3D Digitization Project: 3D Tracking using Active Appearance Models

Guillaume Lemaître, Mojdeh Rastgoo and Rocio Cabrera Lozoya
Heriot-Watt University, Universitat de Girona, Université de Bourgogne
g.lemaitre58@gmail.com

***Abstract*—This paper describes a face tracking algorithm using model-based methods and more precisely Active Appearance Models (AAM). First, an introduction on the background theory about model-based methods on machine vision is presented, followed by a description on statistical shape models and statistical models of appearance, which are necessary to develop AAMs. The implementation description, done in a Matlab programming environment, is presented and its final results.**

I. Introduction

Face detection is a necessary step in many applications, ranging from content-based image retrieval and video coding, to intelligent human-computer interfaces, crowd surveillance, biometric identification, and video conferencing [1] . A wide range of face detection techniques exist, from simple algorithms to advanced composite high level approaches for pattern recognition, as the problem has received attention over the last fifteen years.

For offline processing, the technology has reach a point where the face detection problem is nearly closed [1] , although accurate detection of features such as corners of eyes or lips is more difficult. However, face detection in real-time is still an open problem, as the best algorithms are still to computationally expensive for real-time processing.

Face detection algorithms can be classified into feature-based and image-based approaches [1] (figure 1). The former group, feature-based approaches, constitutes the classical detection methodology making explicit usage of face knowledge. The idea of these approaches is not to consider images as global but extract features which will lead to characterize the object, in this case faces. This group can be subdivided into low-level analysis (edges, grey level, colour, motion, or generalised measures), feature analysis (feature searching, or constellation analysis), active shape models (snakes, deformable templates, or point distribution models (PDMs)).

The latter group, image-based approaches, address face detection as a general recognition problem, implicitly incorporating face knowledge through mapping and training schemes. It can be sub-classified into linear subspace methods, neural networks, and statistical approaches.

Active appearance model is a method which connect both approaches (features-based and images-based methods). In fact, shapes can be consider as features extracted from the image and textures as global information from the image.

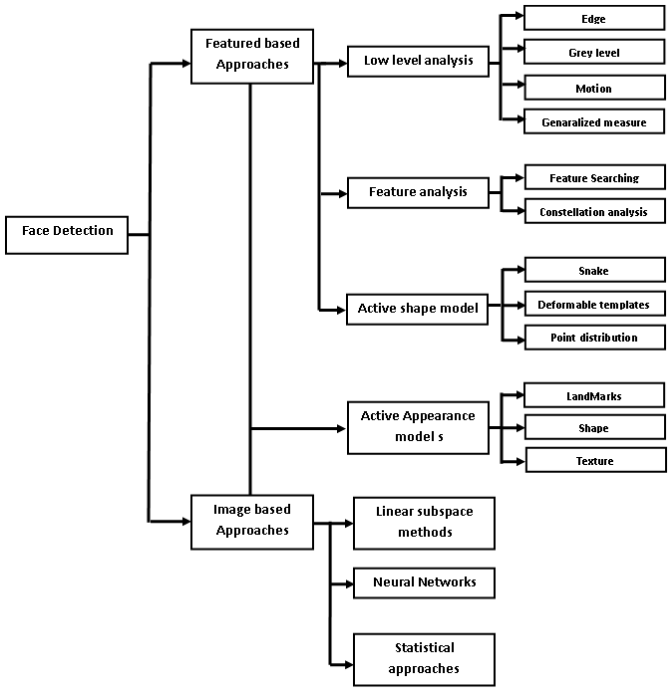

Figure 1. Face detection approaches

In this paper, an introduction about model-based method will be given. Then, creation of the statistical shape models and statistical texture models will be presented. Finally, an overview of the implementation and results will be given.

II. Model-Based Algorithms

Model-based methods have a characteristic of containing prior knowledge of the problem that is to be solved. There is an expected shape of the structures that are to be sought, a spatial relationship between them and probably a grey-level appearance expected pattern [2]. All this prior knowledge helps us pull away from a blind search, because the search can be restricted to only plausible or valid interpretations of the image, those that fit the model.

Model-based methods can also be generative models [2]. Using this models and with the prior information about the object, a realistic image of the target could be built. These methods also are able to deal with large variability, since they can be thought as deformable models [2]. They are general in

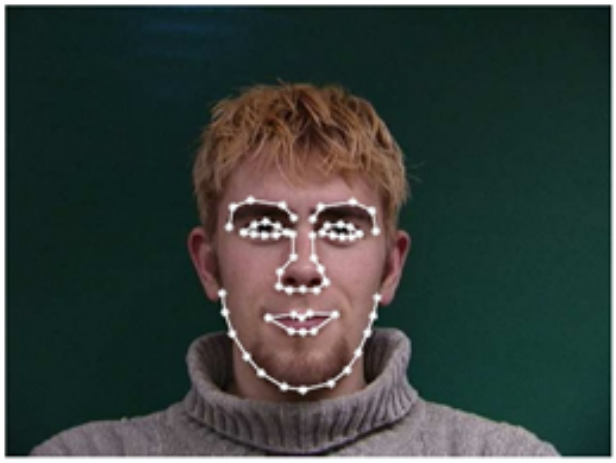

Figure 2. Sample face with 58 manually annotated points

the same time specific enough to allow generation of plausible, valid examples of the class.

Model-based methods are classified as a top-down strategy to solve a problem. They use a prior model to find a best estimation in the image; then, a measurement is developed to evaluate if the target was actually present [2].

This report presents a face tracking system based on active appearance models (AAM).

A. The IMM Face Database

The database used for this project is the *IMM Face Database*, [3], [4], which consists of an annotated set of 240 images of 40 different subjects. The gender distribution is 7 females versus 33 males, all of them which are free of glasses or accessories. The dataset was downloaded from the provided link

$http://www.imm.dtu.dk/\sim aam/$ in [3], [4].

The images were manually annotated with 58 landmarks located on the eyebrows, eyes, nose, mouth and jaw. Figure 2 shows a sample annotated face. Furthermore, the database consisted of six different positions of the same subject, the image types that can be encountered are:

- Full frontal face, neutral expression, diffuse light
- Full frontal face, happy expression, diffuse light
- Face rotated 30 degrees right, neutral expression, diffuse light
- Face rotated 30 degrees left, neutral expression, diffuse light
- Full frontal face, neutral expression, spot light added at the person's left side
- Full frontal face, arbitrary expression, diffuse light

Sample images taken from the database can are shown in Figure 3.

Figure 3. Sample Images from Database

III. Model Generation

This section will describe the theory behind the model generation. As explain in introduction, AAM does not consider only shape but also the texture and combine in an linear way both features. The first part concerns the statistical shape model whereas the second part is about the statistical appearance model.

A. Statistical Shape Models

Faces can be well represented by their shapes which are good features. AAMs will take advantage of this specificity. Shapes are composed of landmarks which are defined at specific locations in the face. However, these shapes can change from one face to another. The aim will be to build a model which will describe either the typical shape or typical variability.

In order to build this shape model, the database presented in section II will be used where each face was annotated with 58 landmarks at specific locations. More explanations are given in [3], [4].

A shape can be seen as a vector $\mathbf{x}$ as:

$$\mathbf{x} = (x_1, ..., x_n, y_1, ..., y_n)^T \tag{1}$$

where $\{(x_i, y_i)\}$ are ordered coordinate pairs from the i^{th} landmark.

1) Align the training dataset: The images were tried to acquired about the same point of view. Thus, shapes are misaligned from one to another. Before to create the shape model, a step which consist to realign all shapes is needed. Cootes and al. proposed to use the most popular approach in the literature being Procrustes Analysis [5], [2]. Procrustes Analysis allows to determine a linear transformation (translation, reflection, orthogonal rotation, and scaling) between two shape by minimizing the sum of distances between these shapes. In order to align the training dataset, one shape will be consider as reference (it could be the first shape of the dataset) and all shapes will be aligned on this shape using the Procrustes Analysis.

2) Modeling shape variation using Principal Component Analysis: The aim of modeling the shape is to find a parameterised model of the form:

$$\mathbf{x} = M(\mathbf{b}_s) \tag{2}$$

where $\mathbf{b}_s$ is a vector of parameters of the shape model.

An effective approach in order to carry information about the shape is to use Principal Component Analysis (PCA). PCA allows to reduce the number of dimensions and keep essential data which are more manageable. Basically equation 2 will be changed into the following equation using PCA:

$$\mathbf{x} = \bar{\mathbf{x}} + \Phi\mathbf{b}_s \tag{3}$$

where $\mathbf{b}_s$ is a vector, containing parameters of the shape model, $\bar{\mathbf{x}}$ is the mean shape and Φ which can be seen as a dictionnary. The following part will explain how to compute Φ in order to make an estimation of the shape $\mathbf{x}$.

In order to create a model, the following steps have to be done which are just PCA steps:

- Compute the mean shape:

$$\bar{\mathbf{x}} = \frac{1}{s}\sum_{i=1}^{s}\mathbf{x}_i \tag{4}$$

  where $\mathbf{x}_i$ is the i^{th} shape.
- Compute the covariance matrix:

$$\mathbf{S} = \frac{1}{s-1}\sum_{i=1}^{s}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T \tag{5}$$

  For computation efficiency, the following trick can be applied:
  - Substract the mean shape from the data in order to be able to compute the scatter matrix

$$\mathbf{x}_i = \mathbf{x}_i - \bar{\mathbf{x}} \tag{6}$$

  - Having a mean equal to zero, it is simple to compute the scatter matrix as:

$$\mathbf{S} = \mathbf{X}\mathbf{X}^t \tag{7}$$

    where $\mathbf{X}$ is the matrix of the different shapes as $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_s\}$
- Compute the eigenvectors φ_i and eigenvalues λ_i of the scatter matrix $\mathbf{S}$. The eigenvectors φ_i have to be sorted regarding the values of the eigenvalues λ_i. Then, Φ and Λ are defined as:

$$\Phi = \{\varphi_1, ..., \varphi_n\} \tag{8}$$
$$\Lambda = \{\lambda_1, ..., \lambda_n\} \tag{9}$$

  At this point, any shape of the training dataset can be computed using the equation 3. However, at this point the number of dimension is n. The main goal of PCA is to reduce a maximum number of dimension to have data, which is more manageable and in the same time keep as much information as possible. In order to obtain a good compromise, the new number of dimension k is computed as:

$$\sum_{i=1}^{k}\lambda_i = f_p\sum_{i=1}^{n}\lambda_i \tag{10}$$

  where f_p is the proportion of the total variation. Usually, this number is around 0.98 or 0.99. Reducing the number of dimensions, equation 3 can be written as:

$$\mathbf{x} \approx \bar{\mathbf{x}} + \Phi\mathbf{b}_s \tag{11}$$

  with Φ defined as:

$$\Phi = \{\varphi_1, ..., \varphi_k\} \tag{12}$$

A problem with this PCA scheme is that if the number of dimension is really big, it will be impossible in practise to compute the scatter matrix S. A small size trick can allow to compute the dictionnary Φ without computing the scatter matrix S. The steps are presented below:

- Compute the mean shape:

$$\bar{\mathbf{x}} = \frac{1}{s}\sum_{i=1}^{s}\mathbf{x}_i \tag{13}$$

  where $\mathbf{x}_i$ is the i^{th} shape.
- Compute the matrix T:
  - Substract the mean shape from the data in order to be able to compute the scatter matrix

$$\mathbf{x}_i = \mathbf{x}_i - \bar{\mathbf{x}} \tag{14}$$

  - Having a mean equal to zero, it is simple to compute the scatter matrix as:

$$\mathbf{T} = \mathbf{X}^t\mathbf{X} \tag{15}$$

    where $\mathbf{X}$ is the matrix of the different shapes as $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_s\}$
- Compute the eigenvectors ψ_i and eigenvalues λ_i of the scatter matrix $\mathbf{S}$. The eigenvectors ψ_i have to be sorted

Figure 4. Effect of varying each of the three first parameters of the vector **b** idenpendently of $\pm 3\sqrt{\lambda_i}$ where λ_i corresponds to the i^{th} parameter.

regarding the values of the eigenvalues λ_i. Then, Ψ and Λ are defined as:

$$\Psi = \{\psi_1, ..., \psi_n\} \quad (16)$$

$$\Lambda = \{\lambda_1, ..., \lambda_n\} \quad (17)$$

- Convert the eigenvectors computed from $\mathbf{X}^t\mathbf{X}$ to $\mathbf{X}\mathbf{X}^t$:

$$\varphi_i = \frac{1}{\sqrt{\lambda_i}}\psi_i \quad (18)$$

and

$$\Phi = \{\varphi_1, ..., \varphi_n\} \quad (19)$$

At this point, any shape of the training dataset can be computed using the equation 3. However, at this point the number of dimension is n. As is mentioned before, the main goal of PCA is to reduce the number of dimensions, in order to have more manageable data and much information as possible. In this case the new dimension k is computed as:

$$\sum_{i=1}^{k} \lambda_i = f_p \sum_{i=1}^{n} \lambda_i \quad (20)$$

where f_p is the proportion of the total variation. Usually, this number is around 0.98 or 0.99. Reducing the number of dimensions, equation 3 can be written as:

$$\mathbf{x} \approx \bar{\mathbf{x}} + \Phi \mathbf{b}_s \quad (21)$$

with Φ defined as:

$$\Phi = \{\varphi_1, ..., \varphi_k\} \quad (22)$$

3) Example of shape models: Using the database presented in section II, the dictionnary Φ was computed keeping a proportion of 0.98 of the sum of the eigenvalues Λ. Figure 4 presents the effect of varying the three first parameters of the vector **b** idenpendently with a value of $\pm 3\sqrt{\lambda_i}$ where λ_i corresponds to the i^{th} parameter.

B. Statistical Appearance Model

In order to have a complete and realistic image of objects or faces both shape and texture of the objects should be modelled. Appearance models are based on variation of shape and texture models as well as correlations between them. Appearance model is the combination of the shape model with texture model on a normalized shape frame. Texture is defined as a pattern of intensities or colours across the image patch [5]. In order to obtain the appearance model a training set of labelled images are required [5]. The mean shape of training set could be obtained based on statistical shape model, which was explained in previous section. In the next step, wrapping each training example to the corresponding mean shape will result in free patch image. The texture model then could be built based on variations of free patches.

Different stage of texture model and final appearance model are explained in the following.

1) Texture models: As it mentioned before texture models are based on intensities or colour variations over image patches. In the first step each example image is wrapped to the corresponding mean shape model. In the second step the obtained result from wrapping are normalized and in the third step the final texture model is obtained from PCA on the normalized data. Detail explanation of three main step of texture modelling, are listed in the following:

- Image Wrapping:
  Wrapping each image to its mean shape will remove specious texture variations. These variations would be the result of eigenvalue decomposition on not normalized data [5]. The basic idea of image wrapping is to map control points in one image to another image. The points are mapped from the original position x_i to new position $x_i^{'}$ based on continuous mapping function f, Equation 23. Using this function all the points in the first image could map to the second image; however it is likely to find some holes in the mapped image [5].

  In practice, in order to avoid these holes, the reverse map function, $f^{'}$ have been used. The reverse map function will find the originated point of each pixel $x_i^{'}$ in mapped image. It should be considered that reverse map function is not necessary the inverse version of f; however this could be good approximation [5]. Several wrapping functions could be used such as piece wise affine function and plate spline interpolator. The piece wise affine function was used in this practice.
  Piece wise affine function is the simplest wrapping function based on the assumption that each f_i is linear in the local regions and zero every where else [5]. Based on this assumption for 2D scene, triangulation could be used to partition the image into smaller regions. The affine motion then could be used to map the corners of each triangle to their new positions in the mapped image. Triangulation is based on $Delauney$ method, which is already implemented by Matlab functions.

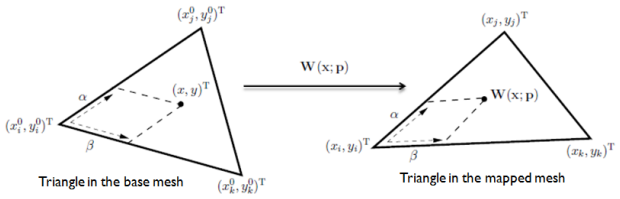

Figure 5. Piece wise affine wrap

$$f(x_i) = x_i^{'}, \forall i = 1 \dots n \tag{23}$$

Considering the triangulation mesh, every pixel,x in image lies in one triangle. Let us assume $(x_i^0, y_i^0)^T$, $(x_j^0, y_j^0)^T$ and $(x_k^0, y_k^0)^T$ as three vertices of the first triangle [6], which correspond to $(x_i, y_i)^T$, $(x_j, y_j)^T$ and $(x_k, y_k)^T$, three vertices of mapped triangle in the new image. The two triangles are shown in Figure (5).
With reference to Figure 5,[6], each point in the original triangle such as, $x = (x, y)^T$ can be defined in accordance to three vertices of the triangle by Equation 24, where α and β are defined as Equations 25 and 26 [6].

$$\begin{aligned} x &= (x_i^0, y_i^0)^T + \alpha[(x_j^0, y_j^0)^T - (x_i^0, y_i^0)^T] \\ &+ \beta[(x_k^0, y_k^0)^T - (x_i^0, y_i^0)^T] \end{aligned} \tag{24}$$

$$\alpha = \frac{(x - x_i^0)(y_k^0 - y_i^0) - (y - y_i^0)(x_k^0 - x_i^0)}{(x_j^0 - x_i^0)(y_k^0 - y_i^0) - (y_{j^0} - y_i^0)(x_k^0 - x_i^0)} \tag{25}$$

$$\beta = \frac{(y - y_i^0)(x_j^0 - x_i^0) - (x - x_i^0)(y_j^0 - y_i^0)}{(x_j^0 - x_i^0)(y_k^0 - y_i^0) - (y_{j^0} - y_i^0)(x_k^0 - x_i^0)} \tag{26}$$

The affine transformation then could be used to map each point, $x = (x, y)^T$ to its corresponding point, Equation 27. This equation can be simplified in terms of Equation 28. Considering equation 28, the six parameters could be obtained from equations 24, 25, 26 and 27, once per triangle [6]. In general the computation of piece wise affine wrap could be structured based on it's pseudocode [6].

$$\begin{aligned} W(x;p) &= (x_i, y_i)^T + \alpha[(x_j, y_j)^T - (x_i, y_i)^T] \\ &+ \beta[(x_k, y_k)^T - (x_i, y_i)^T] \end{aligned} \tag{27}$$

$$W(x;p) = (a_1 + a_2.x + a_3.y, a_4 + a_5.x + a_6.y)^T \tag{28}$$

Figure 6 shows obtained triangulated results with the land marks on three first mean shapes, while λ is changing between -3 and $+3$.

- Normalization:
The normalization is required, in order to minimize the effects of global lighting variations [5]. The scaling factor α and offset factor β are used to normalize example samples [5]. The texture vector g_{im}, then could be normalized based on Equation 29 [5] , where the values of α and β are based on mean of normalized data $\bar{g}$. These values should be chosen in way that sum of elements will be zero and the variance, unity. Based on this definition α and β can be defined as Equation 30 and 31, where n is number of elements in the vector.

$$g = (g_{im} - \beta 1)/\alpha \tag{29}$$

$$\alpha = g_{im}.\bar{g} \tag{30}$$

$$\beta = (g_{im} 1)/n \tag{31}$$

- PCA:
A linear model could be obtained by applying PCA on the normalized data. By using PCA the texture model could be defined as Equation 32, where g is the mean normalized grey level vector, P_g is the set of eigenvectors representing texture models and b_g is the set of grey level parameters [5].

$$g = \bar{g} + P_g b_g \tag{32}$$

Figure 7 shows the results obtained for the texture models, based on the three first mean faces and their corresponding variations of $+3$ and -3, λ.
In the same way, the model can be computed for color images as shown in figure 8. Color texture modeling will

Algorithm 1 Pseudocode of Image wrapping - Piece wise affine function

```
Given the shape parameters
for Each shape model and its parameters do
    Compute the triangulation
    Compute (x_i, y_i)^T for all the vertices in triangulated mesh
    Compute, (a_1, a_2, a_3, a_4, a_5, a_6) for each triangle
    for Each pixel in the original mesh do
        found the triangle it belongs
        Find the corresponding six values ,(a_1, a_2, a_3, a_4, a_5, a_6)
        Compute the wrap using Equation 27
    end for
end for
```

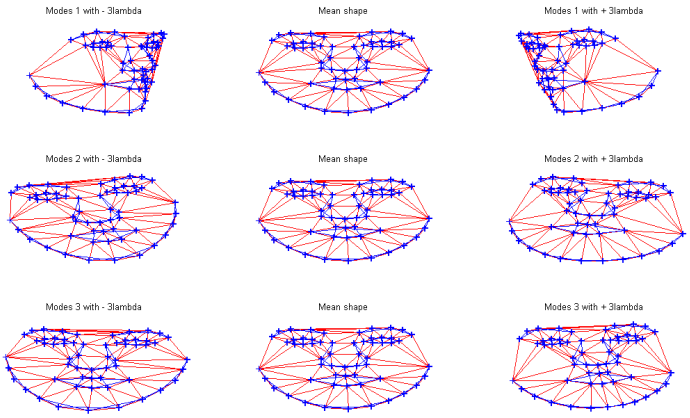

Figure 6. Triangulated results with the landmarks

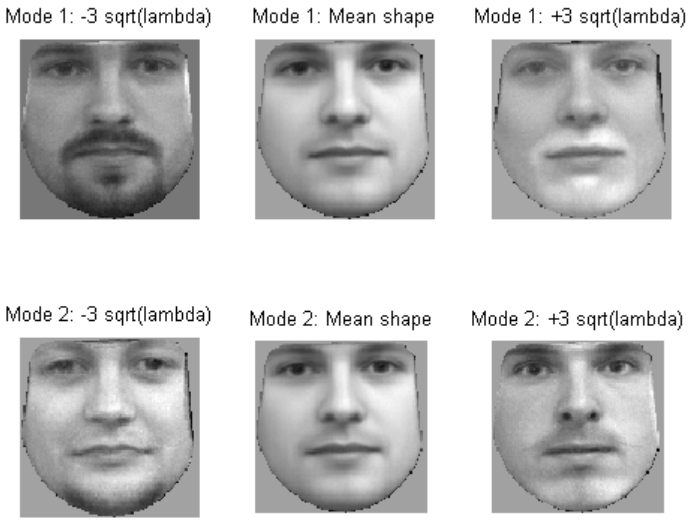

Figure 7. Grayscale texture models

Figure 8. Color texture models

be memory problem because size of the image is multiply by the number of channels.

2) Appearance model: As it was mentioned before since there is correlation between texture and shape models, the final appearance model is based on their combination. In this case the appearance model could be obtained by applying PCA on the combined results. Considering b_g and b_s as parameter vectors of texture and shape respectively, an integrated vector b can be defined based on their combination, Equation 33.

The value W_s is a diagonal matrix of weight for each shape parameters [5]. This matrix is required based on the fact that b_g and b_s are representing different parameters. The texture model is based on intensity variations while the shape model is in accordance to distance unit. In order to combine two parameters the variation effect of b_s on sample g should be considered [5]. The value W_s could be obtained from element displacement of b_s with reference to its optimum value in each training example. The RMS change in g per unit change in

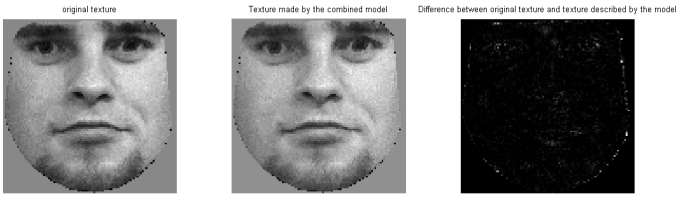

Figure 9. Left: texture model, Center: Appearance model, Right: Difference between both images

shape parameters, b_s will give the appropriate W_s. The weight also could be calculated based on the ration of total intensity variation to total shape variation.

$$b = \begin{bmatrix} W_s b_s \\ b_g \end{bmatrix} = \begin{bmatrix} W_s P_s^T (x - \bar{x}) \\ P_g^T (g - \bar{g}) \end{bmatrix} \tag{33}$$

PCA cab be further applied to this integrated vectors, b as it is shown in Equation 34. Vector P_c stands for eigenvectors and c is the vector of appearance parameters controlling both shape and texture models. This means that shape and grey level models(texture)could be expressed as a function of c.

$$b = P_c c \tag{34}$$

Figure 9 shows the results obtained for the appearance model. First image on the right is texture model obtained from previous section, while the center image represent the combined image(Appearance model) and the last image is the difference between both images.

IV. Implementation

The active appearance model was implemented in two main steps, training and testing. A multi scale implementation was used in order to improve the robustness and efficiency of the framework. The multi scale framework was used both in training and testing stage.

In multi resolution frame the algorithm first will look for the object in the coarse image. In the next steps it will refine the locations in finer scales of the image [5]. In each multi scale frames the base image is the original image, the next scale image is obtained by smoothing the original image while it it's size is reduced to half. The number of scales could be adjusted based on the user interests; however the computational cost will increase in higher scales. This section contains the pseudocode of the training stage as well as testing stage.

V. Active appearance model results

Figure 10 and 11 present the AAM result. However, result shown on figure 11 is actually better as shown after that some bugs were fixed.

Regarding the tracking task, AAM can be used just modifyinf the strategy of searching the model without any multiresolution approach.

Algorithm 2 Pseudocode: Training Stage

Load training data
for Scale 1 : N **do**
Computation of Shape models
- Align shapes with Procrustes Analysis
- Obtain main directions of variations with PCA
- Keep the 98% most significant eigenvectors

Computation of Texture models
- Transform face image into mean texture image (Image wrapping)
- Normalize the grey scale, to compensate for illumination
- Perform PCA
- Keep the 99% most significant eigenvectors

Computation of Combined shape(Appearance Model)
- Addition of shape and texture models
- Perform PCA
- Keep the 99% most significant eigenvectors

Search model
- Find object location in the test set
- Training done by translation and intensity difference computation (keep position with smallest difference)

Transform image to Coarser scale
end for

Algorithm 3 Pseudocode: Test Stage

Manual Initialization
for Scale 1 : N(Start in Coarser scale) **do**
Adopting the model for the current scale
Scale the image
Search iterations
for Number of iterations **do**
Sample image intensities
Compute the difference between model and real image intensities
if Previous error less than current error **then**
Go to the previous location
else
Update the error
end if
end for
Next finer scale
end for
Show the result

VI. Discussion and Conclusion

The active shape model was developed with reference to the well known AAM model presented by [5]. In order to improve the results for more robust algorithm, multi resolution scheme was implemented in both testing and training stage. An iterative steps was also added in the testing stage in order to fit the algorithm to face images. Given the initial starting position the search will converge quickly and computation will not take long time, considering the right training provided. In the other hand training stage appears time consuming. The expensive computational time, is most probably due to MATLAB environment. Finally concerning real time application, the proposed algorithm might fail, due to the long computation time.

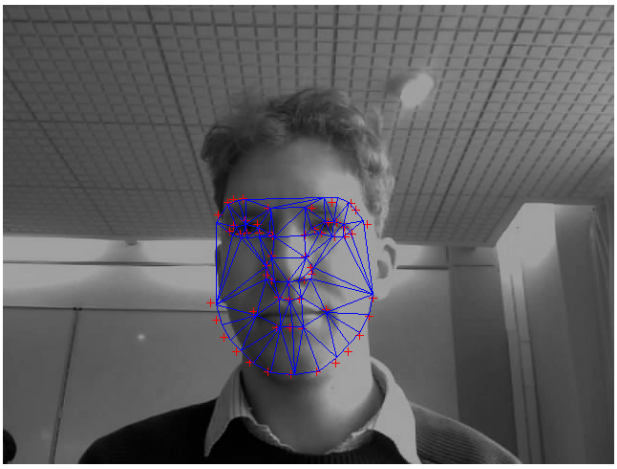

Figure 10. AAM fitting on an image which was not use for the training

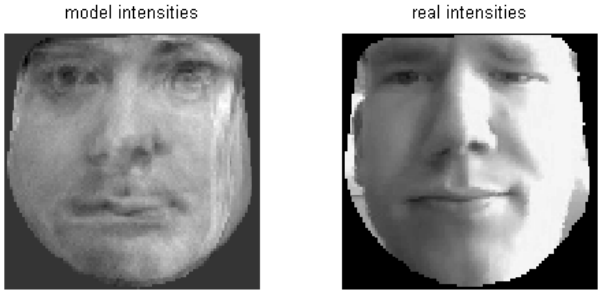

Figure 11. Texture modeling on an image which was not use for the training

Comparing active shape models with active appearance models it was proved that AAM provide more robust results and relatively better performance, however the algorithm still have some difficulty in terms of occluded faces and it will fail in terms of texture models.

Implemented algorithm does not provide satisfied results in terms of face tracking. The main problem is due to the texture models obtained in training stage. Due to the computational cost of training stage the texture models obtained for current tracking stage are not suitable enough.

References

[1] E. Hjelmås and B. K. Low, "Face detection: A survey," *Computer Vision and Image Understanding*, vol. 83, no. 3, pp. 236–274, 2001.
[2] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *Proceedings of the European Conference on Computer Vision*, vol. 2, pp. 484–498, 1998.
[3] M. B. Stegmann, "Analysis and segmentation of face images using point annotations and linear subspace techniques," Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, Tech. Rep., 2002. [Online]. Available: http://www.imm.dtu.dk/ aam/

[4] M. B. Stegmann, B. K. Ersbøll, and R. Larsen, "FAME – a flexible appearance modelling environment," *IEEE Trans. on Medical Imaging*, vol. 22, no. 10, pp. 1319–1331, 2003.
[5] T. Cootes, C. Taylor, and M. M. Pt, "Statistical models of appearance for computer vision," 2000.
[6] I. Matthews and S. Baker, "Active appearance models revisited," *International Journal of Computer Vision*, vol. 60, pp. 135–164, 2003.