Generating 3D Faces by Tracking and Pose Estimation in Video Streams

by Qichen Wang

B.S. in Communication Engineering, July 2011, Fudan University

a Thesis Submitted to

the Faculty of

the School of Engineering and Applied Science

of The George Washington University

in Partial Fulfillment of the Requirements

for the Degree of Master of Science

January 19, 2018

Thesis Directed by

James Hahn

Professor of Computer Science

Abstract of Thesis

This thesis presents a system that can generate animated 3D faces in real time, based on the ideas of Active Shape Model (ASM) and POS with Iterations algorithm (POSIT). The 3D face is controlled by human frontal face tracked in a video stream captured by commodity RGB camera, such as web cams and cameras on mobile phones. The pose of the human face is also estimated by analyzing the input video to create realistic shading on the generated face.

Tracking faces in video streams is the estimation of the positions of a set of facial feature points in each frame. As human face is deformable, the relative distance between feature points varies significantly between facial expressions.

The approach to tracking facial feature points is inspired by Active Shape Model (ASM) proposed by Tim Cootes. However, in the context of face tracking, their method lacks robustness for detecting feature points in motion and is not efficient enough to generate accurate estimations on-the-fly, an alternative method to fit face shapes to replace the original design in ASM. A series of experiments are conducted in this research and a critical analysis is provided.

This thesis also discusses the impact of training data on tracking performance. The performance of generic shape models trained from a publicly available database are compared with its counterpart of person-specific models trained from the author's self-portrait photographs.

Table of Contents

List of Figures

Chapter 1: Introduction

1.1. Background

Facial feature point detection is estimating the location of a set of facial landmarks in images. A variety of ideas from computer graphics, computer vision, machine learning and image processing has been implemented in this context. Although this was once a heated topic in the late 1990s, and vast amount of research has been conducted, due to the limitation of hardware performance, it is only until the last decade that the commercial applications involving detection in video streams is made possible. Feature point detection is the key part of a variety of applications that manipulate facial captures. For instance, the video chat application Google Hangout introduced a feature that let users add virtual masks, glasses, hats and beards to themselves to help users overcome their innate distaste for video conversation. In 2016, Facebook launched its video filter application, MSQRD, which let users retarget their faces in video stream with a wide variety of 3D avatars. Many video game and movie companies used MSQRD to promote their products on Facebook making face swapping AR (augmented reality) applications very trendy on social networks. Nowadays, multiple social network providers have developed their own face-tracking filter applications. Apple announced their new Animoji (animated emoji) feature in iOS 11 that comes with their new device in September 2017, which makes use of the front camera and depth sensor, to track the user's expression then retarget it to an Animoji. It went viral on the media and became one of the major selling points of this device.

1.2. Objectives

The first objective of this research is to develop a system that tracks the face from video streams captured by a webcam on-the-fly, with the assumption that the video streams contain only one frontal face. The second objective is to estimate the orientation of the face, so that the face can be rendered with realistic shading. The third objective is to explore ways to improve accuracy, robustness and efficiency of the system.

1.3. Contributions

Firstly, this thesis introduces a system that can generate animated 3D faces in real time, based on the ideas of Active Shape Model (ASM) and POS with Iterations algorithm (POSIT). The structure of this system is illustrated in Figure 1. The output 3D face is controlled by human frontal face tracked in a video stream captured by commodity RGB camera, such as web cams and cameras on mobile phones. The pose of the human face is also estimated by analyzing the input video to create realistic shading on the generated face.

Secondly, this thesis introduces an alternative method for fitting face shapes for ASM to improve the robustness for detecting feature points in motion and the efficiency to generate accurate estimations on-the-fly, A series of experiments are conducted in this research and a critical analysis is provided.

Thirdly, this thesis discusses the impact of training data on tracking performance. The performance of generic shape models trained from a publicly available database is compared with its counterpart of person-specific models trained from the author's self-portrait photographs.

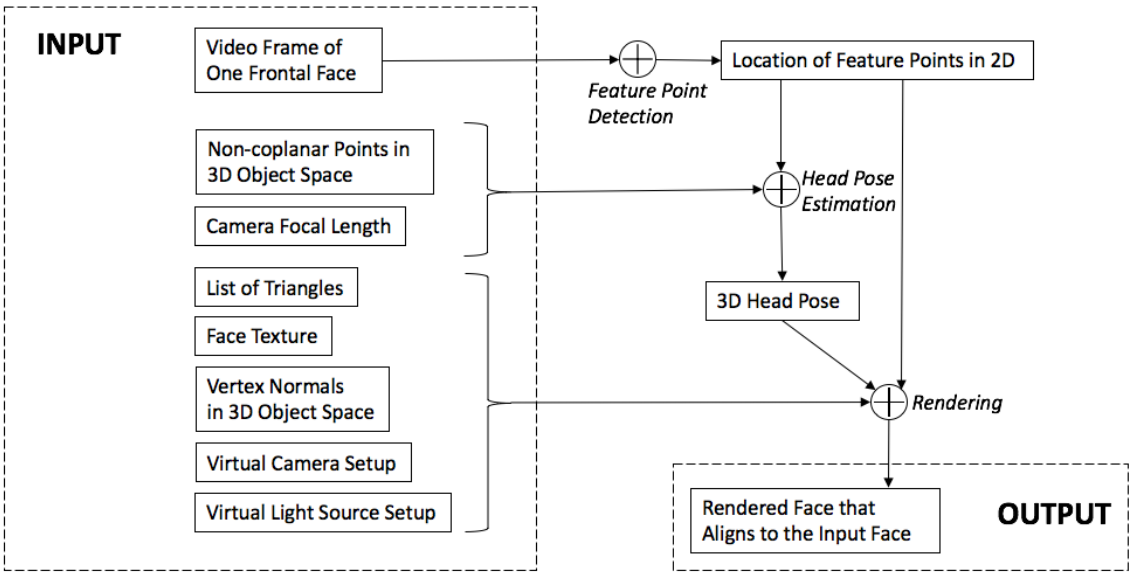

Figure 1 The structure of the proposed system

1.4. Summary of Thesis Organization

Chapter 2 is the literature review of the relevant methods applied in the field of face tracking and research topics related to the face tracking problem.

Chapter 3 features concepts of active shape model (ASM), and a more efficient and accurate face tracking approach inspired by ASM is introduced. A comparison on performance between the classic ASM approach and the proposed approach is provided. In addition, the comparison between generic shape models and person-specified shaped models is illustrated.

Chapter 4 is about estimating the head pose from a video stream, which leads to rendering the tracked face in 3D. In this chapter, POSIT is implemented on top of ASM to conclude the whole system. Details of implementation are revealed.

In Chapter 5, conclusions are drawn from the results obtained from the experiment shown in the previous chapters. The limitations and future works are also discussed.

Chapter 2: Literature Review

2.1. Landmark Locating

Facial landmark locating and tracking have been a heated topic in the field of computer vision for decades, which are prerequisite for applications such as video editing, face recognition, facial expression analysis, 3D face modeling, etc. Landmark locating addresses the problem of matching a group of predefined landmarks to a given facial image. Landmark tracking is to continuously capture the predefined landmarks in a facial image sequence. All the applications require accurate landmark positions. However, accurate landmark locating remains challenging in practice due to lighting conditions, background and pose variations.

The first step in most alignment algorithm [1][2][3] is to locate face region using face detectors [4]. The detected region is used as the initial positions, which are critical to the locating algorithm. For example, active appearance model (AAM) [5] is very sensitive to initialization because the complexity in appearance model affected by illumination may result in local minima.

In [6] and [7], the authors proposed multi-view face shape models to solve the pose variation problem, however, they cannot deal with unlimited possibilities of view change. Therefore, some researchers proposed 3D shape model [8][9] to handle continuous view change.

Traditional parametric methods such as active shape model (ASM) [10] and AAM [5] have been proved successful in terms of applicability. These models statistically parameterize a facial shape by applying principal component analysis (PCA). The major

difference between ASM and AAM is that, ASM maintains a linear system that only describes the shape of each face using a point distribution model (PDM), whereas the linear system of AAM describes both the shape and the appearance of face. However, they are both sensitive to initial shapes and there are many work focusing on mitigating the errors.

To enhance the performance of ASM, a variety of approaches are proposed. A landmark selection scheme based on Gaussian mixture model is proposed in [11]. It selects landmarks that correspond to the most reliable features, whose locations are relatively invariant to external factors. The work in [12] shows that, if the initial positions of three particularly important landmark points can be assigned manually, the accuracy can be improved. The authors also claim that adding a reasonable weight factor for each landmark increases the performance of ASM.

One conventional approach to locating landmarks in an image for ASM is using a collection of greyscale patches for face region around landmarks obtained from a training set of frontal face images by averaging the intensities of corresponding regions. Template matching is performed with the patches to find the areas of the input image that matches them. The work in [13] uses the same method of building the linear system of appearance model from AAM to build the profile of ASM to increase the accuracy of location.

2.2. 3D Pose Estimation

3D pose estimation is the problem of determining the rigid body transformation of an object given a 2D image of the object. It is possible to estimate the transformation if the 3D model of the object is known.

A common technique to solve this problem is the POSIT algorithm, where each reference point's scaled orthographic projection is firstly assumed, and helps to update the pose of object iteratively until a good estimation is found. A lot of research related to head pose tracking use this algorithm [14][15], because of its high accuracy given a small number of reference points and low time complexity. There exist other state-of-the-art techniques, where pose estimation is modeled as a perspective-n-points (PnP) problem, such as Efficient PnP (EPnP)[16], Procrustes PnP (PPnP)[17] and Robust PnP (RPnP) [18]. The aim of the PnP algorithms is to determine the position and orientation of a camera given its intrinsic parameters and a set of n correspondences between 3D points and their 2D projections. The main difference between PnP-based algorithms is the solver to the system of equations that describes the PnP problem.

In [19], the authors tested POSIT against a publicly available video database [20] of head poses from Boston University, where head motions are recorded under varying illumination conditions at a frame rate of 30 fps. The result shows that for estimating tilting, shaking, nodding head motions using POSIT, the mean absolute errors of head orientation are 5.27, 6.00 and 6.23 degrees respectively. Work in [21] shows similar result in terms of rotational estimation of POSIT, but it pointed out that the translational estimation is not ideal because POSIT has the scaled orthographic projection (SOP) assumption, while the true images are generated by a perspective projection. Work in [22] shows that the time complexity of POSIT can be regarded as constant in practice. The running time of POSIT is at least three times faster than all the PnP-based algorithms, and it is the fastest amongst all the conventional pose estimation algorithms. However, the authors pointed out that POSIT can be trapped in local minima in some cases.

2.3. Active Shape Model

Active shape models are statistic models of the shapes of one kind of objects that can iteratively fit to an instance of the objects that appears in arbitrary images. Hence, constructing an ASM consists of two parts: statistically training the shape model and defining the iterative approach to fit the shape model onto object in images. Normally, translation, rotation and scaling are not considered as part of shape models, so before training the shape model, the shapes from the data set should be aligned into a common coordinate system. The most popular approach for alignment is Procrustes Analysis [23].

2.3.1. Statistic Models of Shape

Suppose that in the training set, there are N shapes that are aligned into a common coordinate system. Each shape contains n landmarks, which implies that it is represented by a 2n-dimensional vector. Hence, these vectors form a distribution in the 2n-dimensional space. If the distribution can be modeled, arbitrary shapes similar to those in the training set can be generated, however, the dimensionality makes it hard to model the distribution. Therefore, it is necessary to reduce the dimensionality. An effective approach is to apply Principal Component Analysis (PCA) [24] to the data. PCA computes the mutually perpendicular axes, the eigenvectors, of the distribution in the 2n-dimensional space, and takes the only axes that exhibit major variance as the axes for a new multi-dimensional space of lower dimensionality.

For example, if a shape is originally represented as

$$\mathrm{x} = (x_1, \dots, x_n, y_1, \dots., y_n)^T \qquad \text{(eq. 2. 3. 1)}$$

It can also be represented using the eigenvectors as

$$\mathrm{x} \approx \bar{\mathrm{x}} + Pb,\ P = (p_1, \dots, p_k)^T \qquad \text{(eq. 2. 3. 2)}$$

where $\bar{x}$ is the mean shape in the distribution; $p_1, \ldots, p_k$ are the eigenvectors. Therefore, the parameters *b* that represent x in the k-dimensional space are

$$b = P^T(x - \bar{x}) \qquad \text{(eq. 2. 3. 3)}$$

In conclusion, in the principal component space, an arbitrary shape from the common coordinate system can be approximately represented as a linear combination of the principal components added to the mean shape in the distribution.

2.3.2. Image Interpretation with Models

The iterative approach to improve the fit of a shape instance x, to an image is listed as follows.

1. Examine a region of the image around each point x_i to find the best nearby match
2. Update the parameters for translation, rotation, scaling and the parameters of the principal components to best fit the newly found shape
3. Apply constraints to the parameters of principal components, to ensure the newly fitted shape is valid, then store the result as x
4. Repeat until convergence

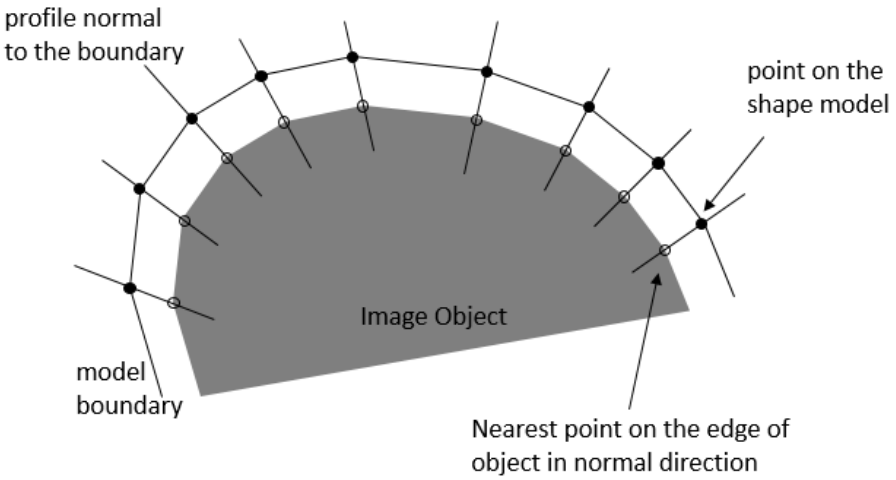

Figure 2 Profile normals to the boundary

To fit the points, we look along profiles perpendicular to the shape boundary through each points, as illustrated in Figure 2. If the diffence between the pixels on both side of the image object boundary is obvious, each point can be fitted by searching for the greatest differences along the corresponding profile normal. However, this assumption is not necessarily true in practice. In ASM, what to look for in the target image is firstly learned. This is done by sampling along the profile normal of each landmark on the boundary of object in each training image. Based on the result, a grey-level statistical model is built, which is refered to as the profile model. Each component of the profile model is determined by normalizing the sum of every sampled profile.

When fitting the shape, the profile model is the baseline to compare with sampled profiles. ASM searches along each sampled profile to find the best fit, which has the smallest Mahalanobis distance [33] to the profile model. This process is illustraed in Figure 3, which shows that it is a sliding window alogrithm and the best fit occurs at the first position to the right of the middle position of the sampled profile.

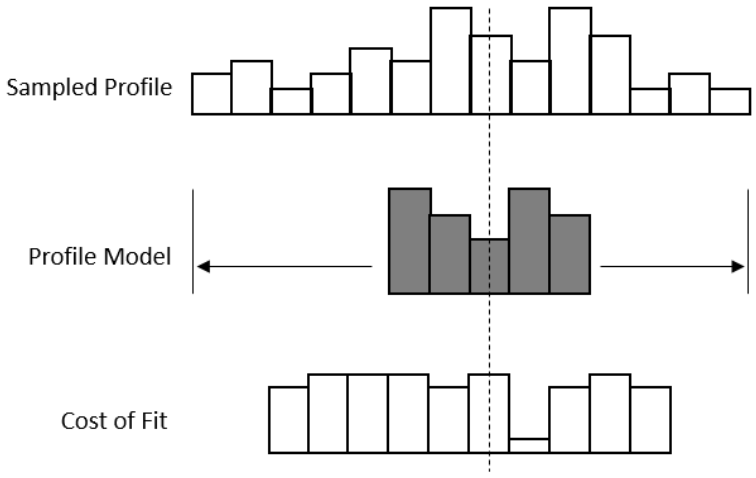

Figure 3 Search for best fit of profile model

One way to improve the robustness of the algorithm is to implement it in a multi-resolution framework. This means searching is firstly done in a low resolution image to find the approximate location of feature points then repeat the search in images of higher resolutions. This approach is less likely to fit feature points onto wrong image structures. As shown in Figure 4, for each image, a Gaussian image pyramid is built by smoothing and subsampling. The lowest level of the pyramid is the original image, when it goes one level up, the size of the image in each dimension halves. Accordingly, the profile model should be duplicated and changed in width for each level of the Gaussian pyramid.

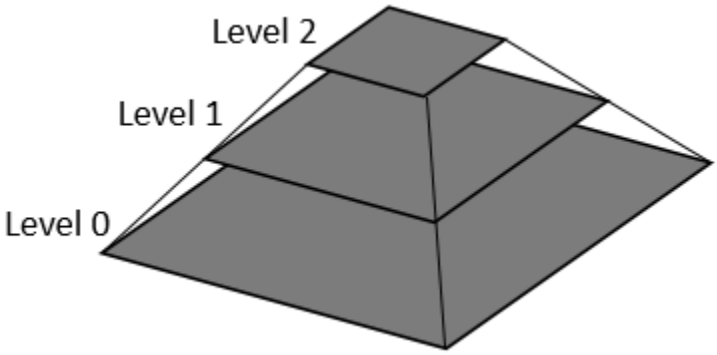

Figure 4 A Gaussian Image Pyramid

2.4. POSIT

POSIT (Pose from Orthography and Scaling with ITerations, DeMenthon & Davis 1995) is an iterative algorithm that finds the pose of an object from a single image showing at least four non-coplanar feature points of the object if their relative geometry on the object is known [25]. The ideas of perspective projection as well as scaled orthographic projection, as illustrated in Figure 5, are exploited to solve this problem. The rest of this section is a detailed review of the development of POSIT presented in [25].

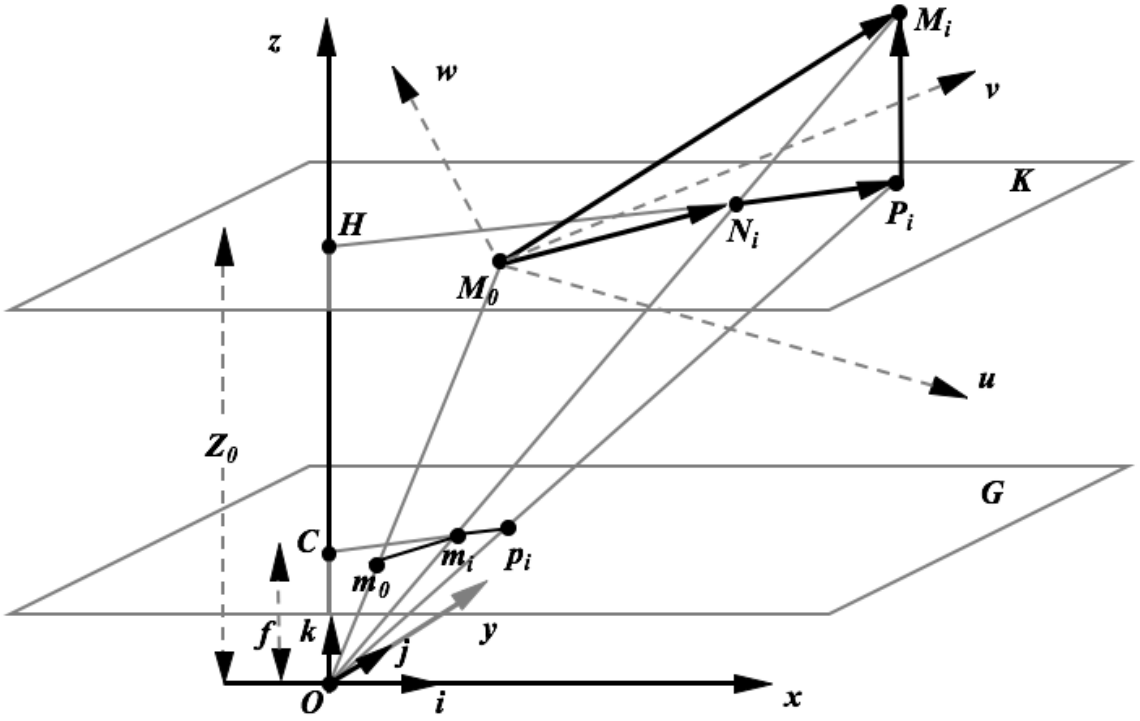

Figure 5 Perspective projection and scaled orthographic projection

2.4.1. Pose from Orthography and Scaling (POS)

A captured image is the perspective projection of the object from the 3D camera coordinate system onto the 2D image plane. As the image plane, G, is associated with the camera's viewing frustum, the pose of object can be represented by the viewing transformation from the object coordinates to the camera coordinates. As shown in Figure 5, O is the origin of camera coordinate system, which uses vectors i, j, k as axis directions; with a focal length f, the projection of O on image plane is C; M_0 is the origin of the object coordinates; M_i is a visible vertex on the object; the vectors that define the object coordinate system are u, v and w; plane K is parallel to G and goes through M_0. On plane K, H is the orthogonal projection of O; P_i is the orthogonal projection of M_i; N_i is the perspective projection of M_i; On image plane, m_0 is the perspective projection of M_0; m_i

is the perspective projection of M_i and N_i; p_i, is the perspective projection of P_i and is also known as the scaled orthogonal projection (SOP) of M_i.

The viewing transformation from object coordinates to camera coordinates is a translation from M_0 to O followed by a rotation to alight u, v, w with i, j, k. Hence the transformation can be presented with the homogeneous matrix shown in eq. 2. 4. 1.

$$T = \begin{bmatrix} i_u & i_v & i_w & -X_0 \\ j_u & j_v & j_w & -Y_0 \\ k_u & k_v & k_w & -Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(eq. 2. 4. 1)}$$

In the viewing frustum, the coordinates of M_0 and M_i are (X_0, Y_0, Z_0) and (X_i, Y_i, Z_i). On the image plane, the coordinates of m_0, m_i and p_i are (x_0, y_0), (x_i, y_i) and (x'_i, y'_i). There exist similar triangles that produce the following relations:

$$x_i' = x_0 + \frac{f}{Z_0}(X_i - X_0) \quad , \qquad y_i' = y_0 + \frac{f}{Z_0}(Y_i - Y_0) \qquad \text{(eq. 2. 4. 2)}$$

The vector M_0M_i is the sum of three vectors

$$M_0M_i = M_0N_i + N_iP_i + P_iM_i \qquad \text{(eq. 2. 4. 3)}$$

Because triangles M_0N_iO and m_0m_iO are similar, and $M_iN_iP_i$ and Om_iC are similar, we have

$$M_0M_i = \frac{Z_0}{f} m_0m_i + k \cdot \frac{M_0M_i}{f} Cm_i + P_iM_i = \frac{Z_0}{f} (m_0m_i + \frac{M_0M_i}{Z_0} Cm_i) + P_iM_i \qquad \text{(eq. 2. 4. 4)}$$

Because triangle $M_iN_iP_i$ is also similar to triangle ON_iH and line segments HP_i and Cp_i are parallel, eq. 2.4.4 can be re-written as

$$M_0M_i = \frac{Z_0}{f}(m_0m_i + m_ip_i) + P_iM_i = \frac{Z_0}{f}m_0p_i + P_iM_i \qquad \text{(eq. 2. 4. 5)}$$

Because vector P_iM_i is perpendicular to the image plane, performing dot product on eq. 2.4.5 with vector i and j respectively will provide

$$i \cdot \frac{f}{Z_0}M_0M_i = x_i' - x_0 \qquad j \cdot \frac{f}{Z_0}M_0M_i = y_i' - y_0 \qquad \text{(eq. 2. 4. 6)}$$

Because p_i is an imaginary point on the image plane, and line Cp_i goes through point m_i, and C is the origin of the image coordinates, there exist ε_i such that

$$x_i' = x_i(1 + \varepsilon_i) \qquad y_i' = y_i(1 + \varepsilon_i) \qquad \text{(eq. 2. 4. 7)}$$

All the parameters in the matrix shown in eq. 2.4.1 are unknown. To reduce the number of unknowns, define vectors I and J as the scaled i and j vectors.

$$I = \frac{f}{Z_0}i \qquad J = \frac{f}{Z_0}j \qquad \text{(eq. 2. 4. 8)}$$

Based on eq. 2.4.6, eq. 2.4.7 and eq. 2.4.8, the linear system representing the pose estimation problem is determined as

$$M_0M_i \cdot I = x_i(1 + \varepsilon_i) - x_0 \qquad M_0M_i \cdot J = y_i(1 + \varepsilon_i) - y_0 \qquad \text{(eq. 2. 4. 9)}$$

Note that if ε_i can be found, the matrix in eq. 2.4.1 can be determined, because i and j are normalized I and J; k is the cross product of i and j, Z_0 can be solved by calculating the magnitude of I and J. X_0 and Y_0 can be calculated using similar triangles OCm_0 and OHM_0.

An algorithm, known as Pose from Orthography and Scaling (POS), to solve eq. 2.4.9 was proposed in [26]. However, it only provides approximation if the value given to ε_i are not exact which is hard to predict.

2.4.2. Iterative solution of POS problems

The exact values of ε_i can be found iteratively as proved in [25]. The algorithm is referred to as POSIT (POS with iterations). POSIT starts with an initial guessed value of ε_i and iteratively adjust it to finally arrive at the exact solution. Based on a variety of similar triangles that appear in Figure 5, one way to present ε_i is

$$\varepsilon_i = \frac{1}{Z_0} M_0 M_i \cdot k \qquad \text{(eq. 2. 4. 10)}$$

Eq. 2.4.10 suggests that if the object is infinitely far away from the camera in the z-direction, ε_i is equal to 0, because the SOP of M_i and the perspective projection of M_i coincide. Compared with Z_0, the dimensions of the object being tracked is relatively small. Hence, it is fair to let ε_i start as 0. It should be also noticed that when tracking an object in a video stream, the initial guess for ε_i in one frame should be similar to the results from the previous frame due to temporal coherence supposing the object does not move too fast.

Suppose there are n visible object points in the image, eq. 2.4.9 can be written in matrix format

$$AI = X' \ , \quad AJ = Y' \qquad \text{(eq. 2. 4. 11)}$$

with

$$A = \begin{bmatrix} U_0 & V_0 & W_0 \\ \vdots & \vdots & \vdots \\ U_i & V_i & W_i \\ \vdots & \vdots & \vdots \end{bmatrix}, \qquad I = [I_u \quad I_v \quad I_w]^T, \qquad J = [J_u \quad J_v \quad J_w]^T$$

$$X' = \begin{bmatrix} x_1(1+\varepsilon_1) - x_0 \\ \vdots \\ x_i(1+\varepsilon_i) - x_0 \\ \vdots \end{bmatrix}, \quad Y' = \begin{bmatrix} y_1(1+\varepsilon_1) - y_0 \\ \vdots \\ y_i(1+\varepsilon_i) - y_0 \\ \vdots \end{bmatrix} \qquad \text{(eq. 2. 4. 12)}$$

where the ε_i terms have the calculated value from the previous iteration. If the rank of matrix A is 3, it suggests there exist at least three object points other than M_0, so that there are at least four non-coplanar points in total. Then matrix A can be pseudo-inversed and the unknown vectors can be calculated with the least square method as

$$I = BX' \quad , \quad J = BY' \qquad \text{(eq. 2. 4. 13)}$$

where the pseudo-inverse matrix B can be calculated as $[A^TA]^{-1}A^T$ or by decomposing matrix A by singular value decomposition (SVD).

The requirement that matrix A has a rank of at least 3 can also be interpreted geometrically. If the tail of vector I is at M_0, eq. 2.4.9 suggests that I lies in the plane that is perpendicular to M_0M_i. If there are three visible point M_1, M_2 and M_3 that contributes to matrix A, this means vector I's head is at the intersection point of plane M_0M_1, M_0M_2 and M_0M_3. If the four points are coplanar, there exists no intersection point. Similarly, this statement holds true for vector J. It is possible that the lengths of I and J are different in some iterations, which suggest two calculated values of the scale factor f/Z_0. The average scale factor is chosen to calculate Z_0, which is used to update the estimation of ε_i, as shown in eq. 2.4.10.

The iterative update mechanism of POSIT can be summarized as the following steps.

1. Create the matrix A of dimension (N - 1) × 3, where each row is a vector M_0M_i connecting the reference feature point M_0, which is also the origin of the object

coordinate system, with another feature point M_i; then compute the 3 × (N - 1) matrix B, the pseudoinverse matrix of A.

2. n = 0; for i = 1, 2, … N – 1, $\varepsilon_{i(n)} = 0$.
3. n = n + 1; Create vector X' and Y', then calculate vector I and J using eq. 2.4.13. Normalize I and J to get i and j. Note the scale factor as the average length of I and J, and calculate Z_0 with s and focal length f. $s = \frac{|I|+|J|}{2}$, $Z_0 = \frac{f}{s}$
4. Compute unit vector k as the cross product of i and j; compute new $\varepsilon_{i(n)}$ values using eq. 2.4.10.
5. Check convergence. If $|\varepsilon_{i(n)} - \varepsilon_{i(n-1)}|$ is not greater than the threshold, go to Step 6; otherwise, go to Step 3
6. Generate the homogeneous matrix in eq. 2.4.1 with parameters from the last iteration, to indicate the pose of the object. The translation vector OM_0 can be calculated as Om_0 / s; the rotation matrix is made up with unit vectors i, j, k.

2.5. Summary

In this chapter, a variety of relevant researches are reviewed. ASM and POSIT are discussed in detail, as they are the key concepts of the proposed approach to generating 3D faces. Their implementation in the proposed approach will be presented in the following chapters.

Chapter 3: Face Tracking Using the Active Shape Model

This chapter describes a system that transforms a video stream filmed by RGB camera featuring a human frontal face to a polygon mesh of face in real-time. Procrustes analysis and PCA is implemented to train a linear shape model that represents both rigid and non-rigid transformations in the 2D image plane. In addition, a greyscale patch model is trained based on correlation, to detect the location of feature points in the video. Both models are trained from a data set of annotated faces.

3.1. Training Set

The algorithm for detecting the locations of facial feature points in a frame of video relies on geometrical dependencies between these points and their counterparts from a structured model trained offline. Thus, firstly, geometric data of a collection of faces is required as a training set. The data can be obtained by manually annotating images of faces to get the coordinates of facial feature points, which are on the contours of face and facial features. The collection should include images featuring adequate facial expressions taken in different directions, where all face features are visible. Examples of annotated images are illustrated shown in Figure 6, where symmetric points are annotated with the same color and points on the same contour are connected. The annotation of symmetric points is necessary for generating mirrored images to provide more training samples. The connections are essential for calculating profile normals for shape fitting using classic ASM. In this thesis, fitting 1D profile models is used as a baseline to compare with the proposed fitting approach. The performances of feature points fitting using the shape model trained with a generic data set and its counterpart trained with a person-specific data set are also compared. The MUCT database [27], which contains

3755 faces of more than 300 individuals, is selected as the generic data set. The person-specific data set is 91 self-portraits of the author with a variety of facial expressions, taken under the same lighting condition.

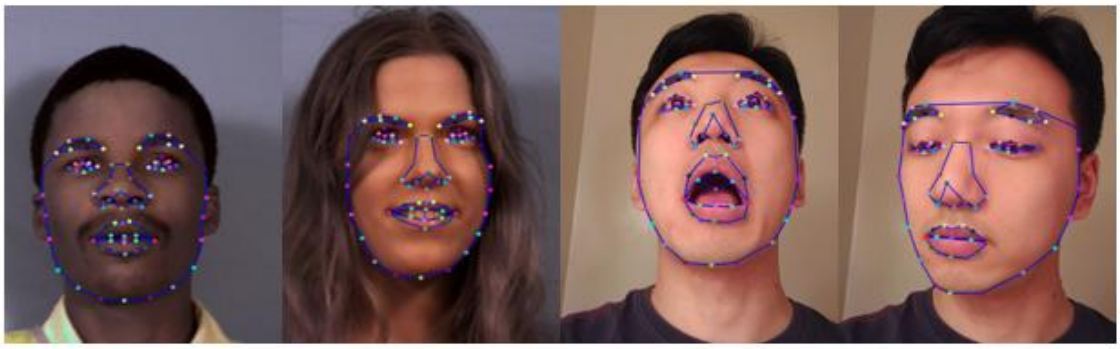

Figure 6 Examples of annotated images

Data obtained from each image should be of the same size, and store the coordinates of feature points in the same order. When training the models, it is necessary to use the original images as well as data from their mirrored counterparts, because it not only doubles the size of the training set, but also evens out the bias of head poses in the collection, otherwise the result reference shape may lean slightly to one side. The comparison is illustrated in Figure 7.

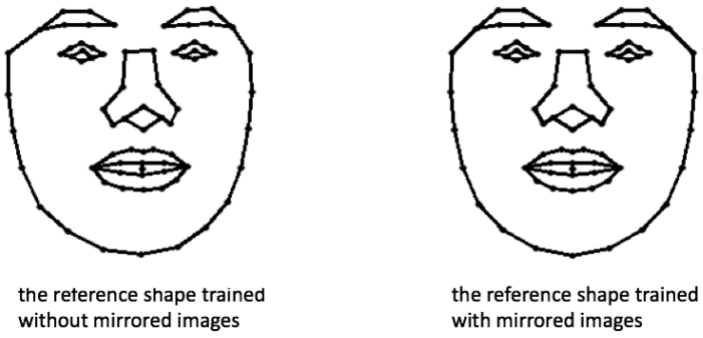

Figure 7 Reference shapes trained with and without mirrored images

The definition of feature points is illustrated in the Appendix. One way to triangulate the collection of feature points is also illustrated in the Appendix, which is used for rendering the face as to be described in the next chapter.

3.2. Procrustes Analysis

To build a shape model of faces from the annotated training set, transformations pertaining to global rigid motions should be removed first. When modelling geometry in 2D, a rigid motion is represented as a similarity transform; this includes the scale, 2D rotation and translation. Figure 8 illustrates the set of permissible motion types under a similarity transform.

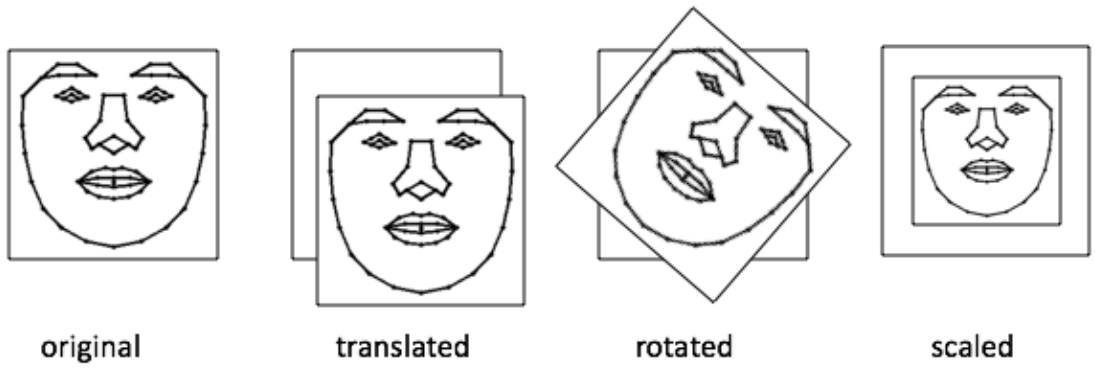

Figure 8 Rigid transformations

Procrustes is the name of a notrious bandit in Greek mythology, who torture people by stretching them or cutting off their legs, so as to force them to fit the size of an iron bed. The process of removing global rigid motion from a collection of points is called Procrustes analysis. It is a conventional approach to pre-processing the data set to train the ASM. The coordiantes of facial feature points in Figure 9 are color coded, where symetric points are in the same color. The left image illustrates directly putting the coordinates on the same image plane; the middle image shows the plotting with only

traslating to align the centers of faces, where the center of each face is calculated as the average of all feature points on the face; the right image shows the Procrustes aligned plot by translating firstly, then rotating and scaling with respect to the same reference. Note that the three images are deliberatly drawn to the same scale. It can be observed that in the raw images, the same landmark from different faces rarely algin; aligning the centers of faces by translating limits feature points of the same kind to a certain area, but the areas are relatively large and overlapping; and Procrustes aligned faces let the training set of feature points of the same kind distribute in a relatively small range and generally distinguished from points of other kinds.

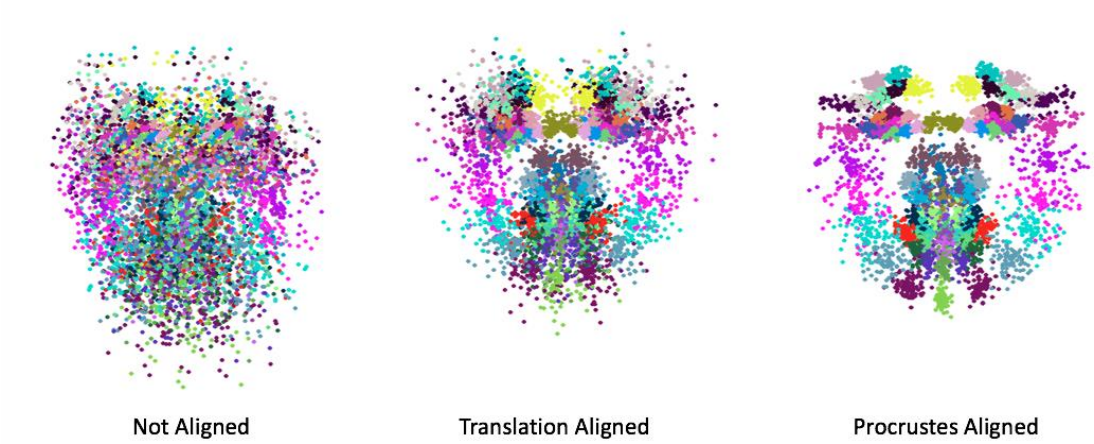

Figure 9 Illustration of Procrustes alignment

The training set of faces is a collection of N shapes, each of which is presented as a set of n points in the 2D plane. Rigid transformation is applied to each shape by translating its center of mass $(\bar{x},\ \bar{y})$ to the origin of image plane to get its translation aligned shape P_i , where

$$\bar{x} = \frac{1}{n}\sum_i x_i,\ \bar{y} = \frac{1}{n}\sum_i y_i \qquad \text{(eq. 3. 1)}$$

The Procrustes aligned shapes and their canonical reference C can be iteratively obtained in the following steps:

1. Set the reference shape *C* as the unit vector of the average of aligned faces,

$$C = \frac{F}{|F|} \quad , \quad F = |\frac{1}{N}\sum_i P_i| \qquad \text{(eq. 3. 2)}$$

2. For each P_i, update it by using least square method to find the scale and rotation to best align it to C.
3. Update C with eq. 3.2, after taking a record of old *C*.
4. Yield the Procrustes aligned shapes P_1, P_2... P_N, and reference shape C, if the deviation from *C* to old *C* is not greater than the convergence threshold; otherwise go to step 1.

Intuitively the alignment in Step 2 consists of rotation and scaling, which provides two variables θ and s and the transformation matrix is

$$T = \begin{bmatrix} s \cdot cos\theta & -s \cdot sin\theta \\ s \cdot sin\theta & s \cdot cos\theta \end{bmatrix} \qquad \text{(eq. 3. 3)}$$

However, sine and cosine add to the complexity of calculation. Without increase the number of degrees of freedom, a and b can be defined as

$$a = s \cdot sin\theta, \qquad b = s \cdot cos\theta \qquad \text{(eq. 3. 4)}$$

The optimization problem in Step 2 is minimizing the residual R defined as

$$R = \min_{a,b} \sum_{i=1}^{n} \left| \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} - \begin{bmatrix} x_{c,i} \\ y_{c,i} \end{bmatrix} \right|^2 \qquad \text{(eq. 3. 5)}$$

and re-formatted as

$$R = \min_{a,b} \sum_{i=1}^{n} \left\| \begin{bmatrix} x_i & -y_i \\ y_i & x_i \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} - \begin{bmatrix} x_{c,i} \\ y_{c,i} \end{bmatrix} \right\|^2 \quad \text{(eq. 3. 6)}$$

The values of a and b that minimize R can be found by making the partial derivative of R on a and b, equal to zero. In matrix format, it is

$$\begin{bmatrix} x_1 & -y_1 \\ y_1 & x_1 \\ \vdots & \vdots \\ x_i & -y_i \\ y_i & x_i \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} - \begin{bmatrix} x_{c,1} \\ y_{c,1} \\ \vdots \\ x_{c,i} \\ y_{c,i} \\ \vdots \\ \vdots \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ 0 \\ \vdots \end{bmatrix} \quad \text{(eq. 3. 7)}$$

The result is

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{\sum_{i=1}^{n} (x_i^2 + y_i^2)} \sum_{i=1}^{n} \begin{bmatrix} x_i x_{c,i} + y_i y_{c,i} \\ x_i y_{c,i} - y_i x_{c,i} \end{bmatrix} \quad \text{(eq. 3. 8)}$$

Note that the shapes in the data set are required to separate the training of the rigid and non-rigid transformation basis. The reference shape *C* is essential because all face shapes are considered as a warp of *C* in a multi-dimensional space. The Procrustes aligned shapes P_1, P_2... P_N are then used to statistically train the shape model.

3.3. Shape Model

If each face has n feature points, it implies that each face is represented by a 2n-dimensional vector. It is vital to generate a new multi-dimensional space to describe the face model, because the 2n-dimensional space only provides the coordinates of each selected feature in the x-y plane. It is not efficient to manipulate a face shape point by point to recognize its rigid transformation and deformation. Hence, we need another representation that reflects the two kinds of transformations for a face shape. In addition, the smaller the number of dimensions is, the less computing expenses are required. This

is especially critical in the tracking phase, where all selected features should be recognized one-the-fly from a video stream. Hence, we need a representation that is of significantly smaller degrees of freedom. In Tim Cootes' original work of ASM, how to represent the scaling and rigid transformation of arbitrary shapes is not discussed. This can be parameterized with respect to a set of ortho-normal vectors in a format that is similar to the eigenvectors for deforming face shapes. In the rest of this section, the former set of vectors are referred to as the rigid basis and the latter set of vectors are referred to as the non-rigid basis.

3.3.1. Rigid Basis

The rigid basis of face shapes is a 4-dimensional space associated with rigid transformations and scaling. The normalized vectors representing the 4 dimensions are obtained from the reference shape *C*, which is introduced in Section 3.2.

The rigid transformation to get an arbitrary face shape *X* that is similar to *C* can be written in the following format, where X represents the result of first rotate and scale one face shape then translate it; columns of matrix B are components of X; and p is the corresponding coefficients.

$$X = \begin{bmatrix} \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \begin{bmatrix} x_{c,1} \\ y_{c,1} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \\ \vdots \\ \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \begin{bmatrix} x_{c,n} \\ y_{c,n} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \end{bmatrix} = \begin{bmatrix} x_{c,1} & -y_{c,1} & 1 & 0 \\ y_{c,1} & x_{c,1} & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{c,n} & -y_{c,n} & 1 & 0 \\ y_{c,n} & x_{c,n} & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ t_x \\ t_y \end{bmatrix} = Bp \qquad \text{(eq. 3. 9)}$$

$C = [x_{c,1}, y_{c,1}, \ldots, x_{c,n}, y_{c,n}]^T$, is the first column of *B.* This equation shows that a linear combination (with rotation and scale constraint on *a* and *b*) produces a shape that is

similar to *C*. Hence, the rigid basis can be obtained by applying Gram-Schmidt ortho-normalization to *B* to get the rigid basis *R*.

The rigid base consists of 4 orthonormal column vectors of *2n* dimensions. Hence, a face shape's parameter associated with the first vector is its scale with respect to the normalized reference shape.

3.3.2. Non-rigid basis

This section describes the implementation of the statistical shape model proposed by Tim Cootes. Similar to the rigid basis, the non-rigid basis of a face shape is a k-dimensional space associated with non-rigid transformations. Applying the principal component analysis (PCA) to the Procrustes aligned shapes (P_1, P_2... P_N), can identify the non-rigid components in the shapes as well as their weights, each of which represent the portion of the corresponding non-rigid component observed in the data set. The least significant components should be removed from the shape model to reduce the dimensionality of the shape model. An example of applying PCA to a 2D data set is shown in Figure 10. The vectors suggest the major directions of variation for the data set. The lengths of vectors illustrate the importance of the directions. Figure 10 suggests that if it is safe to conclude that the 2D data is mainly distributed in one direction, the dimensionality of the data set can be reduced to 1D.

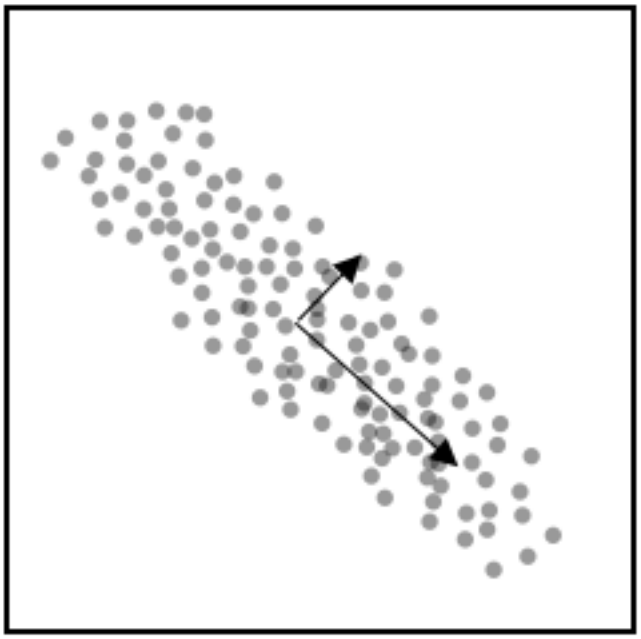

Figure 10 An example of PCA

Based on PCA, the non-rigid basis can be obtained in the following steps:

1. Subtract the rigid components from the Procrustes aligned shapes (P_1, P_2... P_N), to get their non-rigid components Y = (P'_1, P'_2 ... P'_N) using the following equation, where $R^T P_i$ is the rigid component's projection of shape P_i in the 4 directions for rigid transformations.

$$P'_i = P_i - RR^T P_i \quad \text{(eq. 3. 10)}$$

2. Apply principal component analysis (PCA) on Y. First apply the singular value decomposition on Y. The column vectors in its left-singular vector matrix U contains the principal components sorted in descending order. The Σ matrix contains singular values, which are the importance of each column vectors in U.

$$Y_{2n\times N} = U\Sigma V^T = [u_1 \quad u_2 \quad \dots \quad u_m]\begin{bmatrix}\sigma_1 & 0 & \dots & \dots & \dots & \dots & 0\\ 0 & \sigma_2 & 0 & \dots & \dots & \dots & 0\\ \vdots & \ddots & \ddots & \ddots & \dots & \dots & \vdots\\ 0 & 0 & 0 & \sigma_m & 0 & \dots & 0\end{bmatrix}\begin{bmatrix}v_1^T\\ v_2^T\\ \vdots\\ v_l^T\end{bmatrix} \quad \text{(eq. 3.11)}$$

3. Yield the first k column vectors $u_1, u_2 \dots u_k$ in U, whose σ values add up to a predefined threshold.

Figure 11 Examples of composite shapes

As *U* is orthonormal, these *k* vectors representing the subspace of deformations is also the non-rigid basis of the shape model. Note that now there is a collection of *k + 4* orthonormal vectors in the original *2n*-dimensional space that represents the rigid and principal deformation components of a shape, and these vectors are mutually linearly independent. It implies that a (*k + 4*)-dimensional space can be constructed to describe face shapes. Each face can be described as a linear combination of the k+4 orthonormal vectors. Figure 11 illustrates a few shapes that are not rotated and translated, which means the last 3 of the 4 orthonormal vectors representing the rigid basis weight 0 in these shapes.

Figure 12 shows the eigenspectra of principal components trained from the generic data set and the person-specific data set that make up 95% of the non-rigid components in the

corresponding data set. It can be observed that the person-specific data set has 9 eigenvectors within its spectrum and the generic data set has 13 eigenvectors within it spectrum. Given that the images in the person-specific data set contains abundant facial expressions, it can be concluded that individual differences of face shape dominate the size of the eigenspectrum.

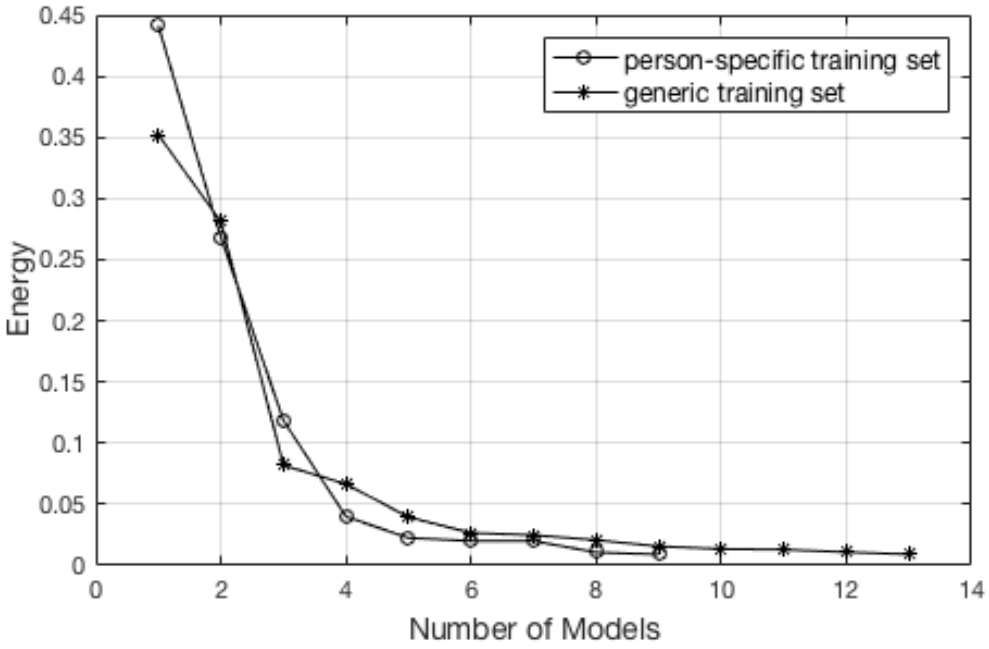

Figure 12 Eigenspectra of the non-rigid components

3.4. Shape Fitting

To improve the accuracy and efficiency of fitting the shapes in video frames, the patch model is introduced, which is inspired by the one-dimensional profile model of original ASM that fits shapes by shifting itself along the profile normal to find the point that provides maximum correlation to it. Analogously, the patch model is a two-dimensional profile model that fits shapes by doing template matching [28] to find the best fit. This collection of patches is also trained from the data set.

3.4.1. Patch Model

This section describes a conventional way to build the patch model. The images in the training set need to be converted to grayscale and natural logarithm need to be applied to the image pixel intensities, because log-scale images are more robust against differences in appearance and changes in illumination conditions. This also holds true when tracking feature points in video stream. Figure 13 shows a comparison between ordinary grayscale images and logarithm grayscale images. The contrast between the logarithm grayscale images are significantly smaller.

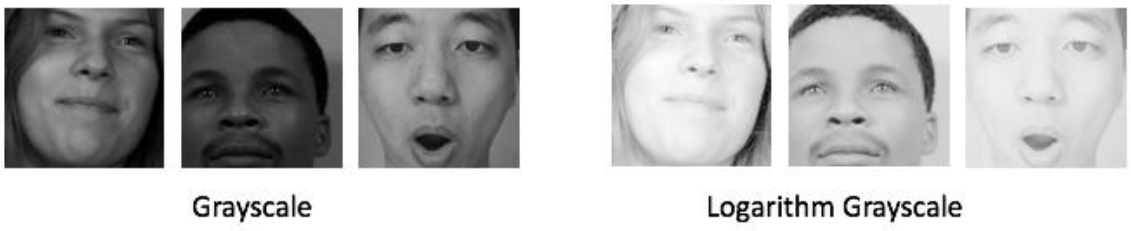

Figure 13 Grayscale images and logarithm grayscale images

When training the patch model, it should be guaranteed that each facial feature is aligned with its counterpart in a face shape C' that is created by providing the width of it in pixels to scale the canonical reference. In addition, the feature point is assumed to be at the center of the window on the image where correlation is calculated, as illustrated in Figure 14. However, the face can appear at any scale and rotation within the image plane. Hence, each feature point has a search window defined with respect to C', where the patch slides to find best fit. The center of the area that provides highest response is then annotated as a detected feature point.

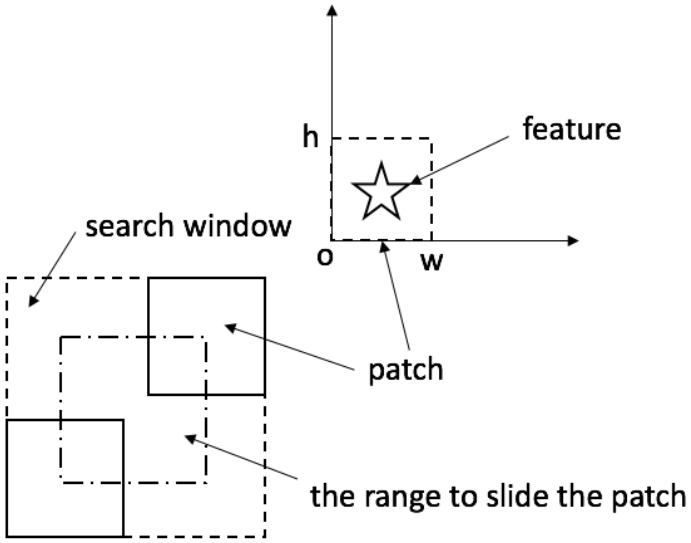

Figure 14 A patch and its search window

As illustrated in Figure 15, an arbitrary image can be warped so that the feature patch is normalized and centered at the window. If the patch being trained is for the j^{th} feature in the model using image *X* and its raw shape I_X, and the window's width and height are *W* and *H* respectively, the warp can be obtained from the inverse of the following steps:

1. $T_1 = \left(-\frac{W}{2}, -\frac{H}{2}\right)$ The feature is translated to the origin.

2. *RS* ←calc_simil*(X)* RS is the rotation and scale from the scaled reference shape *C'*, which is created by scaling C from Section 3.2, to raw *shape* I_X for training current feature patch.

3. $T_2 = (x_j,\ y_j)$ Translate the feature to align it with its original copy in image *X*.

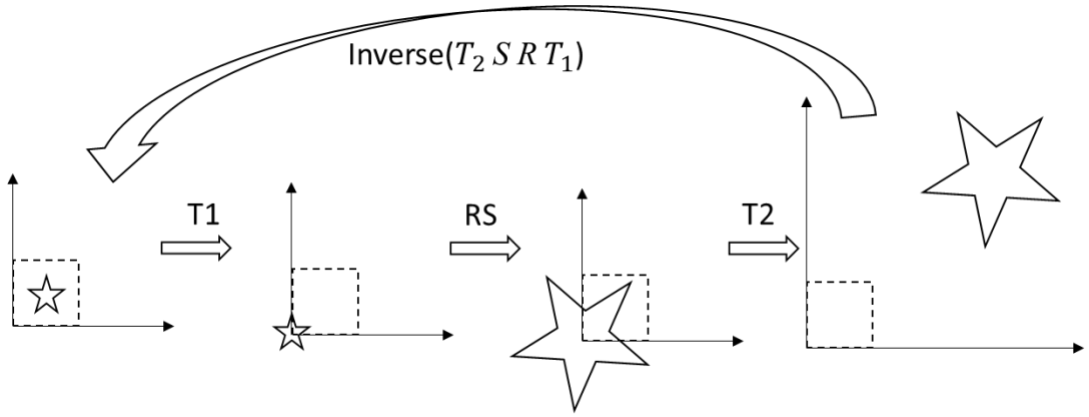

Figure 15 Transformation from raw input to search window

This transformation is similar to Procrustes alignment because the annotated feature points are rotated and scaled with respect to a reference shape, but all points are translated to the same area to train their corresponding patches. The same method is also used in the tracking phase.

Cross-relating a patch with an image region containing the corresponding facial feature yields a strong response and yields weaker responses everywhere else. The constraint can be expressed as minimizing a sum of squares:

$$\min_{P} \sum_{i=1}^{N} \sum_{x,y} [r(x,y) - L \cdot I_i(x - \frac{w}{2} : x + \frac{w}{2}, y - \frac{h}{2} : y + \frac{h}{2})]^2 \qquad \text{(eq. 3. 12)}$$

- Vector $\boldsymbol{L}$ denotes the patch for one facial feature
- Vector $\boldsymbol{I}_i$ denotes the i[th] training image
- *(a:b, c:d)* denotes the rectangular region whose top-left and bottom-right corners are located at (a, c) and (b, d) in image coordinates, respectively.
- $\boldsymbol{r}$ is a 2D-Gaussian distribution.
- *w* and *h* are the width and height of the patch.

As shown in eq. 3.12, the number of degrees of freedom of this problem is equal to the number of pixels contained in one patch, thus the computational cost to solve it is prohibitive. Stochastic gradient descent (SGD) is an alternative way to solving the learning problem. By regarding the learning objective as an error terrain over the degrees of freedom of the patch model, SGD iteratively makes an approximation of the gradient direction of the terrain and takes a small step in the opposite direction. The approximation to gradient can be computed by considering only the gradient of the learning objective for a single, randomly chosen image from the training set:

$$D = -\sum_{x,y}(r(x,y) - L \cdot W)W, \qquad W = I(x - \frac{w}{2} : x + \frac{w}{2}, y - \frac{h}{2} : y + \frac{h}{2}) \qquad \text{(eq. 3. 13)}$$

The patch is updated using the following equation:

$$L \leftarrow L + \eta(D - \lambda L) \qquad \text{(eq. 3. 14)}$$

where η is step size, λ is the regularization parameter to prevent overfitting. The subtraction of λL from the update direction effectively regularizes the solution from growing too large. This is a procedure that is often applied in machine-learning algorithms to promote generalization.

Figure 16 is a collection of screenshots captured when the patch for an eye corner is being trained. In each screenshot, the top left subplot illustrates the area of interest in a Procrustes aligned training image; the top right subplot shows the contrast between ideal and real response map; the bottom right subplot shows the current appearance of the patch; the bottom left subplot is the appearance of the residual used to update the patch. The intensities of the residual and patch are scaled in the drawings only to show their appearances.

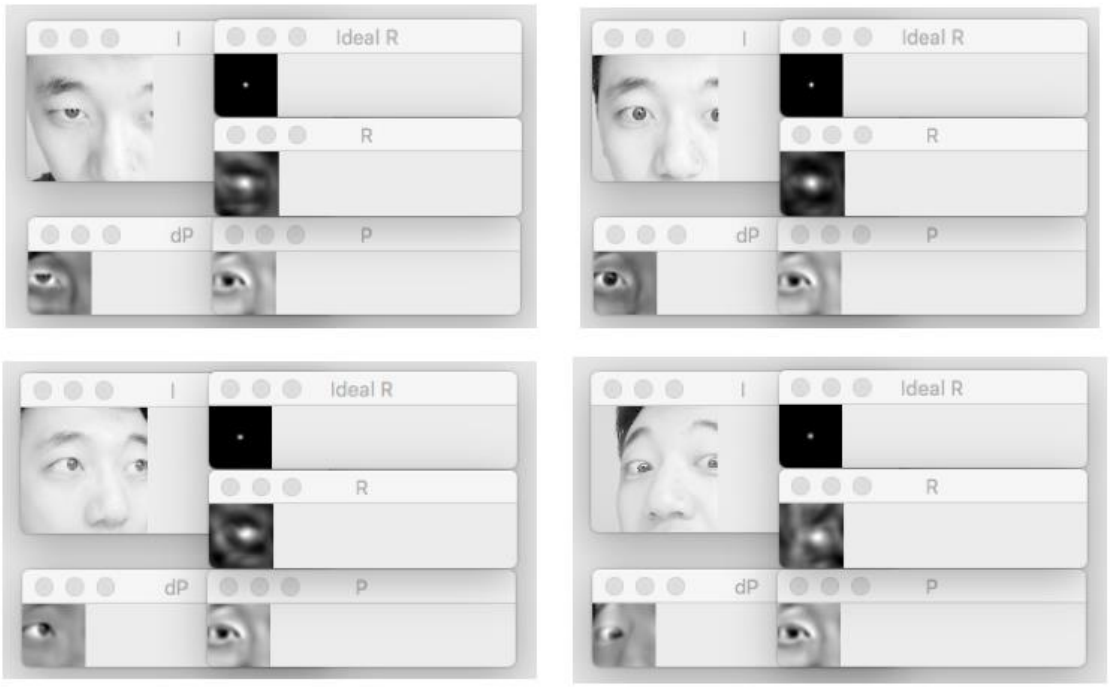

Figure 16 Training the patch model

Figure 17 illustrates the patch models by placing the patches at their corresponding points on the reference shape, which resembles the appearance of the averaged face in the training set. Patch models in the top row are trained from a collection of the author's self-portraits. Patch models in the bottom row are trained from the MUCT database, which contains 3755 faces of more than 300 individuals. The masculine look of the aligned patches indicates the male subjects dominate the data set.

The width of reference face and the size of each patch is needed for training the patch model. Both parameters should be specified in pixels. Width of face is essential for the transformation performed before template matching as illustrated previously in Figure 16. The size of patch added with the size of searching window is equal to the size of region of interest.

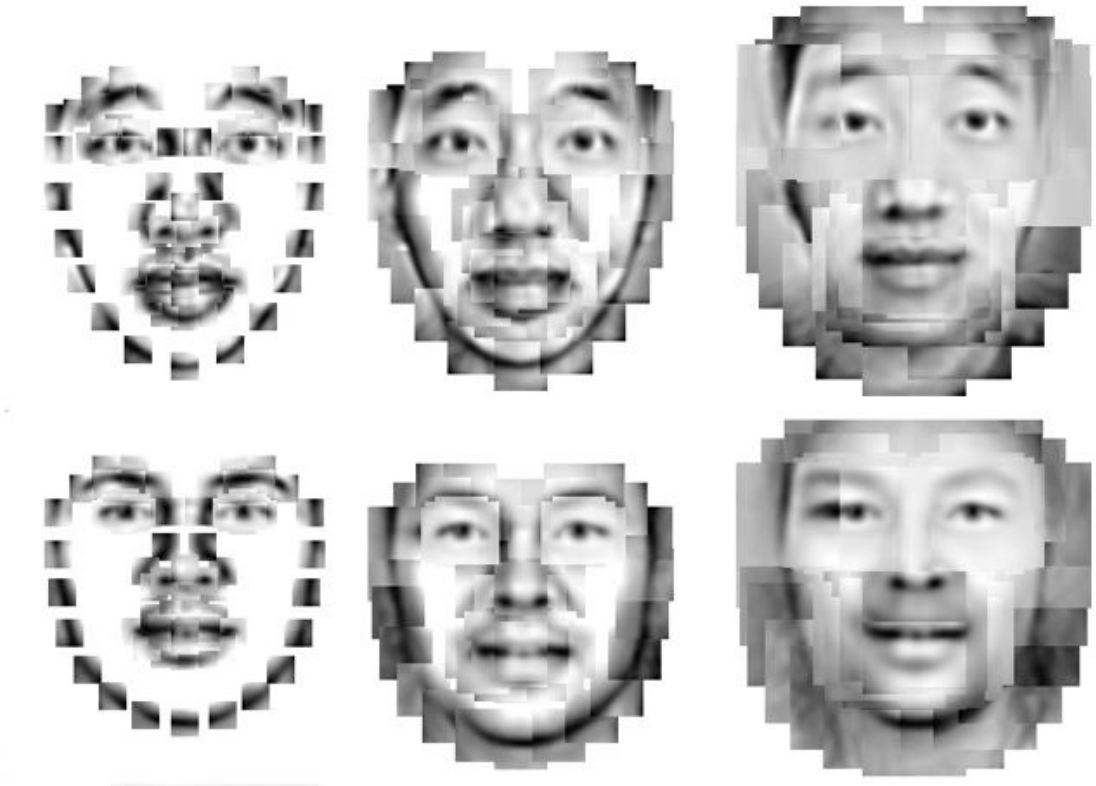

Figure 17 Patch models of person-specific data set and generic data set

3.4.2. Shape Initialization

The original ASM was proposed for shape fitting in an image, where the shape's initial position is normally the center of the image. Whereas, tracking the face in a video stream have strict requirement on efficiency. Hence, Haar cascades [29] or linear binary patterns (LBP) cascades [30] are required to detect the rectangular area that contains facial features and set the initial position of the face shape within this area. However, as Haar cascades or LBP cascades are not designated for feature points detection, there is an offset between the face shape and the detected rectangular area. It is necessary to calculate the averaged face offset against the rectangular area, which is used to generally predict the area where the feature points are located.

For fitting in one frame, when the initial guesses of the location of feature points are determined, the search window for each patch can be determined. Then, each patch slides through its corresponding search window to find the landmarks to let the 2D linear shape models fit onto for a refined estimation of feature point positions. However, applying Haar cascades or LBP cascades to every frame of the video stream is costly. It is assumed that the transformation of face shape from one frame to the next is relatively insignificant, so the cascades are only need once at the beginning of the video. For frames other than the first one, the location of feature points in the result shape from last frame is used as the initial guess.

3.4.3. Iterative Fitting

In the proposed approach, shape components are clamped during fitting to help the result converge and at the same time guarantee the detected points form a valid face shape. Each component is clamped within a range of 6 times the standard deviation observed in the training data. The ranges of shape components trained from the person-specific data set are illustrated in Figure 18.

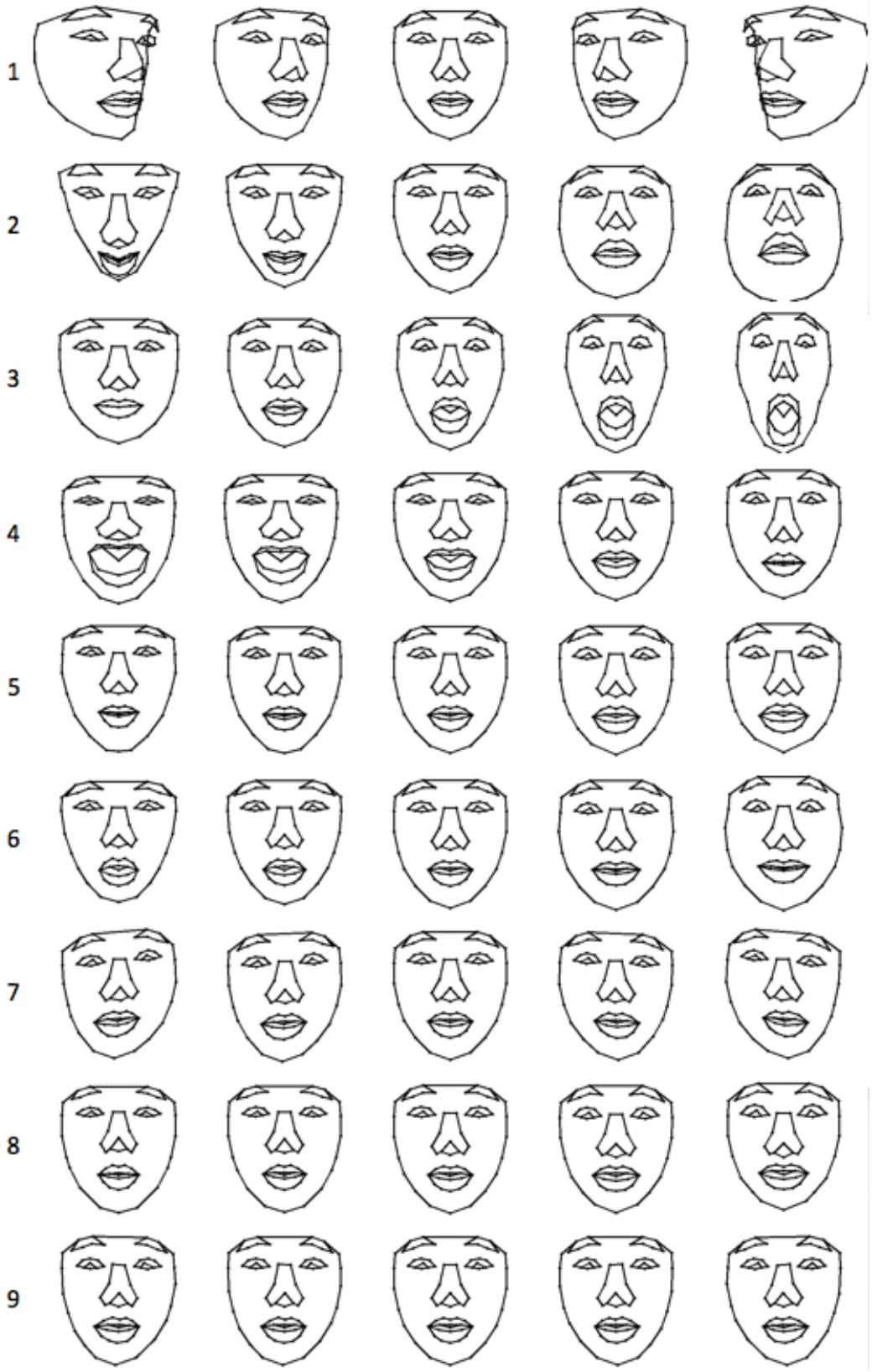

Figure 18 The ranges of non-rigid components of person-specific data set

In classic ASM, there is a convergence-based multi-resolution search method for point fitting. In this research, however, as the motion in video stream is coherent, the multi-resolution search method has obvious redundancy, because the initial guess for one frame is the result from the last frame making search in lower resolution inefficient. The fitting method for each frame of video is listed as the following steps.

1. For each feature point, set its location in the previous frame as the old fitting result.
2. Declare three searching levels, where the sizes of search window are in descending order.
3. For each feature point, do template matching in the current search level
4. Project the shape obtained in Step 3 to the principal component space, then project the result back to the shape space.
5. If the current level is the last level, yield the result as the fitting result for current frame; if not, set the result as the old fitting result, then go to Step 3 to search in the next level.

The clamping of non-rigid component is necessary to eliminate invalid face shapes that may lead to completely missing the face shape. In classic ASM, clamping reduces the number of iteration before the result converges. Figure 19 illustrates the result of clamping invalid face shapes.

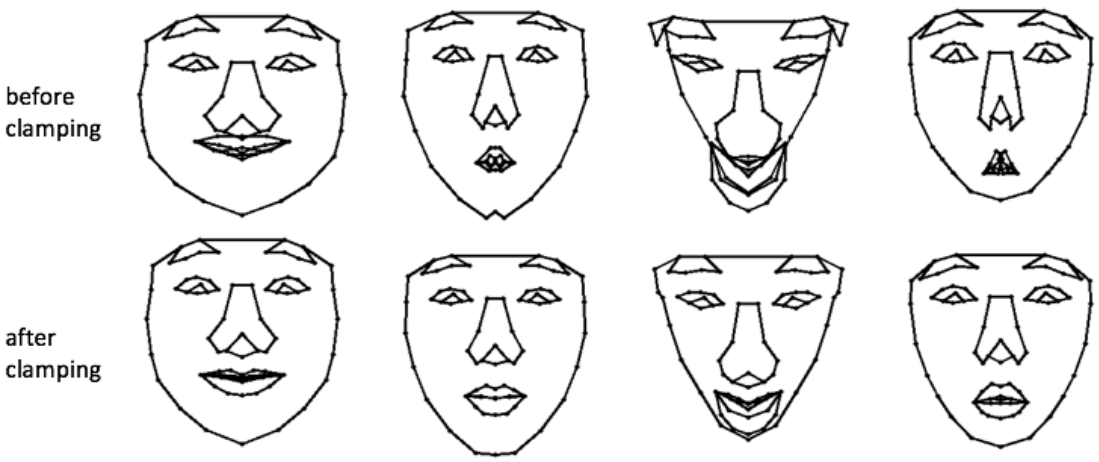

Figure 19 Clamping invalid face shapes

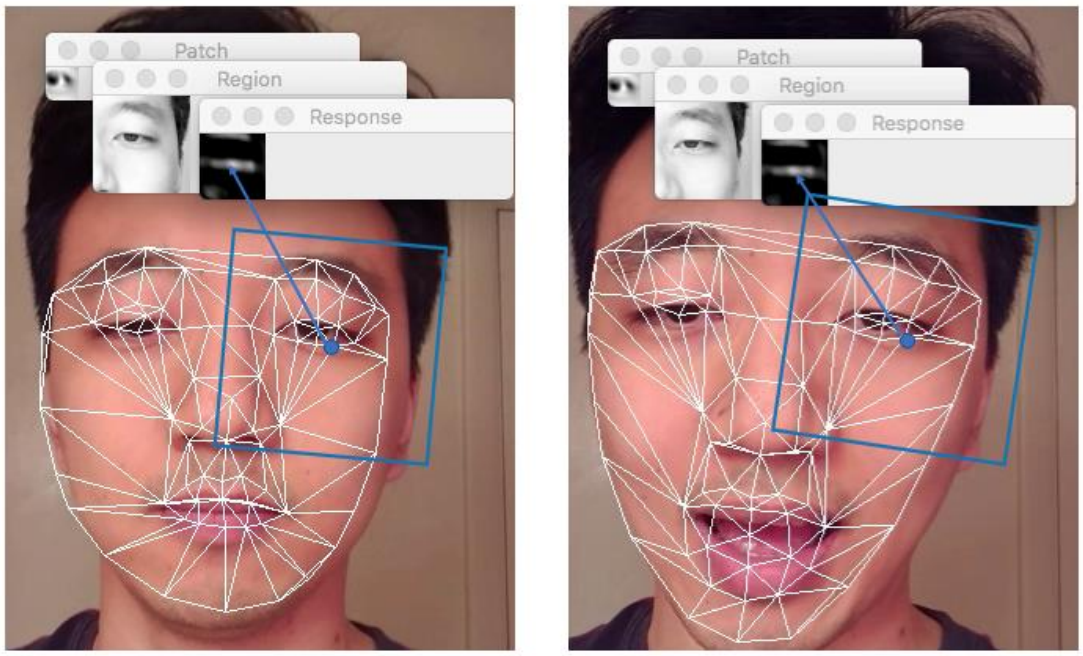

Figure 20 Fitting the shape in real time

Figure 20 shows the fitting of one feature point in the tracking phase. The detected point is very close to the ground truth.

3.5. Experiment Results

Experiments described in this section compare two aspects of the proposed scheme to the original ASM scheme: 1) whether the convergence criterion is necessary for the tracking problem and 2) how much superior template matching of patch model is to calculating correlation of 1D profile model. Lastly, the accuracy of fitting with ASMs trained from generic data set and person-specific data set is compared.

3.5.1. Experiment Setup

For all experiments presented in this section, non-rigid components of face shapes are the eigenvectors whose eigenvalues sums up to 95% of the total sum of eigenvalues. The reference face shape's width is 100 pixels.

For the proposed approach, the size of patch model is 20 by 20 pixels; the size of search window for patch model training is 20 by 20 pixels; the three levels of search window for fitting is 20 by 20 pixels, 10 by 10 pixels and 5 by 5 pixels.

For the baseline approach using 1D profile model, the length of each profile is 20 pixels; the range to shift the profile is 20 pixels for training and 20, 10, 5 pixels for fitting in three levels. This approach uses the same method as other approaches with patch model to calculate the correlation between samples and profile/patch model. The correlation for a point (x,y) in the search window/ range is as follows.

$$R(x,y) = \frac{\sum_{x',y'}[T(x',y') \cdot I(x + x', y + y')]}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x + x', y + y')^2 \cdot}}$$

where T is the result of subtracting the average intensity of the patch/profile from each of its pixels and I is the result of subtracting the average intensity of the sample (region of interest or sampled profile) from each of its pixels.

For convergence-based approach, the ending criterion is the residual sum of squares (RSS) of the shape is smaller than or equal to 0.001. The search also ends if there have been 20 iterations.

The video input is captured at 30 frames per second with a resolution of 404 by 720.

3.5.2. Frame Rate

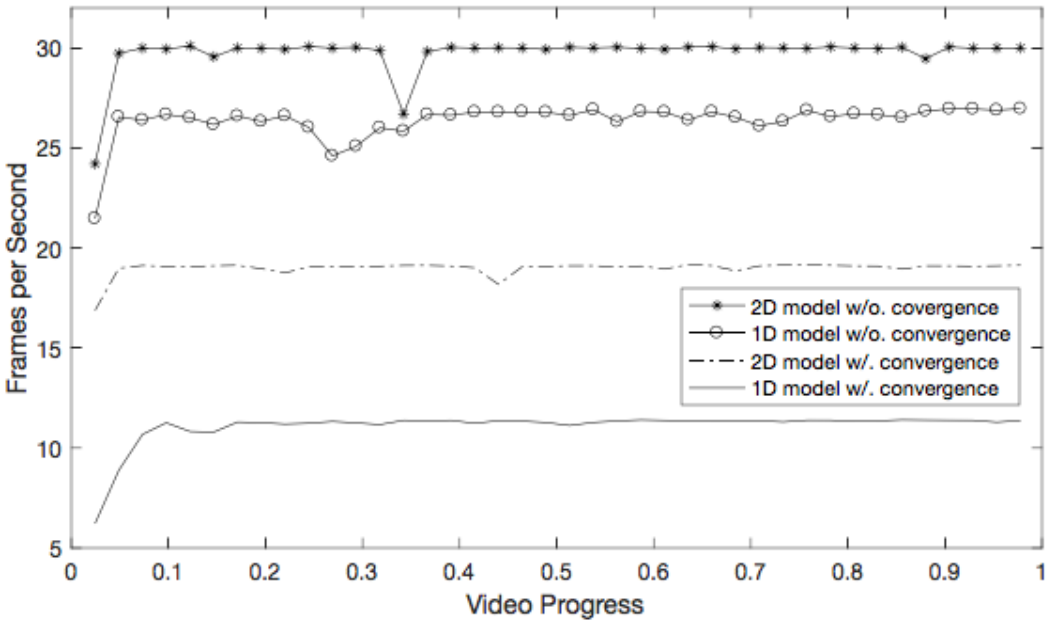

Figure 21 The frame rates of output using different approaches

Figure 21 compares the frame rate of output using four approaches. The frame rate is calculated at intervals that are equivalent to 30 frames in the input video, which is 1 second's worth of input video. The baseline approach is convergence-based and uses the

1D profile model, as implemented in the original ASM. Higher frame rate suggests less computation expenses.

On each curve, the starting point also exhibits the lowest frame rate, because Haar filter is implemented to detect the face region in the first frame, which is compute-intensive. Frame rate here is sampled in intervals of 30 frames, hence, it implies that detecting face region using Haar filter is drastically more intensive than the computation spent on the rest of the video.

It is noticeable that under the same condition, approaches using the 1D profile model have lower frame rates than approaches using the patch model. The expense of calculating covariance for patches is significantly greater than its counterpart for 1D profiles. However, the result suggests that to determine the profile normal directions frame by frame is time consuming, because profile normals are sensitive to shape changes and they should be updated over time. Whereas, the search window of the patch model is an axis-aligned square, which is independent of face shapes.

It can be observed that in convergence-based approaches, the approach using patch model runs much faster than that using profile model. This implies that the result of shape fitting using the patch model converges faster than using the profile model. Fitting using patch model is more efficient because its resultant points moves in a 2D region in each iteration, while those of the profile model approach can only move in a 1D range, which requires constantly changing of direction that result in more iterations.

3.5.3. Fitting Accuracy

The proposed approach uses person-specific data set to train the shape model and 2D patch model. It uses three levels of search windows for fitting in each frame of video, but does not require convergence. The fitting accuracy is analyzed by comparing it to alternative approaches.

Each of the alternative approaches is only different from the proposed approach in one aspect. A collection of frames captured from the experiment is shown in Figure 22. The first column of images is the output of the approach that replaces the patch model with profile model. The second column is from the approach that uses shape model and patch model trained with a generic training set. The third column is captured using the proposed approach. The fourth column is captured using the approach that has the convergence criteria.

It can be observed that the fourth column has the best output, which is slightly more accurate than the result from the proposed approach in terms of the shape of face outline, and the alignment of eye shapes to the input. However, the result from the previous section suggests that its frame rate is only about 65% of the proposed approach.

The alternative approach using models trained from generic data set provides quite good result, except for the mismatch for the looking-up pose, which is presumably absent from the generic data set. In addition, the fitting of eyes, eyebrows, and face outline is not so good as in the proposed approach for the same reason.

Figure 22 Illustration of accuracy of different approaches

The approach exploiting 1D profile model provides the worst result, which implies that quite a few feature points are not properly fitted. In some frames, right eye, right eyebrow and right half of face outline are severely mismatched. The result suggests that 1D profile

model approach needs convergence criteria to produce robust results, however, as discussed in the previous section, it would be too expensive. Hence, the profile model is not a sensible choice in this context.

3.6. Summary

In this chapter, the proposed approach is introduced, validated and compared with alternative approaches. Concepts from classic ASM are amended and supplemented in the context of created animated face from video stream to contribute to the proposed approach. Experiment results suggest that it is a good tradeoff between efficiency and accuracy. It provides a robust result of face feature points tracking that can be converted to a polygon mesh to be rendered as a 3D face.

Chapter 4: Rendering 3D faces

2D Polygon meshes can be abstracted from video stream using ASM as discussed in Chapter 3. Given the texture information, a face can be rendered. Although the 3D mesh of a face cannot be reconstructed from video stream, approximate vertex normals can provide correct shading on the rendered face and make it look like 3D. The key information to calculate shading is the normal direction of every vertex on the polygon mesh. An approach is presented in this chapter to estimation the vertex normals given the shape fitting result as discussed in Chapter 3 and a reference 3D head model.

4.1. Problem Description

Conventionally, rendering a 3D model requires all its geometric information, including the 3D coordinates of all vertices, the normal vectors of vertices, a list that indicates the vertices of each facet and the texture of each vertex. From the implementation in Chapter 3, we can only get vertex coordinates of a face in 2D and the list of facets. However, as the objective of this project is to render a frontal face, it is still practical to draw the 3D face by estimating vertex normals.

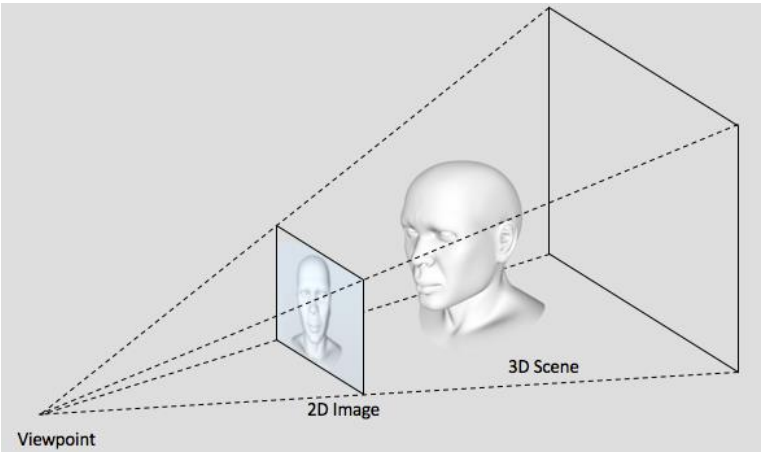

Figure 23 Viewing frustum and image plane

The perspective view frustum of the virtual camera for 3D computer graphics is illustrated in Figure 23. After all phases in the model-view-projection transformation, as illustrated in Figure 24, the image finally drawn to the viewport is the projection of the visible portion of the 3D object on a 2D plane, normally referred to as the image plane. Drawing to the image plane is analog to how an RGB camera works.

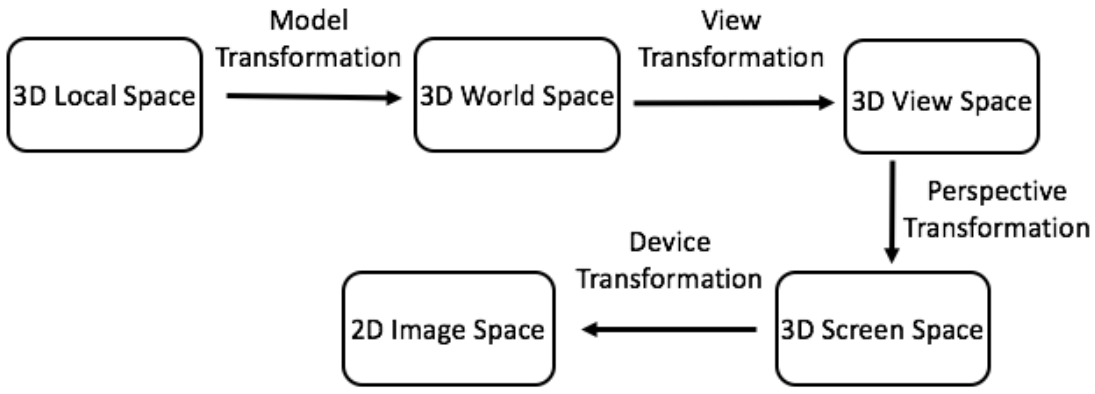

Figure 24 3D transformations in rendering pipeline

If the virtual camera's setup is identical to the RGB camera's setup, drawing the triangulation of face feature points tracked from video stream is equivalent to sending the geometric model of the face through the rendering pipeline to display in an image. Mapping texture to vertices can be manually done, so different texture images can be applied to the same tracked face. However, we need to estimate vertex normals in 3D to provide realistic shading on the face under all lighting conditions in the 3D scene.

It is impossible to calculate vertex normals of feature points from the input video stream, and the normal for a vertex varies from frame to frame in its local coordinate system, because face is a non-rigid body. To simplify the problem, the face is assumed as a rigid body, and its vertex normals in the local coordinate system are measured from a reference 3D head model, then its vertex normals in the scene can be calculated if the pose of head

can be estimated from the input video stream. In the rest of this chapter, an approach to head pose estimation is proposed based on the POSIT algorithm, and the rendering results are presented.

4.2. Validation of POSIT

As discussed in Chapter 2, the focal length of the RGB camera is a required constant for head pose estimation, and it should be measured in pixels. As this parameter is not advertised for consumer cameras, one way to measure the camera's intrinsic matrix is by performing calibration with a series of images of a checkerboard taken with the camera, as shown in Figure 25.

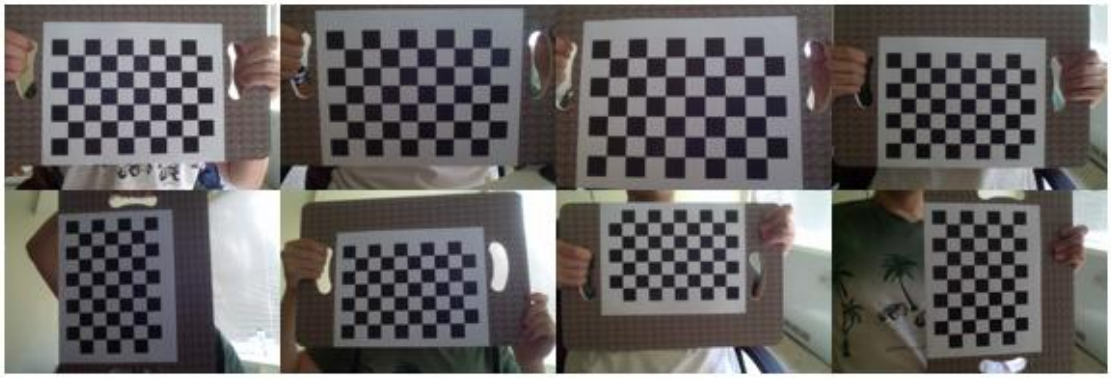

Figure 25 Images for camera calibration

A validation of POSIT in shown in Figure 26, the dimensions of the box in the top-left image is known and four of its visible vertices are annotated as feature points; the bottom-left image shows the estimated pose of the box in 3D, where the feature points are marked with crosses. The right image is overlaying the estimation over the original image which suggests the pose estimation is accurate enough.

Hence, POSIT can be used for head pose estimation if four non-coplanar feature points are provided. The estimated rotation is applied to transform vertex normals from the model coordinate system to the camera coordinate system, to create proper shading on the 3D face.

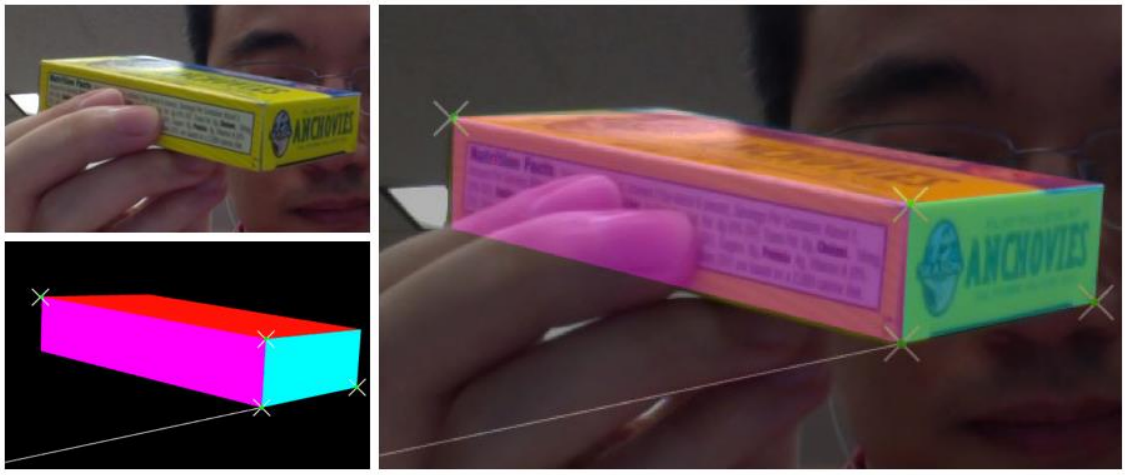

Figure 26 Validation of POSIT

4.3. Implementation

As pointed out in Section 4.1, the first step is to borrow a collection of unit vectors to represent the vertex normals of the tracked feature points in the face model's local coordinate system. Because people's faces vary significantly but their vertex normals of the same vertex are generally in the same direction, a typical 3D head model[1] is selected to provide the vertex normals in the local coordinate system. Although the tracked face is deformable, from the vertex normal's perspective, the face is rigid, and the vertex normals does not change with respect to the local coordinate system. The front view of

[1] Samuel, the 3D head model (Credit: Leah Apanowicz / CGPhoenix). Available under a permissive license.

the reference 3D head model is illustrated Figure 27. The normal vectors are measured in Maya®, a CAD modeling software.

As there are no rigid areas on a human face, the four non-coplanar feature points should be selected carefully. Based on observation, nose tip point, nose bottom point, and another two feature points denoting the width of the bridge of nose, as annotated with red dots in Figure 27, can safely maintain their relative distance from each other in most facial expressions. They are chosen as the input for POSIT, and their local coordinates are also measured in Maya®.

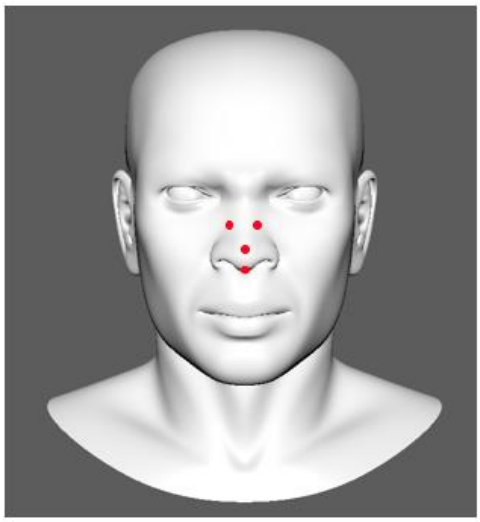

Figure 27 The reference 3D head model

The triangulation of feature points is illustrated in the Appendix. It is obtained by performing Delaunay Triangulation [31] on a typical tracked face shape, then recording the list of connections. This can be regarded as a polygon mesh, hence a PNG file of frontal face is provided as the texture to render the face, where the mapping from the coordinates in the PNG image to feature points is marked manually. A composite face is chosen as the texture to be mapped to the tracked polygon mesh in the experiment.

4.4. Experiment Results

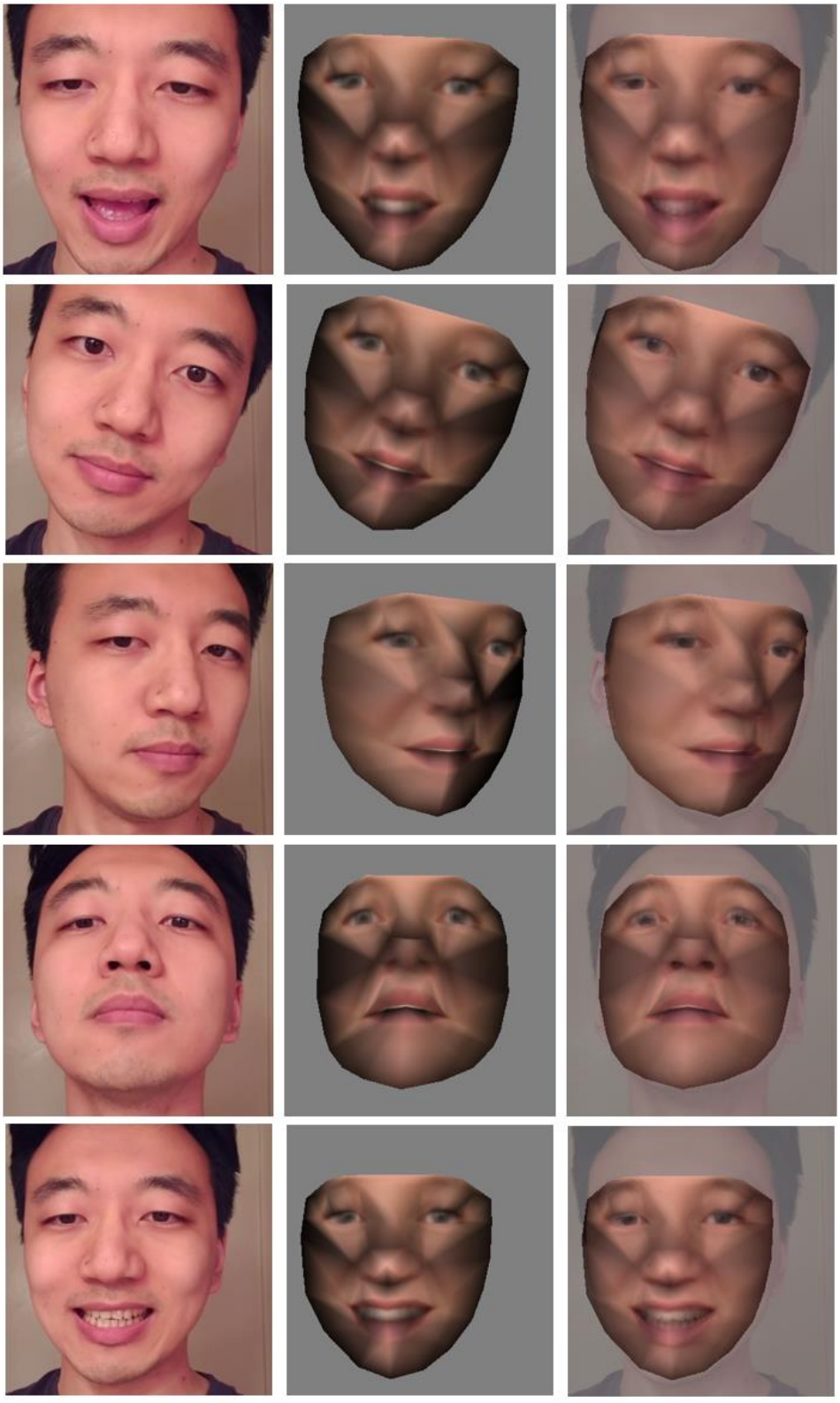

Figure 28 Result of real time rendering

Figure 28 shows the screen capture of the input frames, output frames and the input overlaid on the output. In the 3D scene where the output is generated, the virtual lighting conditions for all the output captures are identical. Phong's Illumination model and Gouraud shading method are implemented to generate the shading on the generated face.

As expected, the output does not resemble the input texture, because of the mismatch between the composite face's shape and the captured polygon mesh as well as the limited texture information in the composite face, which is captured in only one direction. As the Gouraud shading method merely interpolates vertex intensities for the triangular facets and the tracked polygon mesh is sparse, the low intensities of vertices in shades propagate to their neighbor, which is not ideal. However, this phenomenon confirms that the vertex normals obtained from the 3D head model are reasonably close to the ground truth, because the intensities of the most and least lit vertices are exaggerated and coincide human intuition. It can also be observed from Figure 28 that the head motions in the input stream cause correct change of shading on the generated face, which suggests that the POSIT algorithm provides accurate estimation that result in appropriate rotation of vertex normals.

By overlaying the generated face on the input face, it can be concluded that the eye expressions are generally correct but not accurate enough, which is caused by the limited number of subjects for eye expressions in the training set, because in most of the training images, the subject keeps his eyes open and looks at the camera, which leads to limited coverage of eye expressions in ASM. This artifact may be mitigated by introducing more images of the subject with his eyes closed and eyes not focused on the camera.

In addition, the interior of the rendered mouth is only realistic when the input face is showing the teeth, because the composite face in the texture image is showing the teeth. Lastly, there is a slight mismatch between the outlines of the input face and the generated face. This is not ideal for face-swapping applications and should be considered as part of the future work.

It is worth pointing out that the frame rate of the output 3D face animation matches the camera input in real time. If video clips are the input of this system, the frame rate can be even faster than the input frame rate, because of the system's high computing efficiency.

4.5. Summary

In this chapter, the proposed approach to render the tracked face shape in introduced. This approach is based on POSIT, an iterative method to estimate the pose of a rigid body. The estimated head pose is used to transform the vertex normal vectors to the camera coordinate system, which leads to proper shading on the 3D face. Experiment results suggest that this approach generates correct shading on the face, and is responsive to head motions and facial expressions.

Chapter 5: Conclusion and Future Work

5.1. Conclusion

This thesis introduces an efficient solution to reconstruct 3D frontal faces using video stream captured with RGB cameras, e.g. a web cam. The method is based ASM and POSIT algorithm. ASM requires learning from a training set of face images to get the major non-rigid components of a 2D face as well as a collection of image patches to detect face feature points from each frame to fit the shape models. The proposed approach that makes ASM suitable for fitting face shapes in video stream allows face tracking and rendering to be very efficient and can be conducted on-the-fly while the video is being captured.

One factor of tracking accuracy is the probability of composing a face shape that resemble the face in the input video. The experiment comparing the efficiency of using generic shape models and person-specific shape models in the face tracking system, proves that resemblance of the shape models to the input face have influence on both learning and tracking efficiency and tracking accuracy.

Built on top of reliable tracking result, POSIT is implemented to estimate head post of the input face and provide realistic shading on the rendered face. Experiment results suggest that this combination is responsive and accurate for shading.

However, the proposed approach has constraint on the lighting condition in the video stream. The implementation is robust only under lighting condition similar to the training set. In addition, due to the limited amount of facial expressions presented in the training set, some tracked feature points cannot be projected to the shape of the real expression

captured in the video. Due to these reasons, the rendered face may not align to the real face precisely.

In conclusion, the approach introduced in this thesis provides fair tracking and rendering result, given it only requires a collection of annotated images to train shape and patch models, and a RGB camera to capture video streams.

5.2. Future Work

For now, Gauroud shading is implemented to render the 3D face, the shades on it look discontinuous, because the tracked face shape can only provide a sparse polygon mesh. To mitigate this artifact, Phong shading may be implemented instead, because it interpolates vertex normals rather than vertex intensities, which should result in smoother result. The other solution is to replace the RGB camera with a RGBD camera. The depth sensor of RGBD camera generates a point cloud on the surface of the face. The point cloud makes it possible to generate not only dense polygon mesh, but also the accurate vertex normals, both of which will significantly improve the shading result [32].

To increase the accuracy of fitting, it may be effective to do PCA in training phase on a multidimensional space that represents both face shapes and face texture as in the active appearance model (AAM) method. Hence, in tracking phase the correlation between both shapes and textures are checked, which may increase the probability of accurate matches.

In the current approach, it is possible that template matching provides accurate result for a subset of feature points, but after projecting the shape to the principal component space then projecting it back to shape space, the fitting result of the subset became less accurate despite the improvement of fitting overall. Hence, it may also be sensible to train the

shape model or appearance model of each facial features separately and use a hybrid of local shape models and the overall shape model for better fitting results.

Currently, only the face region appears in the output. It is possible to estimate the dynamics in the input and add the effect to the output animation, so that the rendered face can have some detailed features which are sensitive to motions, to make the animation more realistic.

References

[1] Dollár, P., Welinder, P., & Perona, P. (2010, June). Cascaded pose regression. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (pp. 1078-1085). IEEE.

[2] Saragih, J. M., Lucey, S., & Cohn, J. F. (2011). Deformable model fitting by regularized landmark mean-shift. *International Journal of Computer Vision*, *91*(2), (pp. 200-215).

[3] Xiong, X., & De la Torre, F. (2013). Supervised descent method and its applications to face alignment. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 532-539).

[4] Jones, M., & Viola, P. (2003). Fast multi-view face detection. *Mitsubishi Electric Research Lab TR-20003-96*, *3*, 14.

[5] Cootes, T. F., Edwards, G. J., & Taylor, C. J. (2001). Active appearance models. *IEEE Transactions on pattern analysis and machine intelligence*, *23*(6), 681-685.

[6] Cootes, T. F., & Taylor, C. J. (1999). A mixture model for representing shape variation. *Image and Vision Computing*, *17*(8), 567-573.

[7] Zhang, S., Zhan, Y., Dewan, M., Huang, J., Metaxas, D. N., & Zhou, X. S. (2011, June). Sparse shape composition: A new framework for shape prior modeling. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on* (pp. 1025-1032). IEEE.

[8] Gu, L., & Kanade, T. (2006, June). 3D alignment of face in a single image. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on* (Vol. 1, pp. 1305-1312). IEEE.

[9] Vogler, C., Li, Z., Kanaujia, A., Goldenstein, S., & Metaxas, D. (2007, October). The best of both worlds: Combining 3d deformable models with active shape models. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on* (pp. 1-7). IEEE.

[10] Cristinacce, D., & Cootes, T. F. (2007, September). Boosted regression active shape models. In BMVC (Vol. 2, pp. 880-889).

[11] Behaine, C. A., & Scharcanski, J. (2012). Enhancing the performance of active shape models in face recognition applications. IEEE Transactions on Instrumentation and Measurement, 61(8), 2330-2333.

[12] Lee, Y. H., Kim, Y., Kim, H. J., Shin, I. K., Ahn, H., & Lee, Y. (2016, July). Modified Active Shape Model for Realtime Facial Feature Tracking on iPhone. In Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2016 10th International Conference on (pp. 419-421). IEEE.

[13] Dong, S., & Luo, S. (2006, August). Modified grey-level models for active shape model training. In Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE (pp. 3791-3794). IEEE.

[14] Cao, C., Weng, Y., Lin, S., & Zhou, K. (2013). 3D shape regression for real-time facial animation. *ACM Transactions on Graphics (TOG)*, *32*(4), 41.

[15] Huber, P., Feng, Z. H., Christmas, W., Kittler, J., & Ratsch, M. (2015, September). Fitting 3d morphable face models using local features. In *Image Processing (ICIP), 2015 IEEE International Conference on* (pp. 1195-1199). IEEE.

[16] Moreno-Noguer, F., Lepetit, V., & Fua, P. (2007, October). Accurate non-iterative o (n) solution to the pnp problem. In *Computer vision, 2007. ICCV 2007. IEEE 11th international conference on* (pp. 1-8). IEEE.

[17] Garro, V., Crosilla, F., & Fusiello, A. (2012, October). Solving the pnp problem with anisotropic orthogonal procrustes analysis. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on* (pp. 262-269). IEEE.

[18] Li, S., Xu, C., & Xie, M. (2012). A robust O (n) solution to the perspective-n-point problem. *IEEE transactions on pattern analysis and machine intelligence*, *34*(7), 1444-1450.

[19] Mbouna, R. O., Kong, S. G., & Chun, M. G. (2013). Visual analysis of eye state and head pose for driver alertness monitoring. *IEEE transactions on intelligent transportation systems*, *14*(3), 1462-1469.

[20] M. La Cascia, S. Sclaro, and V. Athitsos, "Fast, reliable head tracking under varying illumination: An approach based on registration of texturemapped 3D models," IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, no. 4, pp. 322–336, Apr. 2000.

[21] Xiong, X., & De la Torre, F. (2015). Global supervised descent method. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2664-2673).

[22] Dinc, S., Fahimi, F., & Aygun, R. (2017). Mirage: an O (n) time analytical solution to 3D camera pose estimation with multi-camera support. *Robotica*, 1-19.

[23] Gower, J. C., & Dijksterhuis, G. B. (2004). *Procrustes problems* (Vol. 30). Oxford University Press on Demand.

[24] Abdi, H., & Williams, L. J. (2010). Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, *2*(4), 433-459.

[25] Dementhon, D. F., & Davis, L. S. (1995). Model-based object pose in 25 lines of code. *International journal of computer vision*, *15*(1), 123-141.

[26] Huttenlocher, D. P., & Ullman, S. (1990). Recognizing solid objects by alignment with an image. *International Journal of Computer Vision*, *5*(2), 195-212.

[27] Milborrow, S., Morkel, J., & Nicolls, F. (2010). The MUCT landmarked face database. *Pattern Recognition Association of South Africa*, *201*(0).

[28] Brunelli, R. (2009). *Template matching techniques in computer vision: theory and practice*. John Wiley & Sons.

[29] Kasinski, A., & Schmidt, A. (2010). The architecture and performance of the face and eyes detection system based on the Haar cascade classifiers. *Pattern Analysis and Applications*, *13*(2), 197-211.

[30] Shan, C., Gong, S., & McOwan, P. W. (2009). Facial expression recognition based on local binary patterns: A comprehensive study. *Image and Vision Computing*, *27*(6), 803-816.

[31] Shewchuk, J. R. (2002). Delaunay refinement algorithms for triangular mesh generation. *Computational geometry*, *22*(1-3), 21-74.

[32] Li, H., Yu, J., Ye, Y., & Bregler, C. (2013). Realtime facial animation with on-the-fly correctives. *ACM Trans. Graph.*, *32*(4), 42-1.

[33] De Maesschalck, R., Jouan-Rimbaud, D., & Massart, D. L. (2000). The mahalanobis distance. Chemometrics and intelligent laboratory systems, 50(1), 1-18.

Appendix

A. Face Annotation

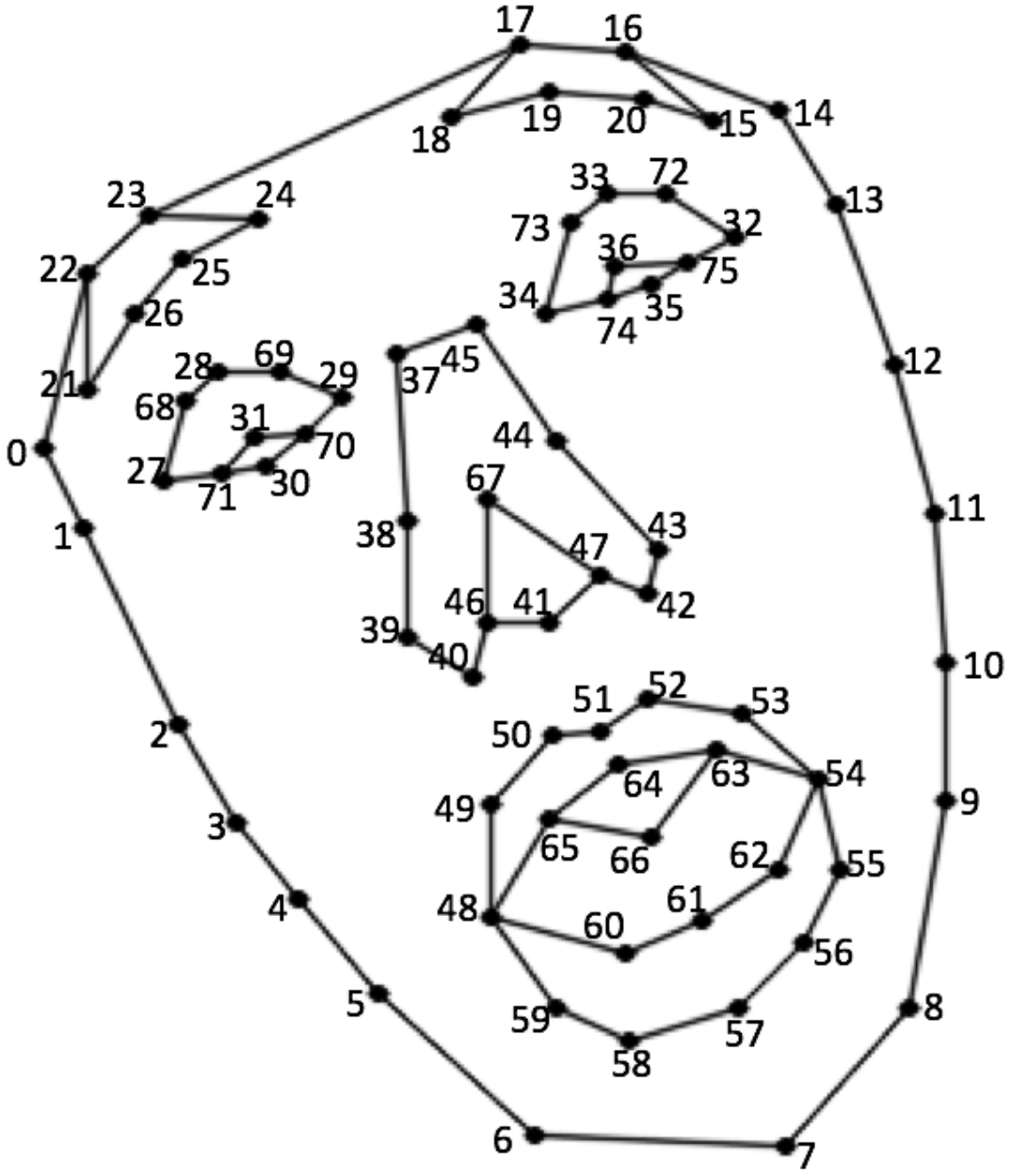

Figure 29 Face annotation

Vertex ID	Description	Vertex ID	Description
0	left outline starting from top of face to bottom	41 *	columella of nose
1		42	right nose outline starting from bottom of face to top
2		43	
3		44 *	
4		45	
5		46	left nostril
6		47	right nostril
7	chin-middle	48	outline of mouth starting from left rear point in clockwise direction
8	right outline starting from bottom of face to top	49	
9		50	
10		51	
11		52	
12		53	
13		54	
14		55	
15	right eyebrow starting from rear point in counter-clockwise direction	56	
16		57	
17		58	
18		59	
19		60	the interior of mouth starting from the bottom-left point in counter-clockwise direction
20		61	
21	left eyebrow from starting rear point in clockwise direction	62	
22		63	
23		64	
24		65	
25		66	center of mouth
26		67 *	nose tip
27	left eye four corners starting from rear point in clockwise direction	68	left eye mid points starting from the top-left one in clockwise direction
28		69	
29		70	
30		71	
31	left pupil	72	right eye mid points starting from the top-right one in counter-clockwise direction
32	right eye four corners starting from rear point in counter-clockwise direction	73	
33		74	
34		75	
35		vertices marked with asterisks are used for head pose estimation.	
36	right pupil		
37	left nose outline starting from top of face to bottom		
38 *			
39			
40			

Table 1 Definition of annotated feature points

B. Face Triangulation

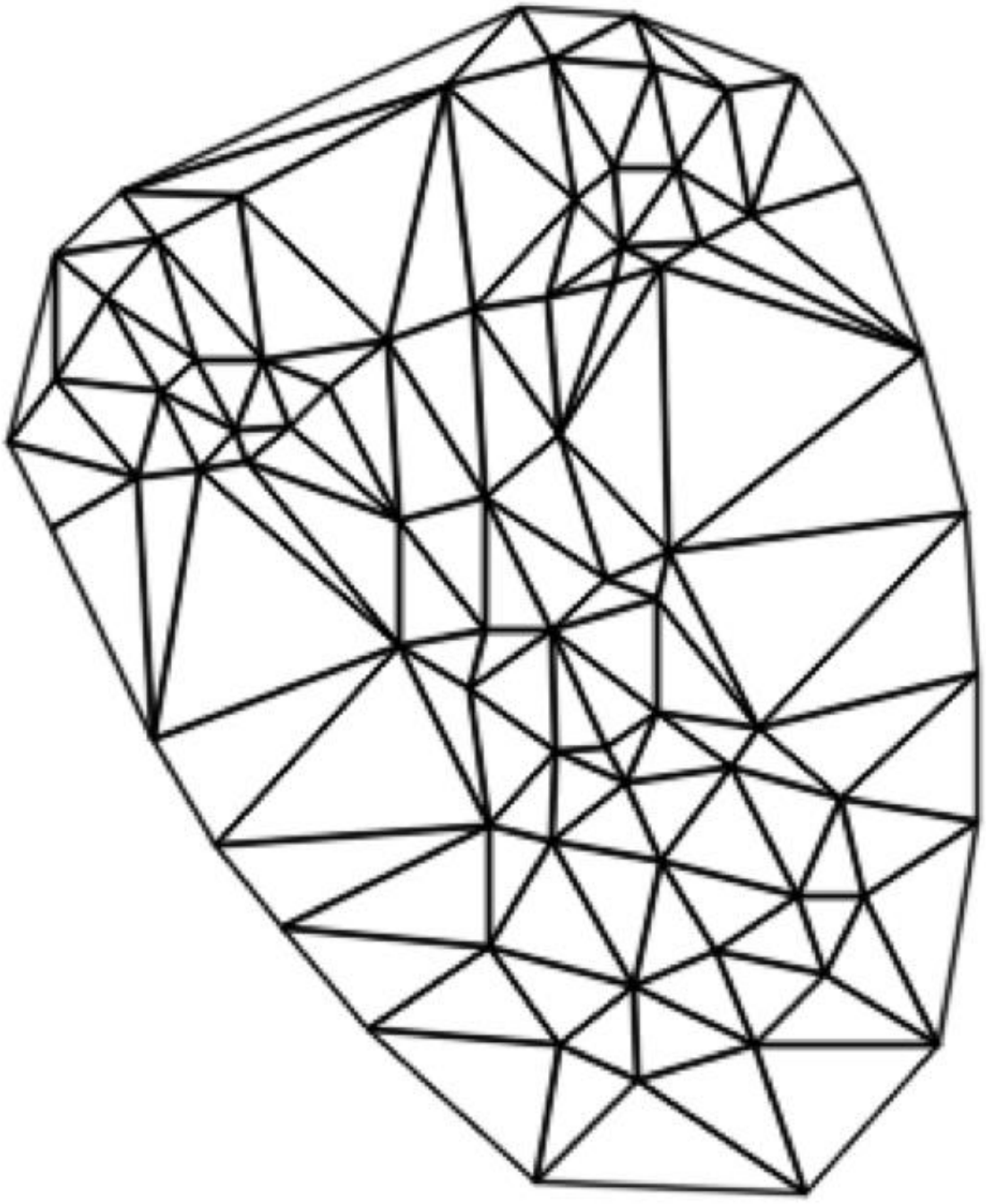

Figure 30 Face triangulation

Face	V1	V2	V3	Face	V1	V2	V3	Face	V1	V2	V3	Face	V1	V2	V3
0	20	17	16	35	14	32	15	70	31	28	69	105	48	49	65
1	17	20	19	36	16	14	15	71	70	31	69	106	65	66	60
2	2	0	1	37	13	75	32	72	19	33	73	107	66	65	64
3	6	58	57	38	75	13	43	73	33	20	72	108	49	50	65
4	58	6	5	39	16	15	20	74	36	72	75	109	51	41	52
5	1	0	27	40	27	0	21	75	72	36	33	110	56	62	55
6	2	48	3	41	25	23	24	76	18	34	45	111	50	51	64
7	48	2	39	42	23	25	26	77	34	18	73	112	54	62	63
8	2	1	39	43	24	29	69	78	37	38	29	113	62	54	55
9	38	70	29	44	29	24	37	79	38	37	44	114	51	52	64
10	70	38	39	45	17	18	23	80	44	67	38	115	52	42	53
11	63	64	52	46	72	15	32	81	67	44	47	116	52	53	63
12	64	63	66	47	15	72	20	82	74	43	34	117	10	55	54
13	4	3	48	48	18	19	73	83	43	74	35	118	53	54	63
14	4	48	59	49	19	20	33	84	75	43	35	119	66	63	62
15	5	4	59	50	27	26	68	85	43	12	54	120	58	61	57
16	71	31	30	51	26	27	21	86	36	75	35	121	61	58	60
17	31	71	68	52	21	0	22	87	36	35	74	122	60	59	48
18	47	43	42	53	23	18	24	88	44	45	34	123	59	60	58
19	43	47	44	54	21	22	26	89	45	44	37	124	58	5	59
20	7	6	57	55	22	23	26	90	50	41	51	125	60	48	65
21	10	8	9	56	30	39	71	91	41	50	40	126	64	65	50
22	8	7	56	57	39	30	70	92	48	39	40	127	61	60	66
23	9	8	56	58	25	24	69	93	61	56	57	128	62	61	66
24	9	56	55	59	26	25	28	94	56	61	62	129	27	68	71
25	10	9	55	60	1	27	39	95	56	7	57	130	69	29	70
26	24	45	37	61	26	28	68	96	41	40	46	131	72	32	75
27	45	24	18	62	39	27	71	97	42	54	53	132	34	73	74
28	11	10	54	63	31	68	28	98	54	42	43				
29	36	73	33	64	28	25	69	99	41	42	52				
30	73	36	74	65	67	41	46	100	42	41	47				
31	12	11	54	66	41	67	47	101	34	43	44				
32	17	19	18	67	46	39	38	102	46	38	67				
33	13	12	43	68	39	46	40	103	48	40	49				
34	32	14	13	69	70	30	31	104	49	40	50				

Table 2 The triangulation of a tracked face