile
Basic	**8.4**	**12.4**	**4.7**	**7.7**	13.8	17.9
Direct	11.2	15.5	7.1	10.2	17.0	21.3
Shape	9.8	15.4	5.1	9.1	**12.7**	**16.7**
Composition	9.8	15.6	5.2	9.1	12.8	16.9

Table 2: Comparative performance of different AAMs (one forced iteration) (XM2VTS set images)

5.2 Forcing iteration steps

For comparison we apply 10 such 'forced' iterations at every level. The results are summarised in Table 5.2.

They suggests that such a strategy is very poor, though the results for the Basic AAM are the least bad.

5.3 Calculating the Update Matrix

All the methods rely on estimating a matrix, $\mathbf{R}$, which predicts the parameter updates from the current texture error, $\delta\mathbf{p} = \mathbf{R}\mathbf{r}$. The original work on AAMs [7, 3] used a regression based approach to compute this. Later it was suggested that a better method was to compute the gradient matrix $\frac{\partial\mathbf{r}}{\partial\mathbf{p}}$ [5] - this was claimed to be more stable, had a

AAM Type	Pt-Pt Error (Pixels)		Pt-Crv Error (Pixels)		Texture Error (Grey-scale)	
	Median	90%ile	Median	90%ile	Median	90%ile
Basic	**9.6**	**29.4**	**5.7**	**20.6**	15.0	21.5
Direct	13.5	25.2	8.7	16.1	16.1	21.4
Shape	13.7	48.8	8.0	34.1	**14.6**	**20.1**
Composition	13.9	55.8	8.2	40.4	15.3	26.2

Table 3: Comparative performance of different AAMs (All forced iterations) (XM2VTS set images)

clearer mathematical interpretation and allowed extra constraints to be easily incorporated [6]. Here we compare the methods and suggest an improved algorithm for model training.

The regression based approach assumes that we have repeatedly displaced the model parameters and computed the resulting residual. If the parameter displacements are stored in the columns of $\mathbf{C}$ and the corresponding residuals in the columns of $\mathbf{V}$ then we require a matrix $\mathbf{R}$ such that $\mathbf{C} = \mathbf{R}\mathbf{V}$. This can then be used in the update method.

A simple approach to obtaining $\mathbf{R}$ is to compute

$$\mathbf{R} = \mathbf{V}^{+}\mathbf{C} \tag{13}$$

where $\mathbf{V}^{+}$ is the pseudo-inverse of $\mathbf{V}$.

However, this will tend to overfit to the training data unless there are more displacements than pixels modelled (a rare occurence). Various techniques have been proposed to correct for this, a simple one being to apply PCA to reduce the dimensionality of the residuals before performing the regression. This has been done by a variety of groups. For instance Hou *et.al.* [9] demonstrate this leads to better performance, presumably because the result is smoother and less likely to be overtrained.

However, an alternative approach is to attempt to compute the gradient matrix, $\frac{\partial \mathbf{r}}{\partial \mathbf{p}}$, such that $\mathbf{r} = \frac{\partial \mathbf{r}}{\partial \mathbf{p}}\delta\mathbf{p}$. This can be estimated directly by displacing each parameter in turn [5]. An alternative novel approach is to estimate it from randomly displaced data as follows. Using the same data as described above (multiple random displacements), we can determine the gradient matrix using

$$\frac{\partial \mathbf{r}}{\partial \mathbf{p}} = \mathbf{C}^{+}\mathbf{V} \tag{14}$$

Because of their relative sizes, it is usually much quicker to compute the pseudoinverse $\mathbf{C}^{+}$ than $\mathbf{V}^{+}$.

In this case we can compute the update matrix simply as $\mathbf{R} = \frac{\partial \mathbf{r}}{\partial \mathbf{p}}^{+}$ [6].

This approach avoids the danger of over-fitting exhibited by the simple regression approach.

Table 5.3 summarises the results of searches using the different calculation methods. The AAM was trained using displacements on 10 images from the original training set. In the case of PCA-Regression we retained as many modes as model parameters. The regression methods were trained using either 20 or 50 random displacements per image.

The results suggest that (on this data) the best approach is to use a regression approach to compute the gradient matrix if position accuracy is important, or the direct estimate of

Displacements	Pt-Pt Error (Pixels)		Pt-Crv Error (Pixels)		Texture Error (Grey-scale)	
	Median	90%ile	Median	90%ile	Median	90%ile
Raw-Regress. (50/eg)	8.3	11.8	5.2	7.6	18.5	23.4
PCA-Regress. (50/eg)	8.3	11.8	5.2	7.6	18.5	23.4
Grad-Regress. (20/eg)	7.8	10.7	4.8	7.0	15.2	19.3
Grad-Regress. (50/eg)	**7.4**	**9.9**	**4.4**	**6.2**	15.6	19.3
Gradient	8.4	12.4	4.7	7.7	**13.8**	**17.9**

Table 4: Comparative performance of different methods of computing update matrix (XM2VTS set images)

the gradient matrix if texture errors are to be reduced. Using more random displacements during the regression training can significantly improve the results.

6 Discussion and Conclusions

We have described and compared several variations on the original AAM matching algorithm. The three alternatives to the basic algorithm all outperformed it when a straightforward test was done. However, a minor modification to the search algorithm (allowing one or more 'forced' iterations in which a step is taken regardless of whether it improves the error) improved the results, with the original AAM then outperforming all others at point location.

Surprisingly, the methods which drove the shape parameters (directly or indirectly) seemed to produce slightly worse localisation accuracy than the basic method (which drives the appearance parameters). The Shape-AAM and the Compositional-AAM performed similarly, and gave slightly better texture matches than the other methods.

It is worth noting that we believe the compositional approach described by Baker and Matthews [14] is a more elegant and mathematically correct method of updating the shape component of the AAM, but found that on our data it did not appear to make much practical difference compared to a simple linear update scheme.

We demonstrated that different methods of learning the update matrix can lead to significantly different results, and that a naive implementation of the original regression algorithm can give worse results than methods which estimate the gradient matrix (either directly or using regression).

The comparisons drawn are likely to depend both on the data and the optimisation regime used. Though we believe we have sensible implementations of each algorithm, there may be tricks we've missed that could significantly affect the results. A more thorough comparison would involve finding the optimal training and search regime for each approach and compare the results of these.

Ideally when presented with a new problem domain, one would try each algorithm on the data of interest with a variety of different search regimes and select the best. However, the results suggest that using the original algorithm is a good default option, as long as one (a) modifies the training method to estimated the gradient (either directly or using a regression approach) and (b) modifies the search regime to include some 'forced' iterations to escape local minima.

References

[1] F. L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):567–585, 1989.

[2] H. Bosch, S.C.Mitchell, P.F.Boudewijn, P.F.Leieveldt, F.Nijland, O. Kamp, M. Sonka, and J. Reiber. Active appearance-motion models for endocardial contour detection in time sequences of echocardiograms. In *SPIE Medical Imaging*, Feb. 2001.

[3] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In H.Burkhardt and B. Neumann, editors, 5^{th} *European Conference on Computer Vision*, volume 2, pages 484–498. Springer, Berlin, 1998.

[4] T. F. Cootes, G. J. Edwards, and C. J. Taylor. A comparative evaluation of active appearance model algorithms. In P. Lewis and M. Nixon, editors, 9^{th} *British Machine Vison Conference*, volume 2, pages 680–689, Southampton, UK, Sept. 1998. BMVA Press.

[5] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:681–685, june 2001.

[6] T. F. Cootes and C. J. Taylor. Constrained active appearance models. In 8^{th} *International Conference on Computer Vision*, volume 1, pages 748–754. IEEE Computer Society Press, July 2001.

[7] G. Edwards, C. J. Taylor, and T. F. Cootes. Interpreting face images using active appearance models. In 3^{rd} *International Conference on Automatic Face and Gesture Recognition 1998*, pages 300–305, Japan, 1998.

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

[9] X. Hou, S. Li, H. Zhang, and Q. Cheng. Direct appearance models. In *Computer Vision and Pattern Recognition Conference 2001*, volume 1, pages 828–833, 2001.

[10] M.B.Stegmann. Active appearance models: Theory, extensions and cases. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, 2000.

[11] M.B.Stegmann and R.Larsen. Multi-band modelling of appearance. In A.Pece, editor, *Workshop on Generative Model-Based Vision*, pages 101–106, Copenhagen, 2002.

[12] K. Messer, J. Matas, J. Kittler, J. Luettin, and G. Maitre. Xm2vtsdb: The extended m2vts database. In *Proc. 2nd Conf. on Audio and Video-based Biometric Personal Verification*. Springer Verlag, 1999.

[13] S. Mitchell, P. Boudewijn, P.F.Lelievedt, R. van der Geest, H. Bosch, J. Reiber, and M. Sonka. Time continuous segmentation of cardiac mr image sequences using active appearance motion models. In *SPIE Medical Imaging*, Feb. 2001.

[14] S.Baker and I.Matthews. Equivalence and efficiency of image alignment algorithms. In *Computer Vision and Pattern Recognition Conference 2001*, volume 1, pages 1090–1097, 2001.

[15] S.Romdhani, A.Psarrou, and S. Gong. On utilising template and feature-based correspondence in multiview appearance models. In 6^{th} *European Conference on Computer Vision*, volume 1, pages 799–813. Springer, 2000.

[16] M. B. Stegmann, R. Fisker, and B. K. Ersbll. Extending and applying active appearance models for automated, high precision segmentation in different image modalities. In *Scandinavian Conference on Image Analysis*, page To appear, 2001.

[17] S.Yand, C.Liu, S.Z.Li, H.Zhang, H.-Y. Shum, and Q.Cheng. Texture-constrained active shape models. In A.Pece, editor, *Workshop on Generative Model-Based Vision*, pages 107–113, Copenhagen, 2002.

[18] T.F.Cootes and C.J.Taylor. On representing edge structure for model matching. In *Computer Vision and Pattern Recognition Conference 2001*, volume 1, pages 1114–1119, 2001.

[19] V.E.Devin and D.C.Hogg. Reactive memories: An interactive talking-head. In T. Cootes and C. Taylor, editors, 12^{th} *British Machine Vison Conference*, volume 2, pages 603–612, 2001.