IT 09 042
Examensarbete 30 hp
September 2009

Image Analysis for Marker-less Facial Tracking

Wenlan Yang

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Image Analysis for Marker-less Facial Tracking

Wenlan Yang

Tracking of facial features is increasingly used in game and film industry as well as for other applications. Most tracking systems are currently using markers which unfortunately are tedious and cumbersome. Marker-less facial tracking is supposed to eliminate the disadvantages of the marker-based approaches. This thesis investigates different algorithms for marker-less tracking and presents how to apply them in a robust way. View-based and component sensitive normalized face images can achieve accurate tracking results based on the Active Appearance Algorithm. Post processing the parameters of global motions of the model smoothes the synthesized video sequence. Tacking results for faces and a tool developed for creating training images are also presented.

Key Words: Marker-less, Texture registration, Active Face Model, Active Appearance Algorithm, View-based texture model, Component sensitive, Post-processing.

Handledare: Christian Sjöström
Ämnesgranskare: Ewert Bengtsson
Examinator: Anders Jansson
IT 09 042
Sponsor: Imagination Studios

Tryckt av: Reprocentralen ITC

Acknowledgement

It has been a pleasure that doing this thesis in Imagination Studios for six mouths. First of all, I would like to thank my supervisor Christian Sjöström and reviewer Prof. Ewert Bengtsson, always giving me advices, helps and good ideas. Also thanks to Filip Malmberg who helps me by emails in summer. Without them, I could not finish the thesis alone. Finally, I would like to thank staff of Imagination Studios for their kindness and patience.

Table of Contents

1. Introduction

This chapter introduces the background of marker-based motion capture, after that, a brief definition of thesis motivation and purpose will be presented with related technologies. At last, a short overview of the rest of the chapters is given which provides an outline to the readers.

1.1 Marker-based motion capture

Nowadays, motion capture is a very high-demand industry for gaming, filming, medicine, virtual reality and so on. Meanwhile it prompts manufacturers to produce capturing systems with more stability, accuracy, efficiency, flexibility and less cost.

Marker-based motion capturing is so far the most widely used and mature technology applied in this industry. A bunch of markers being placed on feature points of body or face, such as hand, elbow, foot, knee, eye corner, lip are tracked throughout capturing. The position of a reflected marker in 3D space can be determined which depends on several cameras shooting simultaneously. Each marker is identified in the vision system and calculated in each frame to get its motion locus. So animations can be synthesized by the parameters of those markers.

Figure 1.1: Marker-based body motion capture sample[1] for EA DICE[2] - "BAD COMPANY"

[1] © Copyright 2009 Imagination Studios AB. All rights reserved.

[2] www.dice.se

Figure 1.2: Marker-based facial tracking sample[3]

1.2 Marker-less motion capture

However, the equipments of optical motion capture system are fairly expensive and less portable; for those companies that intend to improve the way of execution, the cost of transition is very high.

Another purely image-based analysis technology is researched to conquer the flaws above. Using machine vision directly to obtain motion parameters from images without extra auxiliary markers is the crucial concept for marker-less motion capture. This kind of systems is lightweight and portable, quite popular due to its low cost, especially in case only one camera is required. But the robustness and accuracy are normally much lower than for the optical systems described in 1.1.

In this thesis, we mainly focus on marker-less facial tracking which literally tracks human face only without any markers, not only tracking the global motions but also many features in the face so that facial expressions can be followed and animated. This approach is now attracting considerable interest. Active Appearance Modeling (AAM) proposed by Cootes et al. [1][2][8] is one of the most popular methods to achieve good tracking results. Before, Cootes et al. [3] also presented Active Shape Model (ASM), a statistical shape modeling technique to describe the 2D shape of an object. So AAM combined ASM and the eigen-face [4] technique leading to a Point Distribution Model (PDM) [5] achieves high-accuracy face reconstruction with widely varying pose and illumination.

[3]

The global motion of the head is estimated by using a model-based algorithm. Some head-like geometric shapes are used to track faces; the 3D face data sets could be mapped to a cylinder [6] or an ellipsoid [7]. Therefore, the correspondences between 3D point data sets can be established by minimizing the difference of texture or optical flow between observation and their model.

AAM can not only be applied to facial tracking but is also an important method for locating deformable objects in many applications like organ recognition or pathological change detection.

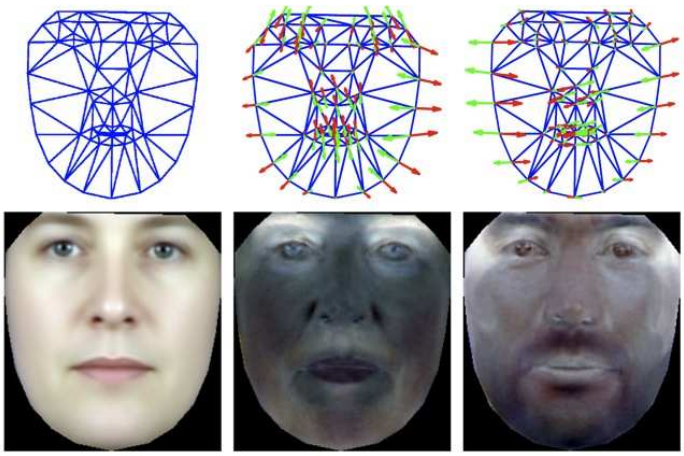


Figure 1.3: Active Appearance Models

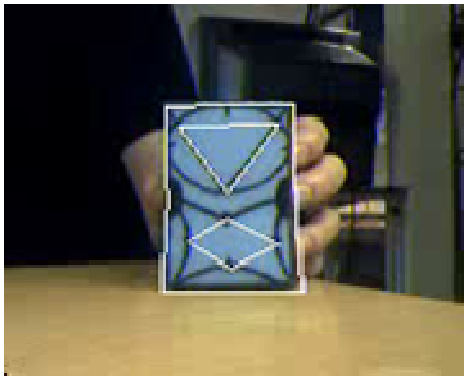


Figure 1.4: Non-face object tracking using AAM[4]

[4] by Mikkel B. Stegmann

1.3 Thesis motivation and purpose

Imagination Studios[5] is a leading motion capture company in Sweden. They use a marker-based approach to track character's body or facial motion. However, the whole set-up for facial tracking is time-consuming; a large number of markers need to be put on the character face which costs over 2 hours each time.

Methods presented by B. Holmberg [9] show the possibility of applying skin texture as virtual markers for estimating human limb motion. It should be possible to use that approach also for human face tracking. J. Ahlberg [10] gives a promising tracking results using active face model.

The task of this thesis project is to do the research and implementation of marker-less facial tracking based on [9][10]. All the work is carried out at Imagination Studios. It is possible that the project encounters unexpected severe difficulties, in that case investigation as to why the methods did not work and any proposals for alternative solutions can be a result of the thesis.

1.4 Thesis overview

Using skin texture as virtual markers for facial feature tracking will be introduced in chapter 2. In chapter 3, the critical part of this thesis, Active Face Model (AFM) and Active Appearance Algorithm (AAA) are studied. Possible improvements of the tracking algorithm are described in Chapter 4-5.

[5] www.nlstudios.biz

2. Facial Feature Tracking using Skin Texture

Feature point tracking is a hot topic in the research community of marker-less motion capture. B. Holmberg [9] estimates three dimensional motion of a human limb using a large number of matched skin texture image patches. In this chapter, the method applied to facial feature tracking is described and tested and it is concluded whether it is suitable for human face.

2.1 Material and calibration

A video camera Sony DCR-HC23E is used for our study which has a resolution of 720×576 pixels and runs at 25 frames per second.

The calibration is conducted according to [11] with a 9×7 chessboard pattern in the camera capturing volume. It provides the estimates of various distortions in the camera for which could be approximately compensated by linearization. For easiness, we display the chessboard pattern on an LCD instead of printing it out since distortions also exist in printers.

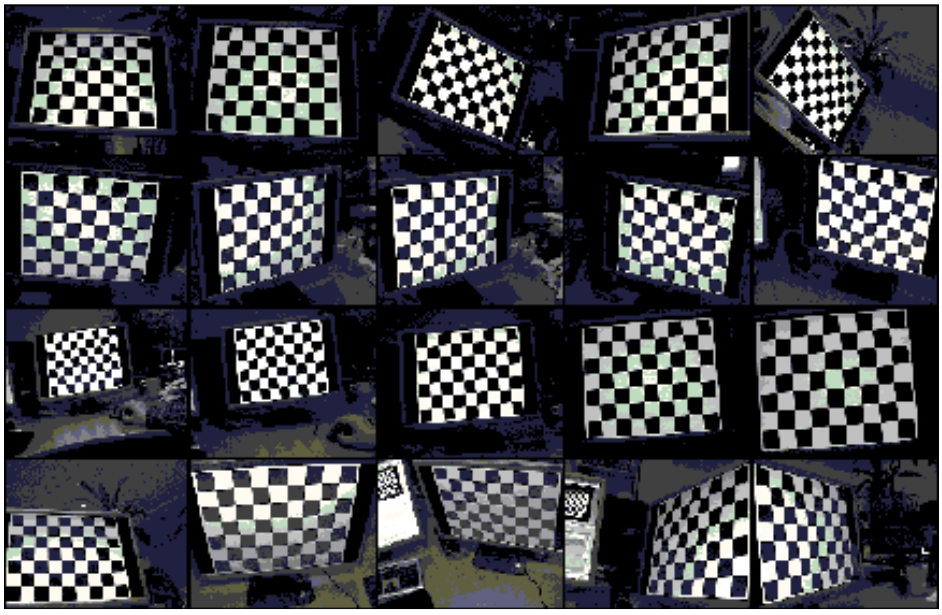


Figure 2.1: 20 chessboard images for calibration

Focal length (fc)	Principal point (cc)	Skew coefficient (alpha_c)	Distortions (kc)
1018.874 pixels	336.226 pixels	0	-0.3681
1120.137 pixels	248.765 pixels		1.5805

			-0.0003
			-0.0031

Table 2.1: Camera internal calibration parameters described in detail in [11]

2.2 Texture matching through 2D correlation

The image patch registration between two adjacent inner-frames is determined by the two dimensional correlation. The two dimensional correlation value between matrix A and B is calculated according to [12] as

$$r = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{(\sum_m \sum_n (A_{mn} - \bar{A})^2)(\sum_m \sum_n (B_{mn} - \bar{B})^2)}}$$

where $\bar{A}$ and $\bar{B}$ are the total matrix means. A_{mn} is the element on the row m and column n in A.

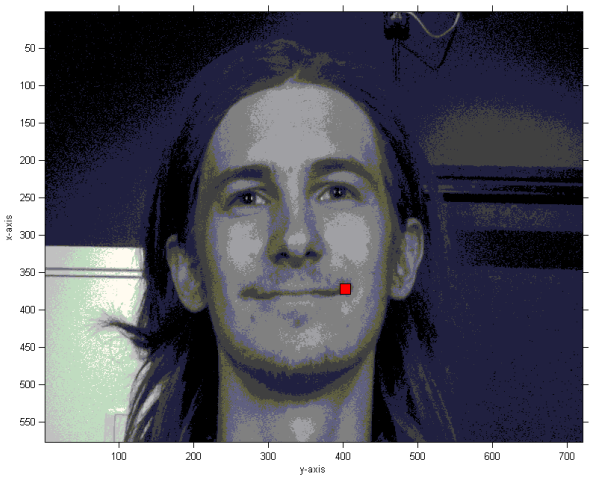


Figure 2.2: A mouth corner image patch is extracted from a random frame of the image sequences

A 15×15-pixel image patch locating at the right mouth corner is extracted from an image of resolution 720×576 pixels. The red area is correlated over a 25×25-pixel reference image area that contains the actual patch location.

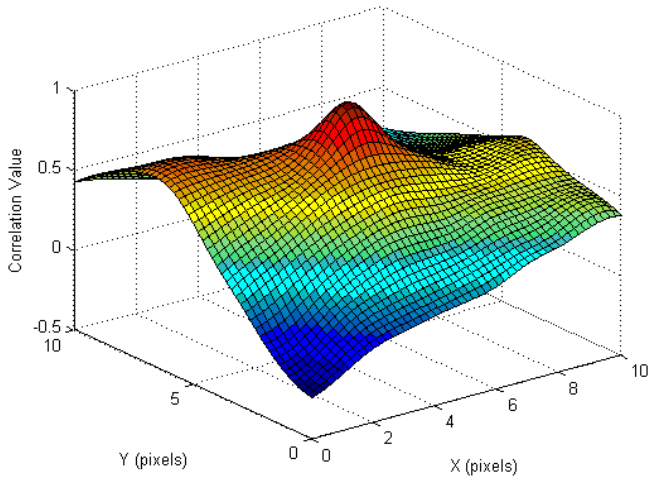


Figure 2.3: 2D correlation values of same image

A small peaky correlation surface is obtained from the example. The peak correlation equals to one since the same image is used for both patches. Next, we compute the cross-correlation between two adjacent images in the video sequence to register the image patch.

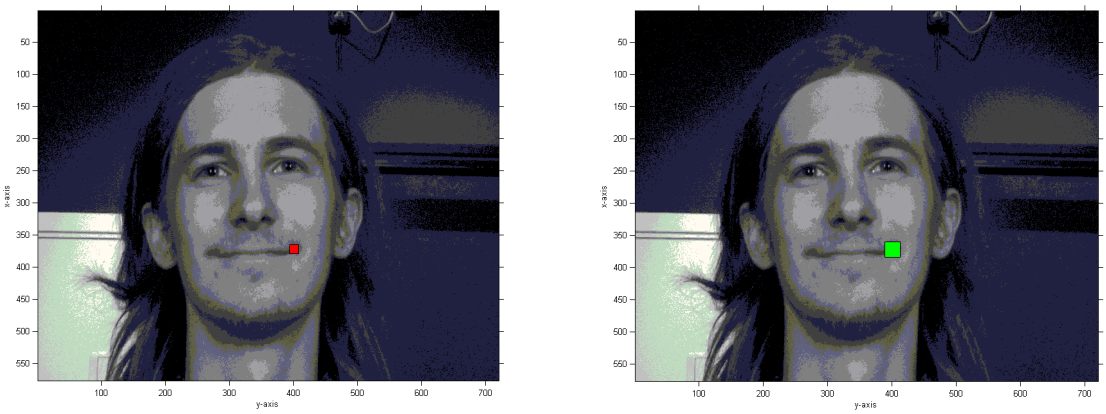


Figure 2.4: Image patch in frame k and reference search area in frame k+1

The green block is the same as described above, a 25×25-pixel reference image area. Within the green region, we calculate the 2D correlation value between selected skin texture (i.e., red image patch) and reference skin texture (i.e., sub-image within green region) for each corresponding position from top-left to bottom-right.

We assume that the variation of head pose, expression or lighting condition will be small enough to match the texture with neighbors. Thus, facial feature points

are chosen as original textures because they could provide good correlation results comparing to rather flat facial area.

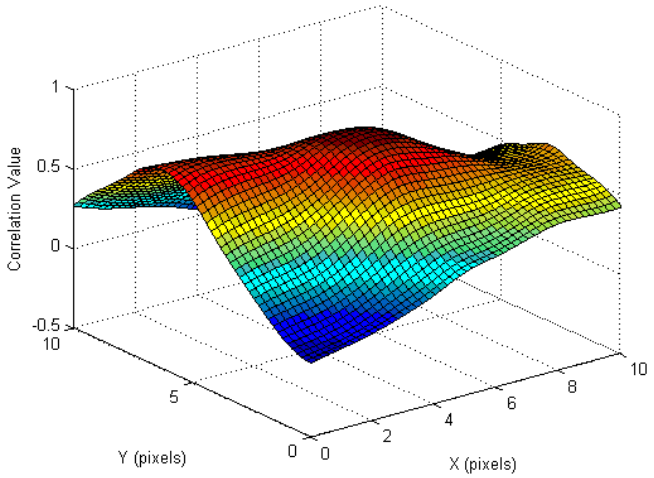


Figure 2.5: 2D correlation values of two adjacent images

Figure 2.5 shows a quite plane correlation surface between image patches taken from frame k and k+1 in the same camera which could result in poor registrations and unstable texture tracking. Therefore we didn't get the expected results presented according to [9]. We need a better objective function for stable tracking.

2.3 Alternative measurements of texture similarity

2.3.1 Mutual information criterion

Mutual information which is widely used in the information theory area could be applied as a measurement of similarity for image registration. Formally, the mutual information of two images A and B can be defined as

$$I(A,B) = \sum_{a,b} p_{AB}(a,b) log \frac{p_{AB}(a,b)}{p_A(a)p_B(b)}$$

where $p_{AB}(a,b)$ is the joint probability distribution function of A and B, and $p_A(a)$ and $p_B(b)$ are the marginal probability distribution functions of A and B respectively. The two image are registered when $I(A,B)$ reaches its maximum value. Mutual information can be equivalently expressed as

$$I(A,B) = H(A) + H(B) - H(A,B)$$

where $H(A)$ and $H(B)$ are the marginal entropies, and $H(A,B)$ is the joint entropy of A and B. We tested the mutual information criterion but found no big difference of tracking result as compared to 2D correlation.

2.3.2 N_4 filter & L_1 distance

Another more straightforward measurement was tested which applies N_4 filter to each pixel within the image patch and computes L_1 distance to find the optimal match. Each pixel is treated as

X_1	X_2	X_3
X_4	X_5	X_6
X_7	X_8	X_9

$$X_5' = 4X_5 - X_2 - X_4 - X_6 - X_8 \,.$$

L_1 distance defined as

$$L_1 = \sum_{k=1}^{n} |p_k - q_k|$$

where p and q are corresponding image patches respectively. The minimum value of L_1 should be considered as registration.

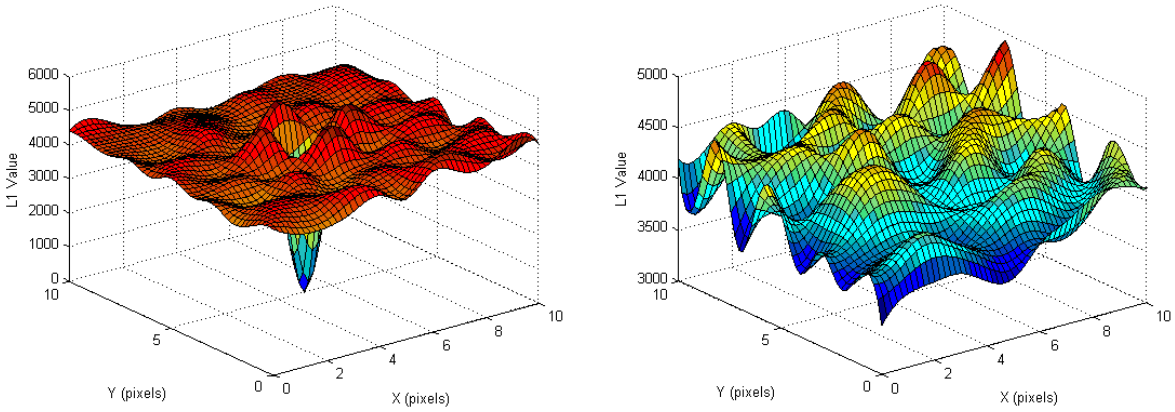


Figure 2.6: L1 distance values of same image and two adjacent images

No obvious peak could be found from second picture in Figure 2.6 when matching the image patch to the next frame. Furthermore, the result is less robust than both 2D correlation and mutual information.

2.4 Possible improvements

2.4.1 Gray level appearance

Studying and modeling the gray level information may help us finding the objective function to improve the skin texture matching. We can extract the gray level profile for the region around a landmark. The gray level profile g_{ij} consists of n pixels along a line passing through the landmark, here we just focus on the horizontal information. The normalized derivative of g_{ij} gives us invariance to offsets and uniform scaling [13].
The gray level profile of landmark j in image i as

$$g_{ij} = [g_{ij0}\ g_{ij1} \cdots\ g_{ijn-1}]^T .$$

The derivative profile is given by

$$dg_{ij} = [g_{ij1} - g_{ij0}\ g_{ij2} - g_{ij1} \cdots\ g_{ijn-1} - g_{ijn-2}]$$

and the normalized derivative profile becomes

$$y_{ij} = \frac{dg_{ij}}{\sum_{k=0}^{n-2}|dg_{ijk}|} .$$

The mean of the normalized derivative profile of landmark j is

$$\overline{y_j} = \frac{1}{N}\sum_{i=1}^{N} dg_{ij} .$$

So the covariance matrix is given by

$$C_{yj} = \frac{1}{N}\sum_{i=1}^{N}(y_{ij} - \overline{y_j})(y_{ij} - \overline{y_j})^T .$$

Using these gray scale statistics, we can find the desired movements to adjust position of the image patch.

2.4.2 Derivative profile

The derivative profile dg_{ij} described in 2.4.1 is a weaker condition to improve the tracking result by searching for strong edges [3]. We select a point along the profile at the strongest edge, but sometimes the true point may not be the one on the strongest edge. Modeling local structure can help locate the point.

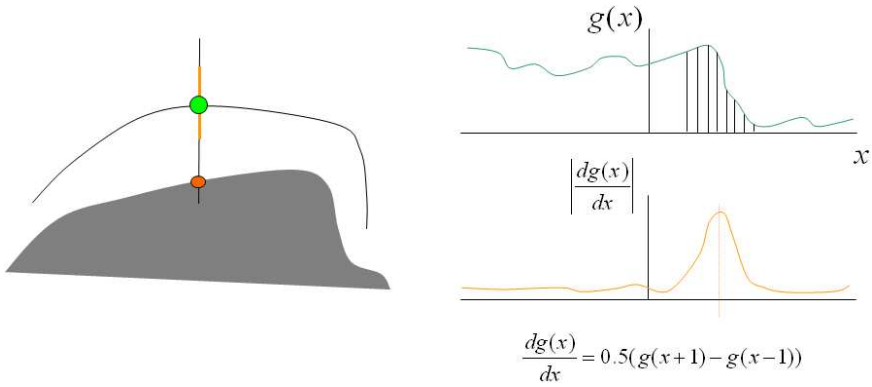


Figure 2.7: Find strongest edge by first derivative

2.5 Tracking experiments

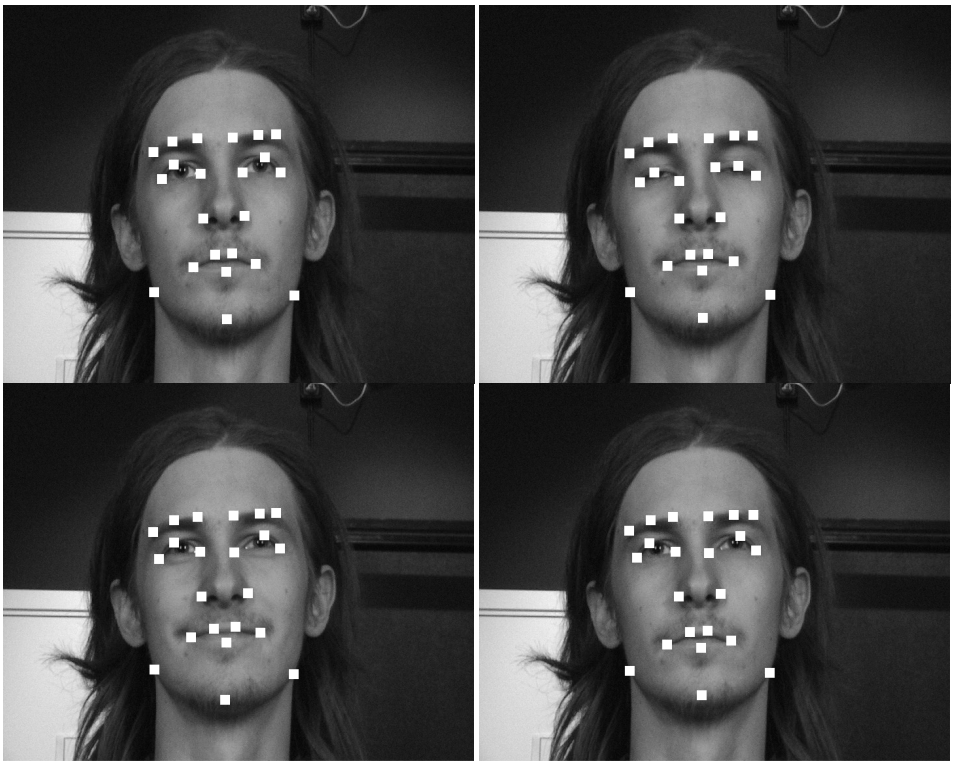
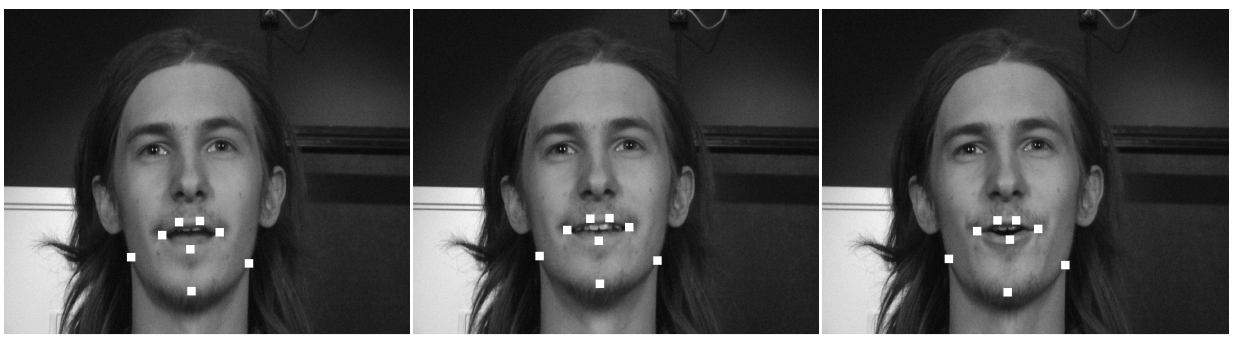


Figure 2.8: Facial tracking (smile) by 22 virtual markers



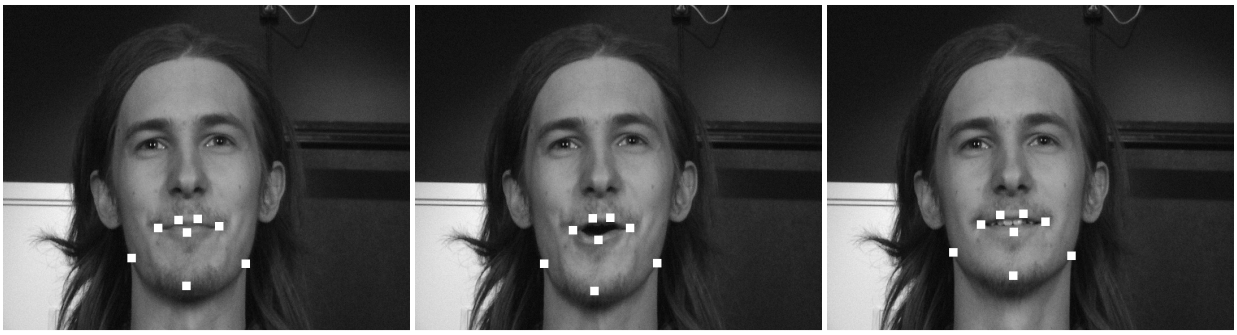


Figure 2.9: Facial tracking (talking) by 8 virtual markers

2.6 Conclusions

For now, it seems not enough to track the facial motion purely by using skin texture as virtual markers. Unstable locating and miss matching are main problems with this strategy. Soft tissue movement, illumination variation and other factors like fps and resolution will affect the result though some issues can be minimized by modeling appearance variation and equipments setup. Another limitation is that we can only track a frontal face since poor registrations will occur due to out-of-plane rotation. Still, a better solution should be studied before we can use the natural skin texture in motion tracking.

3. Facial Motion Tracking using Active Face Model and Active Appearance Algorithm

In this chapter, we parameterize a deformable face model to find the optimal adaption to the human face in each frame of an image sequence. The content is mainly based on [10] and developed tools are shown.

3.1 Candide-3 face model

The Candide-3 face model [14] is a fairly simple model which has the MPEG-4 [16] standard for synthesizing facial animation. Facial animation and definition parameters (FAPs and FDPs) are defined in MPEG-4.

Candide-3 is controlled by global motions, Shape Units (SUs) and Action Units (AUs). The global motions correspond to translation, scaling and rotation around three axes. Shape Units are used to change person-specific facial shape so that the model can be fit to different individuals. The Action Units (also mentioned as Animation Units) control the mimics of the face to make various expressions.

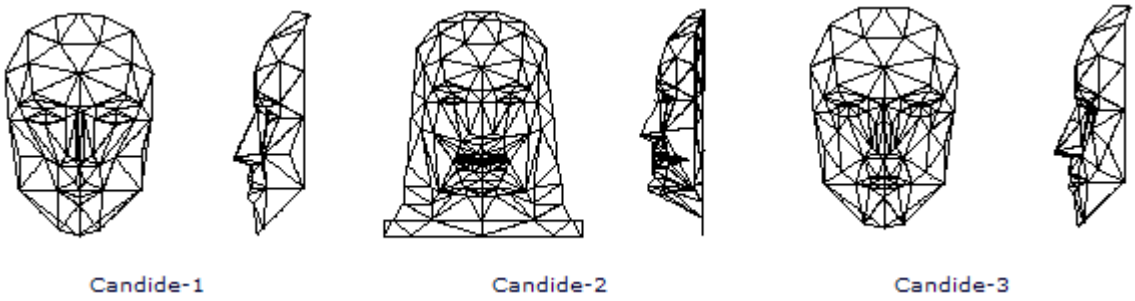


Figure 3.1: The different versions of Candide

Candide-3 holds 113 vertices and 168 surfaces which can be downloaded from [15] to get the latest version (v3.1.6) of the WFM (wireframe) file defining the Candide model.

The model is reshaped according to

$$g(\sigma, \alpha) = (z + 1)R(\bar{g} + S\sigma + A\alpha) + t$$

where the resulting vector g contains the new (x, y, z) vertex coordinates. The columns of S and A are the Shape and Animation Units respectively, and thus the vectors σ and α contain the shape and animation parameters. $R = R(r_x, r_y, r_z)$ is a rotation matrix, z is the scale and $t = t(t_x, t_y)$ is the 2D translation vector.

3.2 Face model adaptation

In order to get a large set of normalized training images (to be discussed in 3.4), we needed to develop a tool which could ease the step of adapting the face model. A Matlab-based tool named “adapt_candide3” was developed to control the Candide-3 so that face model adaption could be approached manually. In fact, it is a procedure that optimize g over the parameter vector

$$p = [r_x, r_y, r_z, z, t_x, t_y, \sigma^T, \alpha^T]^T .$$

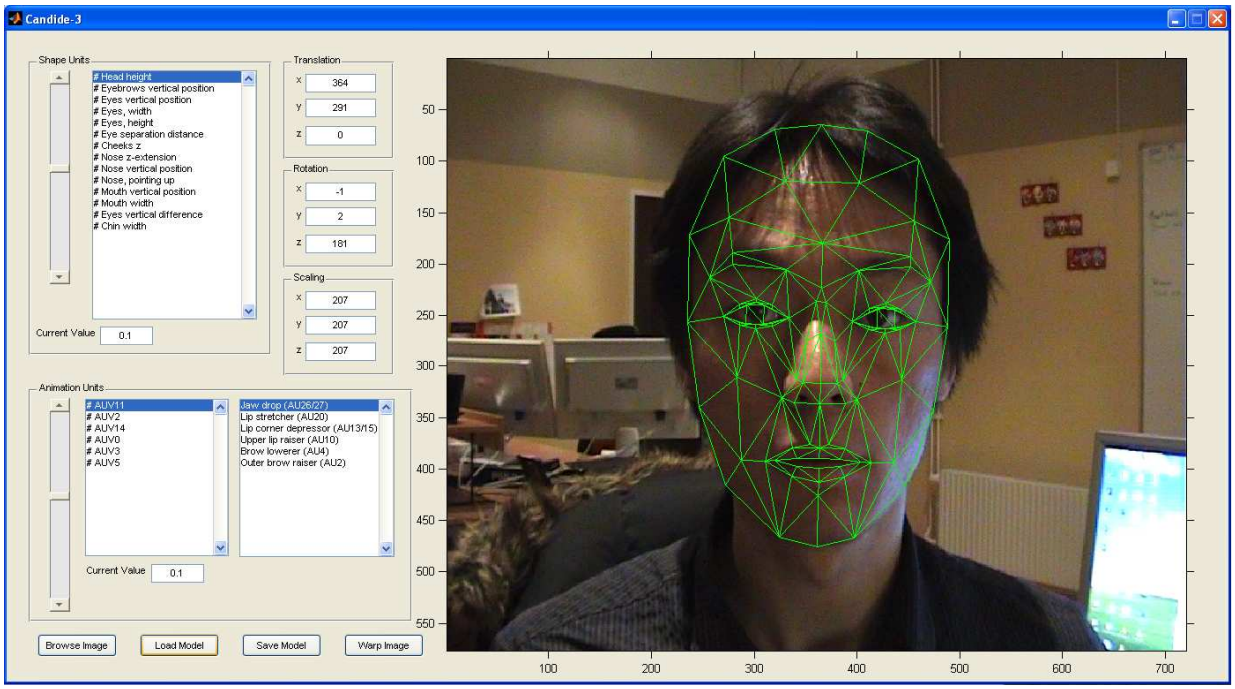


Figure 3.2: Screenshot of "adapt_candide3"

In Matlab, we get g as

$$g = Z \cdot R_x \cdot R_y \cdot R_z \cdot (\bar{g} + S\sigma + A\alpha) + T$$

where

$$Z = \begin{bmatrix} z+1 & 0 & 0 \\ 0 & z+1 & 0 \\ 0 & 0 & z+1 \end{bmatrix}, R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cosx & -sinx \\ 0 & sinx & cosx \end{bmatrix}, R_y = \begin{bmatrix} cosy & 0 & siny \\ 0 & 1 & 0 \\ -siny & 0 & cosy \end{bmatrix},$$

$$R_z = \begin{bmatrix} cosz & -sinz & 0 \\ sinz & cosz & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } T = \begin{bmatrix} t_x \\ t_y \\ 0 \end{bmatrix}.$$

Since Shape Unit parameters are static for the same person while the global motions and Animation Unit parameters are varying throughout the tracking procedure, the parameter vector p can be reduced to

$$p' = [r_x, r_y, r_z, z, t_x, t_y, \alpha^T]^T .$$

Different vertices correspond to unique MPEG-4 Facial Feature Point (FFP), but not all FFPs are implemented in Candide-3 such as tongue, ears and teeth. The WFM file format, the relationship between the Candide-3 vertices and MPEG-4 Facial Feature Points, Shape Units and Action Unit Vectors are described in detail in [14].

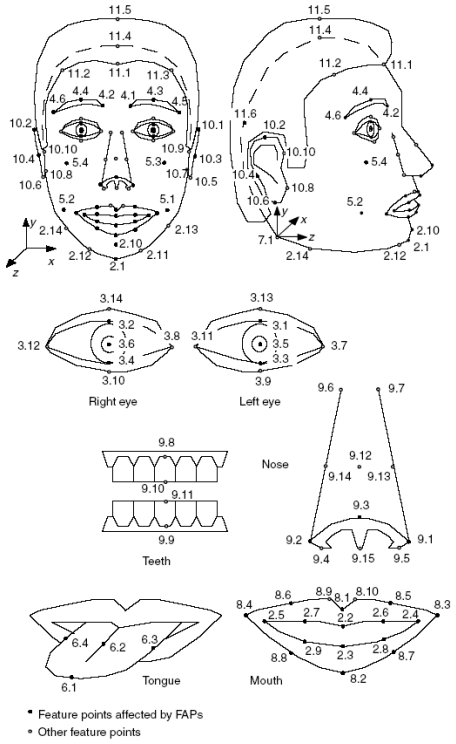


Figure 3.3: Facial Feature Points defined by MPEG-4

3.3 GSA model file

The parameter vector p of the adapted face model should be saved either when training data or tracking unseen images. The Candide-3 wireframe could be reconstructed by loading GSA (Global motion, Shape, Animation) file.

File content	Description
# GLOBAL MOTION:	Optional comment

<float t_x> <float t_y> <float t_z> <float r_x> <float r_y> <float r_z> <float z_x> <float z_y> <float z_z>	Translation parameters Rotation parameters Scaling parameters
# SHAPE PARAMETERS: <int NoSPs> for i=0 to NoSPs-1 <float SPval> end	Optional comment The number of Shape Unit parameters The value of Shape Unit parameter
# ANIMATION PARAMETERS: <int NoAPs> for i=0 to NoAPs-1 <float APval> end	Optional comment The number of Animation Unit parameters The value of Animation Unit parameter
# END OF FILE	Optional comment

Table 3.1: GSA model file format (.gsa)

3.4 Normalized face model

After well adapting the Candide-3 model to the face, we need to warp the image to the normalized face model.

3.4.1 Eigenface

Eigenfaces are a set of eigenvectors used in the computer vision problem of human face recognition [20]. To create a set of eigenfaces involving following steps:

- Collect a large number of face images as a training set.
- All the images should be scaled in same resolution and roughly aligned.
- The mean image vector needs to be calculated from the training set.
- Perform a principal component analysis (PCA) to compute the eigenvectors and eigenvalues of the covariance matrix.
- Choose the principal components, i.e., the eigenvectors with largest associated eigenvalue.

3.4.2 Non-linear face space

The problem with eigenfaces is that they are often be used to synthesize images which are also non-faces, and thus those images are classified as faces. In fact, the set of face images is not a convex space.

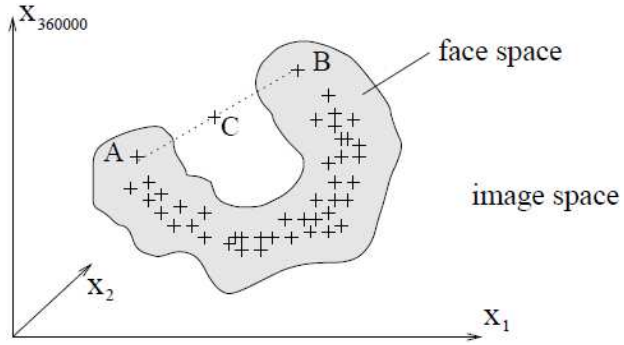


Figure 3.4: The non-linear face space

Figure 3.4 illustrates the face space for 600×600 pixels images. An object C locates outside the face space when it is represented linearly by face A and B.

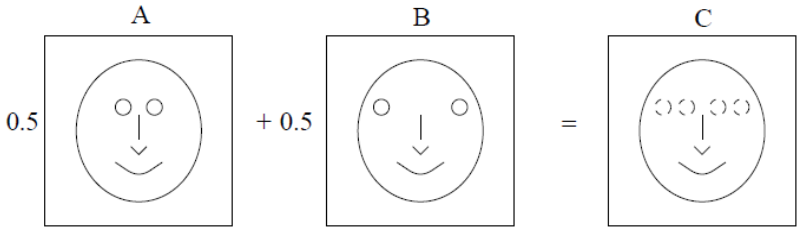


Figure 3.5: Linear representation of non-face image C

Face A and face B synthesize something that should not belong to the face space which has four eyes due to the non-linear face space, $C = 0.5A + 0.5B$.

3.4.3 Geometrical normalization

To model a more suitable face set, we transform the face space into a convex one by creating geometrically normalized [17] or shape-free [2] face images. Here, we adopt geometrical normalization in this case.

We choose to work with 64×77 pixels images which are conveniently small and effective for image warping. Since the image coordinate system in Matlab is different from face model in WFM file, translation, rotation and scaling should be performed to produce the normalized face model. Twelve surfaces are removed to avoid the effects caused by hair.

Translation	[32.11 32.926 0]
Rotation	[0 0 180]
Scaling	[50 50 50]
Shape Units	[0 … 0]
Animation Units	[0 … 0]
Removed surfaces	(0 34 1) (44 0 34) (0 11 1) (45 44 34) (1 11 12) (45 46 34) (1 12 13) (47 45 46) (12 13 14) (1 13 2) (1 2 34) (2 46 34)

Table 3.2: Parameters of normalized face model and removed surfaces

Scaling and rotation parameters should be set in advance. 180 degrees rotation around z-axis is performed at first due to the opposite direction of y-axis. Then find the minimum coordinates for x, y-axis of the model vertices with un-removed surfaces since the image origin in Matlab is at the top-left corner start from $(1,1)$ instead of central point. After observation, we know that the vertex 14 and 13 (shown in Figure 3.6) hold most left and top coordinates respectively.

3.5 Image warping

3.5.1 Barycentric coordinate computation

Given the vertices of a triangle in 2D Euclidean space

$$V = [v_1, v_2, v_3]$$

where

$$v_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix},$$

the barycentric coordinates of a point x with respect to V are values a, b, c so that

$$x = V \begin{bmatrix} a \\ b \\ c \end{bmatrix}.$$

If the vertices v_i lie on a line or in a single point, the barycentric coordinates are undefined.

The barycentric coordinates a, b, c are valid if and only if

$$\begin{cases} 0 \le a \le 1 \\ 0 \le b \le 1 \\ 0 \le c \le 1 \\ a + b + c = 1 \end{cases}.$$

A point x is inside a triangle V if and only if its barycentric coordinates with respect to V are valid.

The barycentric coordinates are computed as

$$x' = x - v_3$$

and

$$V = [v_1 - v_3, v_2 - v_3]$$

it follows that

$$x' = V \begin{bmatrix} a \\ b \end{bmatrix}.$$

Thus a, b, c can be computed as

$$\begin{bmatrix} a \\ b \end{bmatrix} = V^{-1} x'$$

and

$$c = 1 - a - b\,.$$

If the vertices of the triangle lie on a line or in a single point, V will be singular and the barycentric coordinates undefined.

3.5.2 Warping process

Suppose the normalized face model is the destination mesh M that contains triangles V_1, … V_N and we want to warp an image from one shape, the source mesh M', to the normalized one. Each triangle is a triplet of 2D vertex coordinates as

$$V_n = [v_1, v_2, v_3]$$

where

$$v_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}.$$

Source image and destination image are $f'(x)$ and $f(x)$ respectively.

- Compute the barycentric coordinates a, b, c for each pixel coordinate x in destination image. Find the correct triangle V_n in the destination mesh M so that the barycentric coordinates of x are valid.

- Obtain the source coordinate x' from the barycentric coordinates multiplied by corresponding triangle in the source mesh M',

$$x' = V_n' \begin{bmatrix} a \\ b \\ c \end{bmatrix}.$$

- Interpolate (nearest neighbor) the source image f' in x' and set $f(x) = f'(x')$.

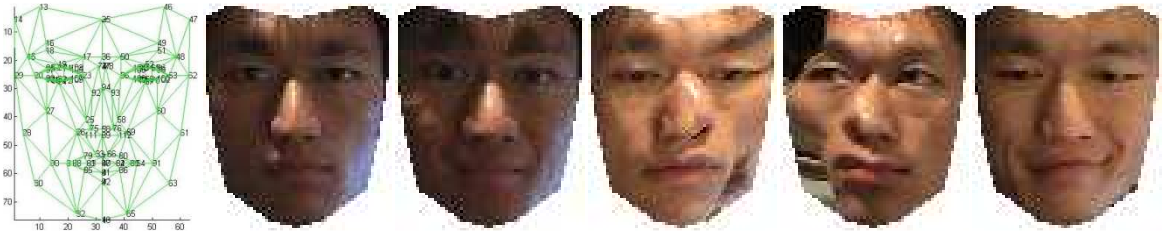


Figure 3.6: Normalized Candide-3 face model and warped images

Figure 3.6 illustrates after warping the images using the destination triangle mesh, how normalized face images look like with different face expressions, global motions and light conditions.

3.6 Texture synthesis

Perform PCA on the training set (stored as 64×77×3 dimensional texture vector) so that we obtain the principal modes of variation, i.e., the eigenfaces.

In this case, we collect 20 eigenfaces in a matrix X which could represent 90% of variance. A face vector x can be parameterized as

$$\xi = X^T(x - \bar{x})$$

where $\bar{x}$ is the mean face, or we could synthesize a face image as

$$x = \bar{x} + X\xi$$

from a face parameter vector.

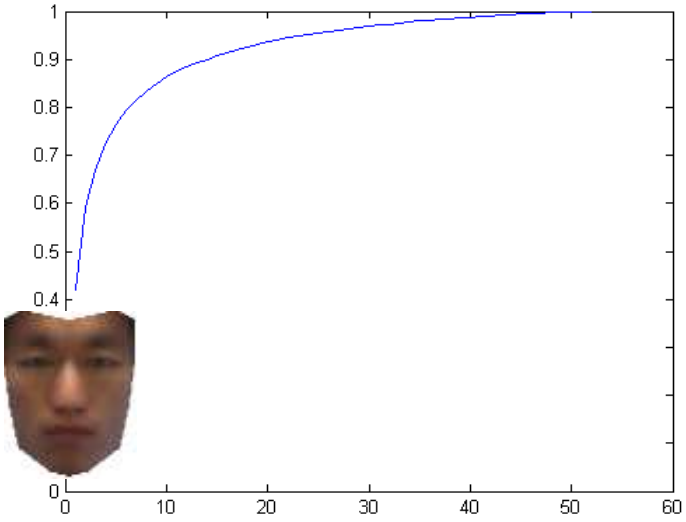


Figure 3.7: Percentage of variance

Since

$$20 \leq the\ number\ of\ training\ examples \leq the\ number\ of\ pixels\ in\ the\ image$$

we get the combination as

$$\hat{x} = \bar{x} + XX^T(x - \bar{x})$$

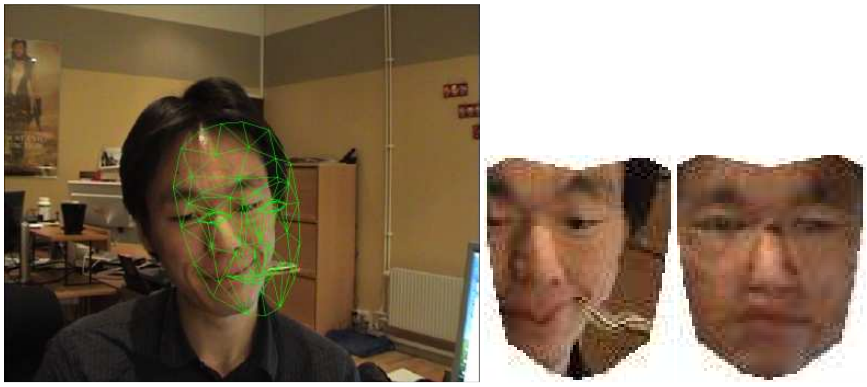


Figure 3.8: Texture synthesis

A bad model adaptation is shown in the first picture of Figure 3.8 that warped to the standard shape producing a normalized face which actually contains part of the noisy background. In the last picture, a synthesized texture $\hat{x}$ is computed according to the above equation.

3.7 Facial tracking

The procedure of facial tracking is to find the optimal adaptation of the model to a frame in an image sequence, i.e., to find the parameter vector p (or p') that minimizes the distance between normalized and synthesized faces.

The initial value of p we use is the optimal adaptation to the previous frame. Assuming that the motion from one frame to another is small enough, we reshape the model to $s(p)$ and map the image i (the new frame) onto the model. Then we geometrically normalize the shape and get the resulting image as a vector

$$j(i,p) = j\big(i, s(p)\big).$$

Since our model is parameterized separately in shape and texture, we compute the texture parameters from $j(i,p)$ according to

$$\xi(i,p) = X^T(j(i,p) - \bar{x})\ .$$

The texture could be reconstructed as

$$x(i,p) = \bar{x} + XX^T(j(i,p) - \bar{x})$$

and then we compute the residual image

$$r(i,p) = j(i,p) - x(i,p)\,.$$

The summed square error (SSE) is chosen for the error measure

$$e = \|r(i,p)\|^2\,.$$

Figure 3.9: Residual images for successful and failed (Figure 3.8) adaptation

A successful adaptation would synthesize a kind of "good-looking" face which is apparently different from the one in Figure 3.8. Also the residual image e is much smaller, i.e., an almost black image could be the best adaptation for the face model.

The update vector

$$\Delta p = Ur(i,p)\,,$$

thus the new error measure for updated parameters

$$e_0 = \|r(i,p+\Delta p)\|^2\,.$$

If $e_0 \leq e$ we update

$$e = e_0, p = p + \Delta p\,,$$

if not, e is re-computed as

$$e_k = \left\| r\left(i, p + \frac{1}{2^k}\Delta p\right)\right\|^2$$

for $k = 1,2,\ldots$, and if $e_k \leq e$ we update

$$e = e_k, p = p + \frac{1}{2^k}\Delta p\,.$$

The scheme will iterate until convergence when $e_k > e$ for $k = 1,2,3$. We set a limit of 20 iterations for optimizing the parameter vector p to force a convergence in case e_0 is always less than e.

3.8 Estimating the gradient matrix

In order to obtain the update matrix U, we need to know the gradient matrix G. According to 3.7, our optimal parameter vector after convergence should be

$$p^*(i) = \underset{p}{argmin}\, e(i,p)\ ,$$

using Taylor-expansion to r around $p + \Delta p$, we get

$$r(i, p + \Delta p) = r(i,p) + G\Delta p + O(\Delta p^2)$$

where

$$G = \frac{\partial}{\partial p} r(i,p)\ .$$

Thus, the Δp should be calculated with respect to a given p which minimizes

$$e(i, p + \Delta p) \approx \|r(i,p) + G\Delta p\|^2\ .$$

The solution for the least squares problem gives us

$$\Delta p = -(G^T G)^{-1} G^T r(i,p)\ ,$$

therefore, the update matrix U is the negative pseudo-inverse of the gradient matrix G

$$U = -G^{\dagger} = -(G^T G)^{-1} G^T\ .$$

Since G is similar for different images, it could be estimated from the training data in advance.

The j^{th} column in G,

$$G_j = \frac{\partial}{\partial p_j} r(i,p)\ ,$$

the approximation could be as

$$G_j \approx \frac{r(i, p + hq_j) - r(i, p - hq_j)}{2h}\ ,$$

where h is the step size for perturbation and q_j is a vector with one in j^{th} column and zero in the rest elements.

The Candide-3 face model was adapted to every training image in the training set to compute the shape and texture modes. So, a set of corresponding parameter vectors p_n is obtained for a suitable step size to estimate G_j by averaging as

$$G_j \approx \frac{1}{NK} \sum_{n=1}^{1} \sum_{k=1}^{1} \frac{r(i_n, p_n + khq_j) - r(i_n, p_n - khq_j)}{2h}$$

where N is the number of training images and K is the number of steps to perturb the parameters.

Here, we choose the perturbation step $K = 20$ and h from empirical tests, $N = 24|29$ training images are used, and top 20 texture modes is computed to synthesized image $x(i, p)$. The following six Action Units as defined in Candide-3 have been selected as animation modes (corresponding Action Unit Vector):

- Jaw drop (# AUV11)
- Lip stretcher (# AUV2)
- Lip corner depressor (# AUV14)
- Upper lip raiser (# AUV0)
- Eyebrow lower (# AUV3)
- Outer brow raiser (# AUV5)

Thus, $20 \times 12 \times 24 = 5760$ and $20 \times 12 \times 29 = 6960$ different images have been computed to estimate the gradient matrix G with $K = 20$, $N = 24|29$ and the size of parameter vector $p' = 12$. The update matrix U can be computed from G. The step size h needs to be treated slightly different. For rotation, h has been converted to radians $h = 0.01 \times 180/pi$, in the range $[-hK, hK] = [-11.459,11.459]$; for scaling, $h = 100 \times 0.01$, in the range $[-hK, hK] = [-20,20]$; for translation, h is relative to the size of model, $h = 0.01 \times (z + 1)$, in the range $[-hK, hK] = [-0.2 \times (z + 1), 0.2 \times (z + 1)]$; for action units, $h = 0.01$, in the range $[-hK, hK] = [-0.2,0.2]$.

3.9 Tracking experiments

In order to test the facial tracking, two video sequences were recorded using the same video camera Sony DCR-H23E with 25 Fps.

For the first frame in video sequence, we used “adapt_candide3” to fit the model to the face manually so that we could get an initial parameter vector p which is close to the optimal one. The parameter vector p' containing the global motion and animation parameter vector α was then iteratively optimized for each frame.

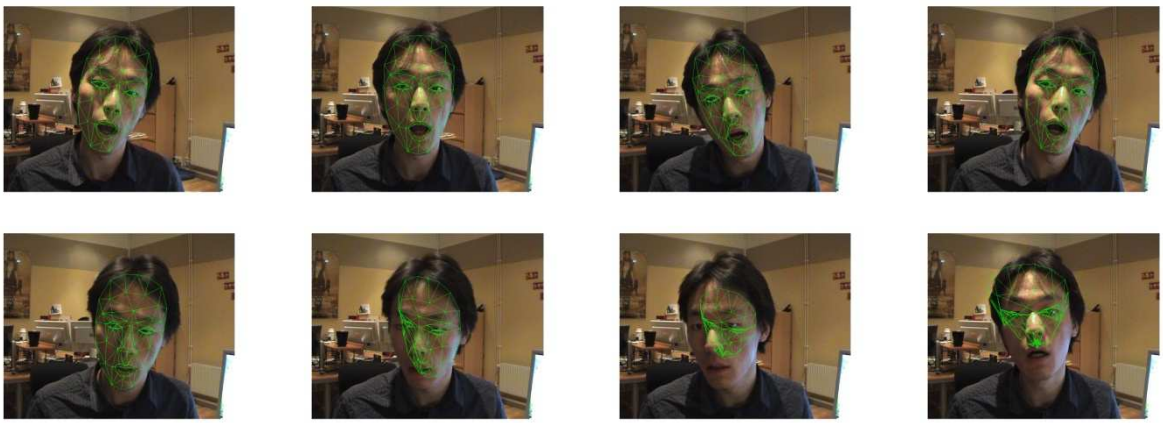


Figure 3.10: Test video sequence 1 with 24 training images

Random head movements and facial expressions were shown in Figure 3.10 that the wireframe model didn't fit well in last several frames apparently.

Figure 3.11: Last several frames in test video sequence with 29 training images

A slight improvement was obtained when the training images were increased from 24 to 29 which means both eigenfaces and gradient matrix G got a little bit more robust.

Figure 3.12: Test video sequence 2 with 29 training images

A much faster head movements and facial expressions were tracked with 29 training images. The global adaptation has a good behavior most of the time. However, the mouth of the model lost tracking in the beginning, recovered after several seconds and lost again at last.

3.10 Discussion

The reason for lost tracking is possibly lack of robust training data with identical background and light condition. Less accurate adaptation of training images could be another reason, since it's hard to decide parameters for scaling and some rotation situations manually. Still the training session is very time consuming. Hair will also affect the tracking results due to different angels of frontal face when synthesizing a new image.

3.11 Possible improvements

- Since training session is the most important and time consuming part throughout the method, a more accurate and faster model adaptation solution should be considered.

- More training images should be collected with identical background and light condition. For now, the facial tracking is very limited to a particular person because the same face in all training images does not generate good eigenfaces for an unseen face. Although person-specific texture models could improve the accuracy.

- The rotation of Candide-3 face model needs to be re-written because the original one is based on the world coordinate system and what we want to do is actually rotate the model around object coordinates that would facilitate the manual adaptation.

- The normalized face model should be refined slightly to minimize the effects caused by person's hair.

- Different step size h could be tried out to get a good estimation for the gradient matrix G, especially for scaling.

- Performing DC level (the average value) removal and setting the norm to one on the training texture vectors would make the eigenfaces tolerant to various backgrounds and light conditions. Note that this also requires the image $j(i, p)$ to be processed in the same way.

- Find a mechanism to recover the failed tracked frames during the tracking. This is a demanding task since one must be able to robustly detect whether the frame lost tracking and to do the recovery within an acceptable time.

- In our experiments, the same texture model was used for all faces regardless of angle of the face. In-plane rotation (rotation around z-axis) is compensated for when normalizing the texture shape, but the out-of-plane rotations (rotation around x, y-axis) are not treated. Particularly y-rotation might be problematic while the x-rotation is rather small. A view-based texture model [18] could be used to increase the accuracy of tracking by replacing training sets according to face angle range.

- Image warping is the bottleneck for the performance of tracking, in fact, this is a rendering process. An implementation based on C++ and OpenGL might increase the speed greatly.

- More Action Units could be added for training and tracking, but the computing time may grow linearly.

4. View-based Texture Modeling

In this chapter we will mainly focus on how to improve the methodology described in chapter 3 based on [18]. View-based texture modeling provides more robust eigenfaces for specific head rotation ranges that wouldn't be affected by large out-of-plane rotation angels.

4.1 Capturing Setup

Since the experiments discussed in previous chapter showed that noisy background and varying illumination will degenerate the tracking result, we set up the camera in front of a dark background as follow this time.

Figure 4.1: Captured frames from original and refined environments respectively

The new captured frame looks rather boring but with less varying background and lighting. We benefit from more accurate gradient matrix and residual images due to the better capturing situation to calculate Δp.

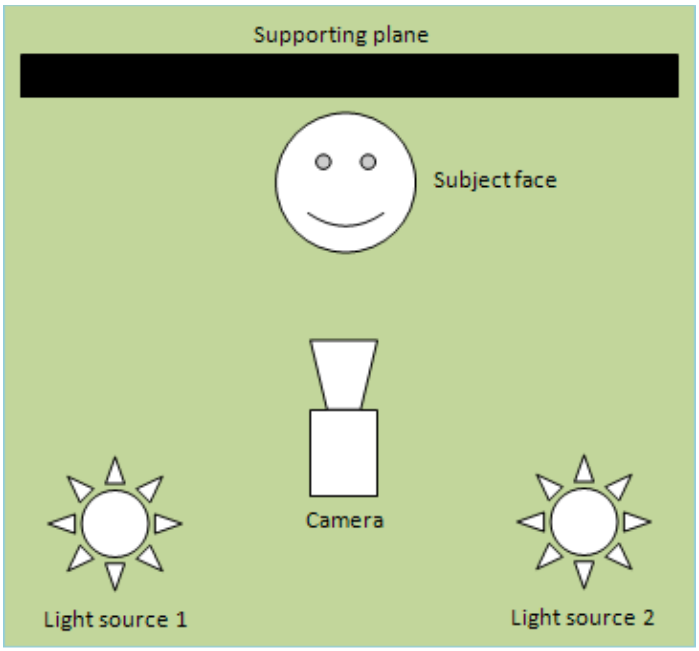


Figure 4.2: Graphic of capturing situation

4.2 Modifications

4.2.1 Formula of model

The formula of Candide-3 face model is refined to

$$g(\sigma, \alpha) = M \cdot Z \cdot R_x \cdot R_y \cdot R_z \cdot (\bar{g} + S\sigma + A\alpha) + T$$

where

$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $Z = \begin{bmatrix} z+1 & 0 & 0 \\ 0 & z+1 & 0 \\ 0 & 0 & z+1 \end{bmatrix}$, $R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cosx & -sinx \\ 0 & sinx & cosx \end{bmatrix}$,

$R_y = \begin{bmatrix} cosy & 0 & siny \\ 0 & 1 & 0 \\ -siny & 0 & cosy \end{bmatrix}$, $R_z = \begin{bmatrix} cosz & -sinz & 0 \\ sinz & cosz & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and $T = \begin{bmatrix} t_x \\ t_y \\ 0 \end{bmatrix}$. The columns of S and A are the Shape and Animation Units respectively, and the vectors σ and α contain the shape and animation parameters.

The global motion of the previous model is rotated about the world coordinates which is fairly difficult for the user to fit it to the training images. The new equation makes it easier to control the model since it's kind of semi world coordinates and semi object coordinates rotation. As long as the equation is identical throughout the training and tracking procedure, the result won't be affected.

4.2.2 Cropped frontal normalized face

As we mentioned before, the hair would influence the tracking results. Here, we crop the normalized face by 7 pixels from the top of the image to obtain the face-only image. This kind of treatment is also feasible for view-based normalized faces.

4.2.3 Head pose estimation

For adapting model with more accuracy, a method of 3D head pose estimation [19] is applied to estimate Z and Y-axis rotations from monocular image. First we need to landmark some facial points manually, a bunch of feature points are selected:

- ell, elr – left/right corner of left eye
- erl, err – left/right corner of right eye
- em – midpoint of eyes
- lel, ler – left/right edge point of face at eyes
- nm – midpoint of nose
- lnl, lnr – left/right edge point of face at nose
- mm – midpoint of mouth
- lml, lmr – left/right edge point of face at mouth

Roll (rotation angle of Z-axis, γ):

$$\gamma_1 = \arctan\left(\frac{y_{ell} - y_{elr}}{x_{ell} - x_{elr}}\right)$$

$$\gamma_2 = \arctan\left(\frac{y_{erl} - y_{err}}{x_{erl} - x_{err}}\right)$$

$$\gamma_3 = \arctan\left(\frac{\frac{y_{ell} + y_{elr}}{2} - \frac{y_{erl} + y_{err}}{2}}{\frac{x_{ell} + x_{elr}}{2} - \frac{x_{erl} + x_{err}}{2}}\right)$$

$$\gamma = (\gamma_1 + \gamma_2 + \gamma_3)/3$$

Yaw (rotation angle of Y-axis, β):

$$lx = \sqrt{(x_{lxr} - x_{lxl})^2 + (y_{lxr} - y_{lxl})^2}$$

where lx could be le, ln and lm which correspond to the face width at eyes, nose and mouth points respectively.

$$xm2l = \sqrt{(x_{xm} - x_{lxl})^2 + (y_{xm} - y_{lxl})^2}$$

$$xm2r = \sqrt{(x_{xm} - x_{lxr})^2 + (y_{xm} - y_{lxr})^2}$$

where $xm2l$ and $xm2r$ could be $em2l$, $em2r$, $nm2l$, $nm2r$, $mm2l$ and $mm2r$ respectively, which is the distance from the symmetric points at eyes, nose and mouth to the left edge and right edge of face respectively.

$$\delta x = [\frac{lx}{2} - \min(xm2l, xm2r)]$$

$$\beta_x = \arcsin(\frac{\delta x}{lx/2})$$

$$\beta = (\beta_e + \beta_n + \beta_m)/3$$

However, this method may introduce big errors due to the manual landmarks and we just use it as rough pose estimation for the training process.

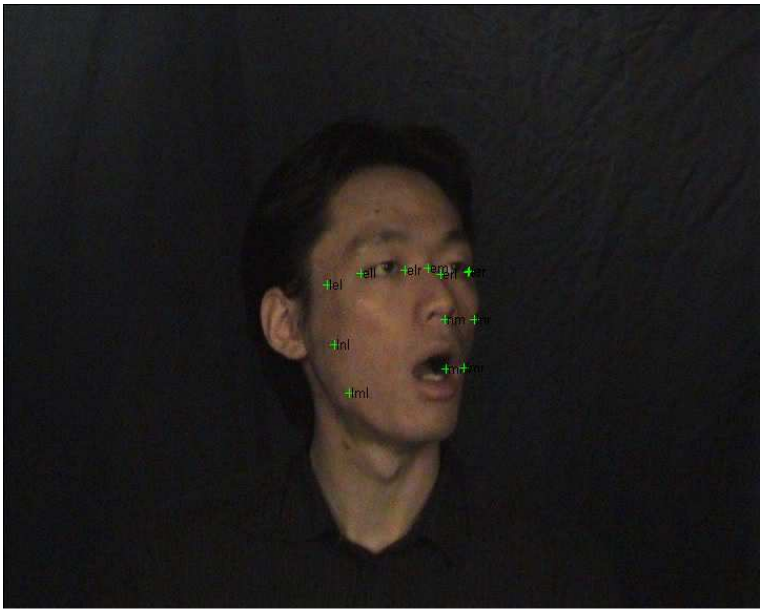


Figure 4.3: Landmarks for 3D head pose estimation

4.2.4 Energy function

Based on 4.2.3, we further align the face model to a training image by minimizing the following energy function,

$$\sum_{j=1}^{n} \|c_i - c_i^r\|^2$$

where c_i means the coordinates of i-th feature point in Candide-3 face model, c_i^r denotes the coordinates of reference feature points, in this case, $n = 10$ and $i = 20, 23, 56, 53, 26, 59, 5, 39, 31, 64$.

Red crosses indicate reference feature points; the value of the energy function is shown below the training image area. After adapting model to 40 images, the values of energy function range in $(50, 250)$ which is quite accurate for manual alignment.

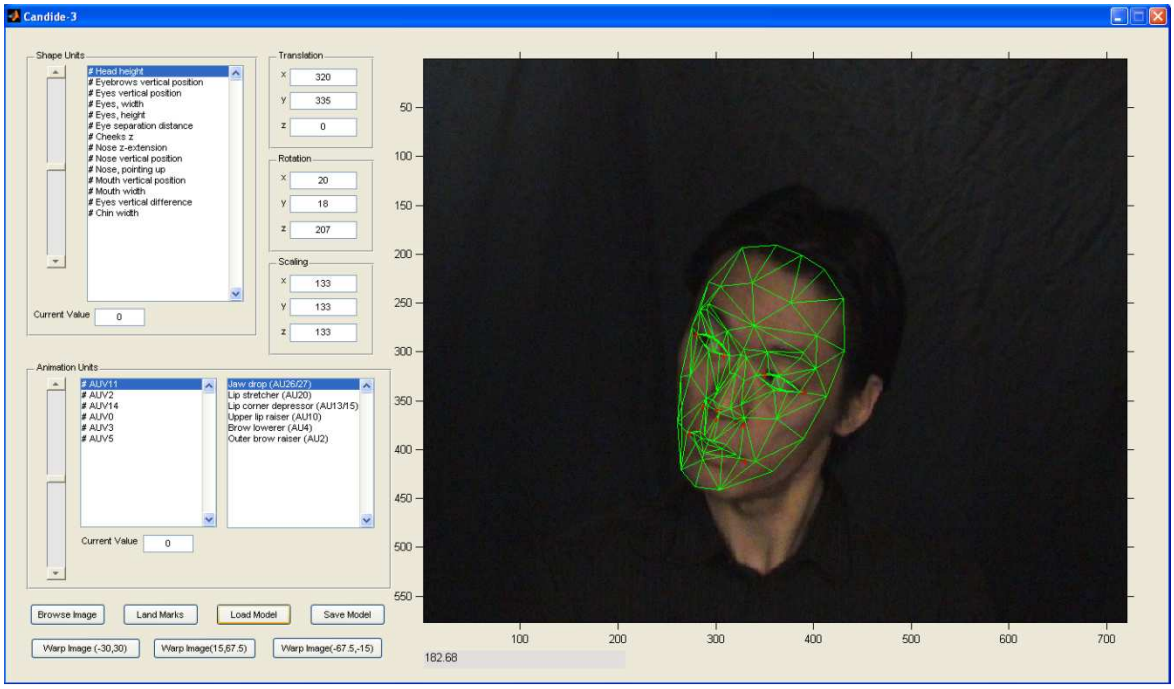


Figure 4.4: Adapted face model with reference facial feature points

4.3 View-based texture modeling

In previous tracking, the same texture model was used for all faces regardless of the angel of the face for the out-of-place rotation which is around the X and Y-axis. Especially the Y-axis rotation would be a problem when the angle is from 45 degrees to 90 degrees.

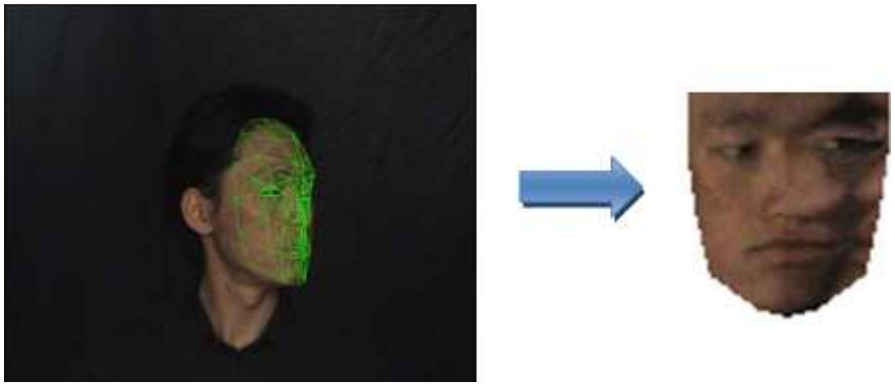


Figure 4.5: A half profile model adaptation and its frontal normalized texture

Figure 4.5 illustrates that while warping a half profile (approximately -50 degrees rotation around Y-axis) to the frontal normalized face, the right hand side of the normalized texture seems a little bit mirrored and distorted.

If we use a set of texture models each handling a different range of angles of Y-axis, it possibly would improve the tracking result. For example, Y-axis rotation from minus 22.5 to 22.5 degrees, the frontal texture model and corresponding update matrix U are used; when turning to 22.5 to 67.5 degrees, the texture model should be replaced by one of 45 degrees; From 67.5 to 125 degrees, a 90-degree texture model would be applied for warping image.

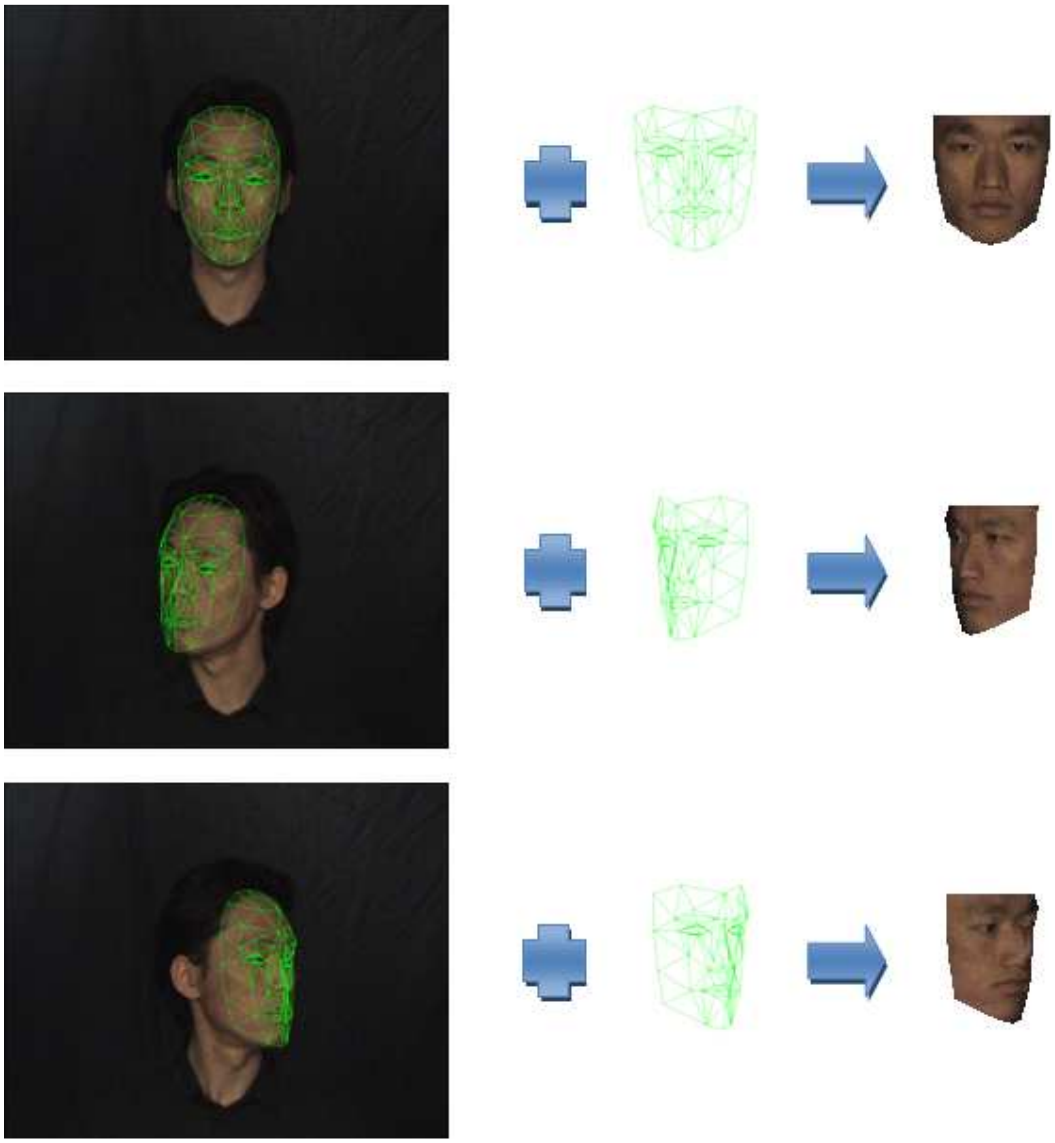


Figure 4.6: View-based texture models, [0 45 -45] degrees

Different model adaptation around Y-axis rotation was warped to corresponding normalized face model, i.e., 0, 45, -45-degree models.

This leads to a question, how can the computer know when to switch to proper texture model during tracking? Actually, it doesn't know. We do the tracking experiment with 3 mutually exclusive ranges of texture models which are $[-22.5^{\circ}\ 22.5^{\circ}]$, $(22.5^{\circ}\ 67.5^{\circ}]$ and $[-67.5^{\circ}\ 22.5^{\circ})$. However the computer lost tracking when the texture model should switch from one to another.
To conquer the problem, we overlapped some part of two different texture models that means they intersect each other in some portion of the angle range.

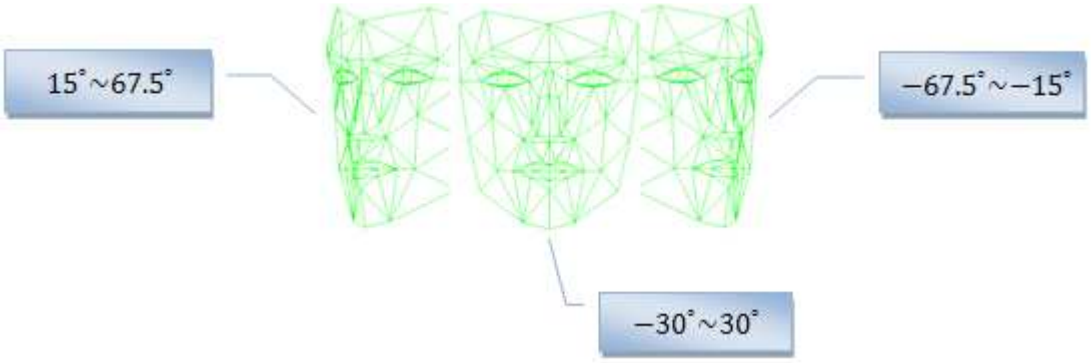


Figure 4.7: Overlaps between different texture models

Every two adjacent texture models have 15-degree intersection which could indicate when to switch the model. In fact, the actual warping ranges are $[-25^{\circ}\ 25^{\circ}]$, $[-67.5^{\circ}\ 25^{\circ})$ and $(25^{\circ}\ 67.5^{\circ}]$ respectively while doing the tracking. But in training session, when a face locates in both ranges illustrated in Figure 4.7, we should warp it twice for different texture models.

4.4 Tracking experiments

Again, we captured video using Sony DCR-H23E video camera with 25 fps. We got 28 images for frontal normalized face, 20 texture modes for over 90% reconstruction. 14 images for right half profile, 10 texture modes were selected. 10 images for left half profile, 7 texture modes were chosen.

Figure 4.8: Test video sequence

Since the sizes of normalized images for different models are not the same, our new error function changes to

$$e = c_x \cdot \|r(i,p)\|^2$$

where c_x is the coefficient for view-based residual images, $x = 1, 2, 3$.

4.5 Discussions

As shown in Figure 4.8, we get a much better tracking result than the one in 3.9, but still it has some defects such as scaling varying larger than we expect, inaccurate tracking of eyebrow motions and so on. We will try to fix those problems in Chapter 5.

5. Component sensitive, post-processing, stereovision and real-time tracking

Active Appearance Algorithm and view-based modeling give a good tracking result, however there still exist some problematic frames such as lost tracking for eye brows, inaccurate scaling parameters and so on. We will present the solutions to fix the problems in this report and further discuss about the stereo vision and real-time tracking.

5.1 Component-based normalized face model

Normally, the most important components of the human face are mouth, nose, eyes and eye brows which dominate the facial expressions. But previous methods we provided didn't give a good response to those partial face features, especially for eye brows and eyes since during the perturbation phase, the variations in those parameters didn't make a big difference in the warped images. A component sensitive normalized face model might solve the problem; several surfaces are removed from the original Candide-3 face model to segment the components. Making a new component-based face model is a better choice to track the expression accurately.

Left eye and eye brow	(15,19,18) (15,19,95) (18,17,19) (19,17,103) (15,95,20) (17,23,103)
Right eye and eye brow	(50,56,104) (50,104,52) (50,51,52) (51,52,48) (52,48,96) (48,96,53)
Nose and upper lip	(31,26,79) (26,79,33) (26,111,33) (111,33,6) (33,6,7) (6,7,66) (112,6,66) (66,112,59) (66,80,59) (80,59,64)
Lower lip and chin	(9 85 8) (9 86 8) (31 85 32) (32 85 9) (32 9 10) (9 10 65) (65 9 86) (65 86 64)
Forehead and eye brow	(29 15 14) (15 14 13) (15 16 13) (16 13 2) (16 17 2) (17 3 2) (3 2 50) (2 50 49) (2 46 49) (49 48 46) (46 47 48) (48 47 62)

Table 5.1: Removed surfaces between components

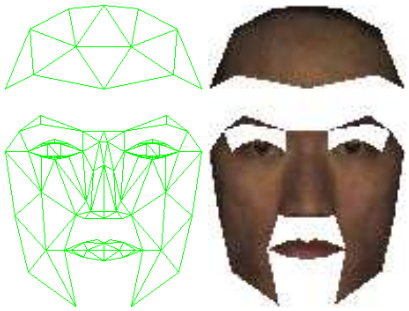


Figure 5.1: Component sensitive normalized face model and warped image

From Figure 5.1, both mouth and eye brows are segmented from the whole face model. Same test video sequence in 4.4 was used to evaluate the method. More sensitive and accurate tracking results are obtained as judged subjectively.

5.2 Post Processing

Due to the inaccurate manual alignment, inappropriate step size or lack of training images, after first tracking, some parameters are not good enough to generate a synthetic video sequence. If plotting the parameters throughout all frames, we will see some values which apparently need to be adjusted since they result in erratic movements that seem to be unnatural for a human face. In this case, we consider these values as outliers within a discrete curve. Therefore, a moving average (a lowpass filter with coefficients equal to the reciprocal of the span) can smooth the curve with respect to the choice of span.

Following figures show moving average applied in global motions of component sensitive facial tracking throughout a test video sequence with 220 frames, the sizes of the spans were chosen from several runs.

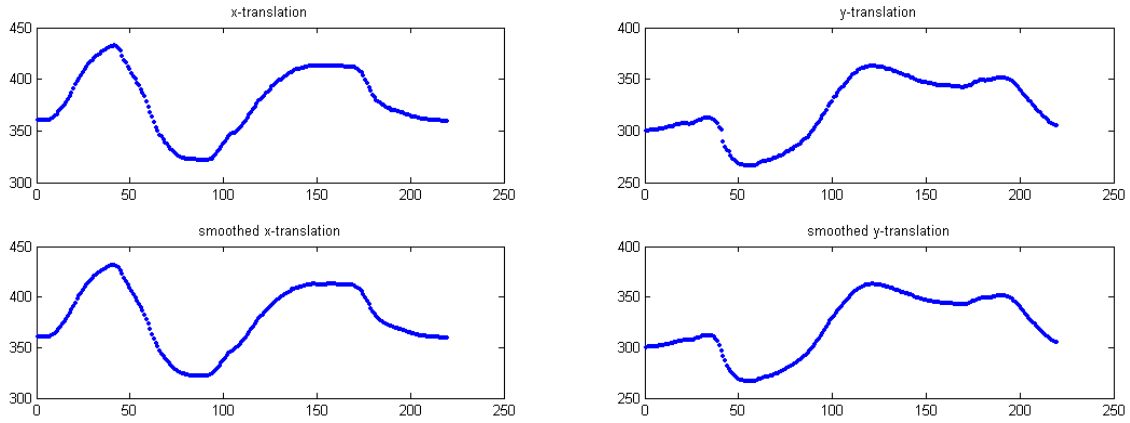


Figure 5.2: Translation parameters before and after post-processing

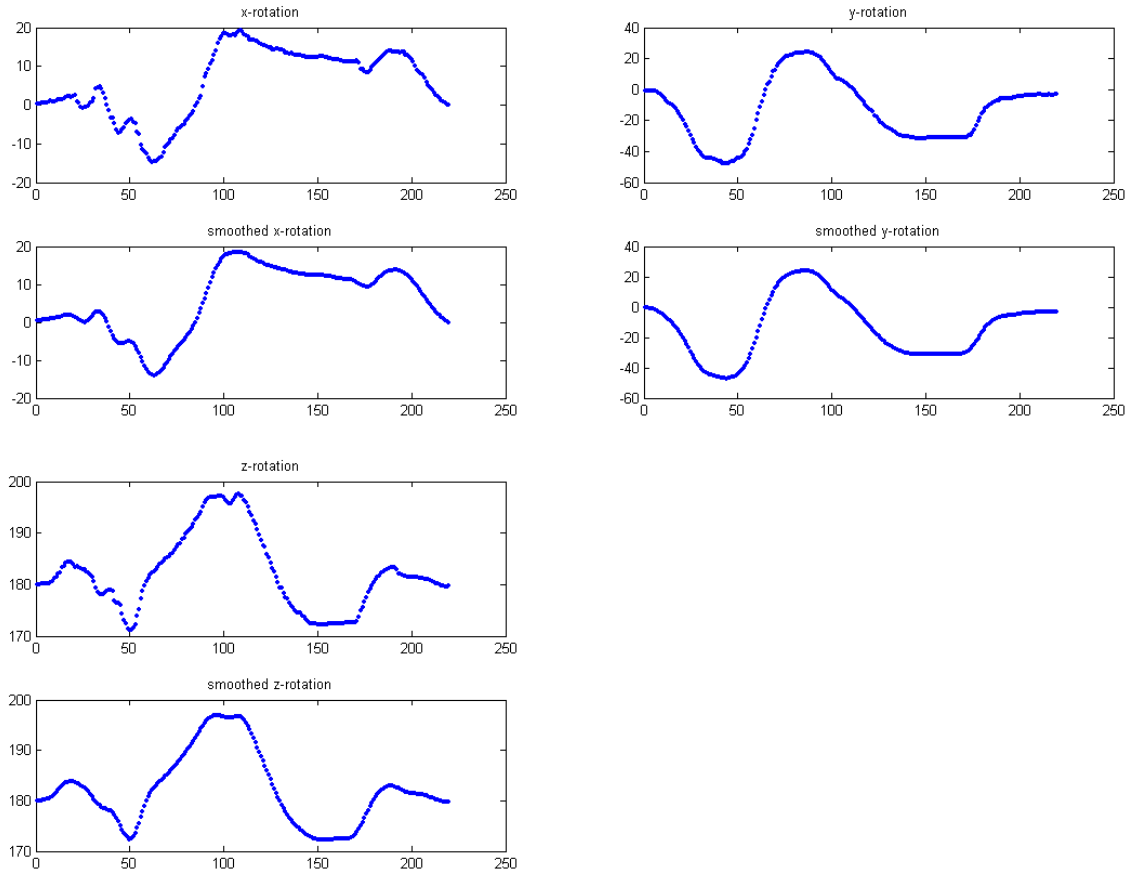


Figure 5.3: Rotation parameters before and after post-processing

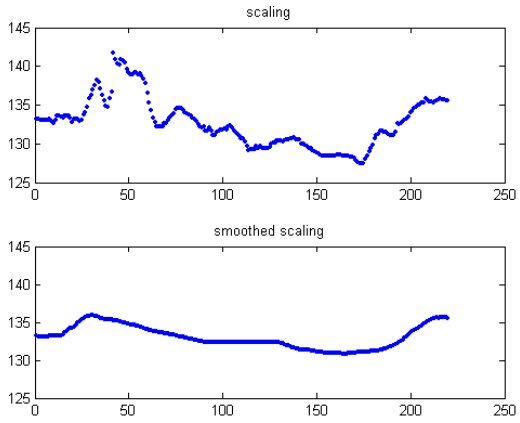


Figure 5.4: Scaling parameters before and after post-processing

The curves of translation and rotation perform well since they rarely have unstable values or outliers. However, scaling got a lot of noisy parameters and made the tracking volatile both in synthetic video sequence and plotting graphic as shown in Figure 5.4. Thus, a choice of bigger size of span smoothes the tracking and produces the result that we expect. Of course, frame-to-frame manual post processing is another effective but very time-consuming option and re-adapting those "bad" frames to update the gradient matrix can also improve the tracking results.

5.3 Tracking experiments

We used the same test video sequence as in Chapter 4 by applying view-based texture modeling, component-based normalized face model and post-processing in order to get satisfactory tracking result.

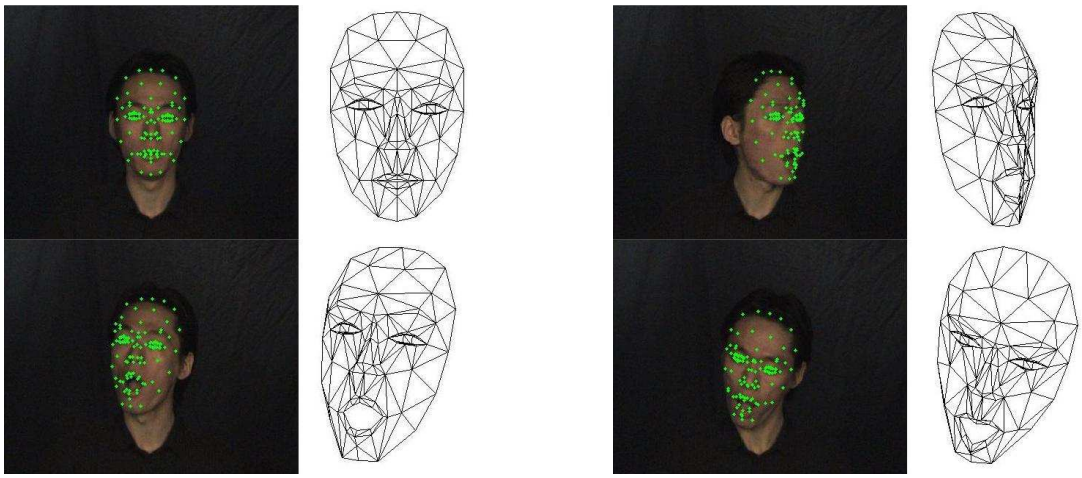


Figure 5.5: Test video sequence with full features

5.4 Capturing in stereo vision

Setting three or more cameras to capture the facial motion simultaneously gives us large coverage of face and more accurate and robust results which need to be implemented by view-based texture modeling. Three Candide-3 face models of 0, 90, -90 degrees around Y-axis are considered to be the normalized face models to which the textures will be warped.

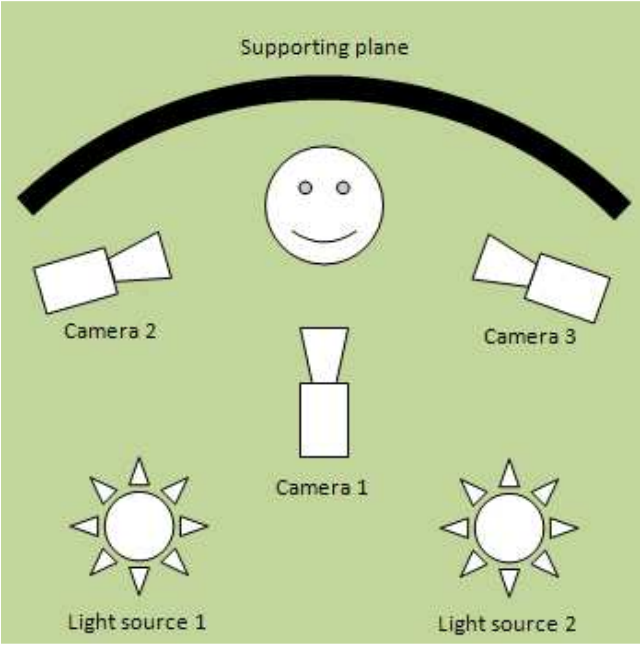


Figure 5.6: Graphic of the capturing situation using stereovision

Facial capturing with single camera suffers from inaccurate tracking when there's big out-of-plane rotation, undetectable facial movements from frontal view and so on. If we have several models fitting on same subject face, weighted animation parameters can be estimated from at least two cameras. For example, when camera 1 is shooting on actor's profile, the frontal face will be captured by camera 2 or 3. In that case, we won't lose tracking about the other side of the face. After pair-wise calibrating the cameras, proper rotation matrix R and translation vector T would be found according to [11]. Each vertex in one camera can be projected to another as $X' = R \cdot X + T$. Therefore, if the face is not within the tracking volume (e.g., 150 degrees of Y-axis), the camera will simply stop tracking and wait until the rotation angle is acceptable again. By projecting the vertices from the adjacent camera, the initial estimation of global motions are

computed using methods in [19], the animation parameter will keep intact most of the time.

5.5 Possibility of real-time tracking

So far, it takes approximately one second on each frame during the tracking. The time consuming parts can be identified as image warping, texture synthesis and update vector computation in one iteration. The image warping is the main bottleneck in the algorithm since it takes a lot of time to warp an image from one shape to another. Also reducing the number of iterations can improve the performance.

According to J. Ahlberg, using barycentric coordinate pre-computation to compute and interpolate the source coordinates implemented by C++ only costs 0.9ms in one iteration (depending on the size of the normalized face image). Another way to accelerate the algorithm is rendering the model on a graphic card using OpenGL. These two methods should be compared.

The tested time for the remaining computing time is around 0.05s on Matlab. So we reasonably assume that 10ms per iteration is the worst case running on C++, the average number of iterations is 10, leading to an overall 0.1s for one frame. So it should be possible to achieve real-time tracking.

5.6 Conclusions

By using component-based normalized face model and post-processing, the Animation Units perform more sensitively and unstable global motions are smoothed as shown in Figure 5.5. Future work should deal with implementing and testing stereo vision tracking for more coverage of the face and real-time tracking for immediate effects acquisition.

References

[1] Cootes, T. F., Edwards, G. J., and Taylor, C. J. 1998. Active Appearance Models. In *Proceedings of the 5th European Conference on Computer Vision-Volume II - Volume II* (June 02 - 06, 1998). H. Burkhardt and B. Neumann, Eds. Lecture Notes In Computer Science, vol. 1407. Springer-Verlag, London, 484-498.

[2] Cootes, T. F., Edwards, G. J., and Taylor, C. J. 2001. Active Appearance Models. *IEEE Trans. Pattern Anal. Mach. Intell.* 23, 6 (Jun. 2001), 681-685.

[3] Cootes, T. F., Taylor, C. J., Cooper, D. H., and Graham, J. 1995. Active shape models—their training and application. *Comput. Vis. Image Underst.* 61, 1 (Jan. 1995), 38-59.

[4] Turk, M. and Pentland, A. 1991. Eigenfaces for recognition. *J. Cognitive Neuroscience* 3, 1 (Jan. 1991), 71-86.

[5] Cootes, T. F., Cooper, D. H., Taylor, C. J., and Graham, J. 1992. Trainable method of parametric shape description. *Image Vision Comput.* 10, 5 (Jun. 1992), 289-294.

[6] La Cascia, M., Sclaroff, S., and Athitsos, V. 1999 *Fast, Reliable Head Tracking Under Varying Illumination: an Approach Based on Registration of Texture-Mapped 3D Models*. Technical Report. UMI Order Number: 1999-005., Boston University.

[7] Basu, S., Essa, I., and Pentland, A. 1996. Motion Regularization for Model-Based Head Tracking. In *Proceedings of the international Conference on Pattern Recognition (ICPR '96) Volume Iii-Volume 7276 - Volume 7276* (August 25 - 29, 1996). ICPR. IEEE Computer Society, Washington, DC, 611.

[8] Matthews, I. and Baker, S. 2004. Active Appearance Models Revisited. *Int. J. Comput. Vision* 60, 2 (Nov. 2004), 135-164.

[9] Holmberg B. 2008. Estimating Human Limb Motion Using Skin Texture and Particle Filtering. Uppsala: Universitetsbiblioteket, 2008. Digital

Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology, 568.

[10] Ahlberg, J. 2001. Using the Active Appearance Algorithm for Face and Facial Feature Tracking. In *Proceedings of the IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems (Ratfg-Rts'01)* (July 13 - August 13, 2001). RATFG-RTS. IEEE Computer Society, Washington, DC, 68.

[11] Bouguet, J.-Y. 2008. Camera calibration toolbox for Matlab. www.vision.caltech.edu/bouguetj/calib_doc/.

[12] Matlab. www.themathworks.com.

[13] Cootes, T. F., Hill, A., Taylor, C. J., and Haslam, J. 1993. The Use of Active Shape Models for Locating Structures in Medical Images. In *Proceedings of the 13th International Conference on Information Processing in Medical Imaging* (June 14 - 18, 1993). H. H. Barrett and A. F. Gmitro, Eds. Lecture Notes In Computer Science, vol. 687. Springer-Verlag, London, 33-47.

[14] Ahlberg, J. 2001. *CANDIDE-3 -- an updated parameterized face*, Report No. LiTH-ISY-R-2326, Dept. of Electrical Engineering, Linköping University, Sweden.

[15] CANDIDE – a parameterized face. www.icg.isy.liu.se/candide/main.html.

[16] Moving Picture Expert Group, ISO/IEC 14496, International Standard on Coding of Audio-Visual Objects (MPEG-4), 1999.

[17] Ström, J., Davoine, F., Ahlberg, J., Li, H., and Forchheimer, R. 1997. Very low bit rate facial texture coding. In *Proc. Int. Workshop on Synthetic/Natural Hybrid Coding and 3D Imaging*, pp. 237-240, Rhodes, Greece.

[18] Cootes, T. F., Walker, K., and Taylor, C. J. 2000. View-Based Active Appearance Models. In *Proceedings of the Fourth IEEE international*

Conference on Automatic Face and Gesture Recognition 2000 (March 26 - 30, 2000). FG. IEEE Computer Society, Washington, DC, 227.

[19] Wang, L. and Jin, Y. 2005. 3-D Head Pose Estimation for Monocular Image. FSKD 2005, LNAI 3614. Springer-Verlag, Berlin Heidelberg, pp. 293-301.

[20] Eigenface. http://en.wikipedia.org/wiki/Eigenface.

Table of Figures

Appendix

Sample videos in YouTube:
http://www.youtube.com/watch?v=lQyQtbj6SlA
http://www.youtube.com/watch?v=mtEAm7zb_Rw
http://www.youtube.com/watch?v=1Gd3g4tnbV8
http://www.youtube.com/watch?v=afDr111raEE
http://www.youtube.com/watch?v=1GK3vOUoGDM