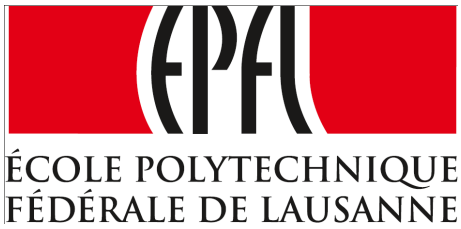
Approximation of Face Images by Analytical Models



Yu Zhan
Assistant: Elif Vural
Director: Prof. Dr. Pascal Frossard
Signal Processing Laboratory 4(LTS4)
École polytechnique fédérale de Lausanne (EPFL)
February, 2012

Abstract

In this thesis, we address the problem of learning a face manifold model from an input data, consisting of 2D face images such that the input images can be approximated well using the learned manifold. We compute the face manifold in terms of a depth pattern and a texture pattern, which define a face model together. Given the depth pattern and the texture pattern of the individual face, the images of this face from different viewpoints can be rendered by changing the camera parameters. Along with the depth and texture patterns, we compute a parameter vector for each input image, which gives the registration of the image with respect to the computed manifold. Then, the approximation of each input image is given by its projection on the manifold, i.e., the image rendered from the model with the parameter vector corresponding to the image.

In the computation of the depth pattern and the texture pattern, we represent both patterns in terms of some elementary functions called “atoms”. We have chosen to use parametric and smooth atoms, i.e., atoms derived from smooth generating functions, in the construction of the manifold.

Table of Contents

List of Figures

List of Tables

Acknowledgements

This project was developed between October 2011 and February 2012 at the Signal Processing Institute of the Swiss Federal Institute of Technology in Lausanne, Switzerland.

Firstly, I would like to show gratitude to the director of this thesis, Professor Pascal Frossard, for his help and for giving me the great opportunity to work in Signal Processing Laboratory 4 (LTS4). I also give my best thanks to my assistant Elif Vural, for her patience during the development of the project, for her smart advices and for sharing with me her knowledge.

I would especially give thanks to my parents, for their unconditional support of my stay in Lausanne.

Finally, I would like to give thanks to all the people that I have met in Lausanne, who made this experience unforgettable.

Lausanne, 29 February 2011

Chapter 1. INTRODUCTION

Automatic recognition of human faces is one of the most challenging topics in the field of image processing and computer vision. The difficulty is due to the strong dependence of the recognition accuracy on the features that are extracted to represent the face pattern and on the classification method used to distinguish between faces. There is a considerable progress in the area of two-dimensional (2D) face recognition where intensity/color images of human images are employed, for instance, the Eigenface purposed by Matthew A. et al. and also the Fisherface approach summited by Peter N.Belhumeur et al. However, the 2D based systems are sensitive to illumination, pose variations, and facial expressions. In addition, storage problems appear when the number of individuals in the database becomes large. On the other hand, 3D based methods such as 3D face recognition with sparse spherical representations have the potential for greater recognition accuracy and are capable of overcoming these limitations. However, besides the same storage problem, they are computationally expensive and 3D face data are usually not easily collectible, because 3D facial models have to be obtained using active devices.

The face recognition problem can be analyzed from the viewpoint of face subspaces or manifolds. Generally speaking, the face detection problem then corresponds to the discrimination between face manifolds and non-face manifolds, and the face recognition problem involves the characterization of the face images of each person using an individual manifold. However, these two problems are not trivial. If we look into face manifolds in the image space, we find them highly nonlinear and non-convex. The left image of Figure 1.1 illustrates face versus non-face manifolds, and the right one illustrates the manifolds of two individuals in the entire face manifold. Furthermore, there is also the high dimensionality problem. For example, a tiny gray-scale (0-255 levels) image of size 32x32 has 1024 pixels, this means that we have around 256^{1024} "configurations". However, only a few correspond to a face pattern. Immediately, this leads us to think of some subspace analyses in order to obtain a dimensionality reduction.

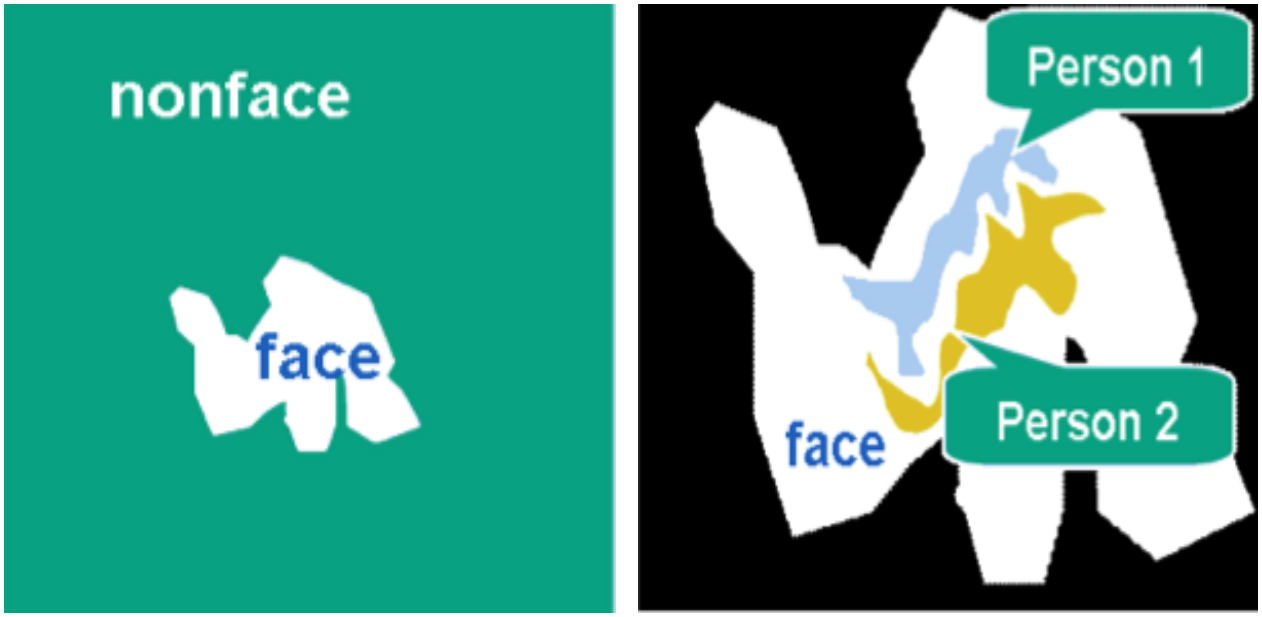


Figure 1.1 Left image: The manifolds of two individuals in the entire face manifold. Right image: Face manifold vs non-face manifold. **(13) (14)**

In this thesis, we address the problem of learning a face manifold model from an input data, consisting of 2D face images such that the input images can be approximated well using the learned manifold. We compute the face manifold in terms of a depth pattern and a texture pattern, which define a face model together. Given the depth pattern and the texture pattern of the individual face, the images of this face from different viewpoints can be rendered by changing the camera parameters. Along with the depth and texture patterns, we compute a parameter vector for each input image, which gives the registration of the image with respect to the computed manifold. Then, the approximation of each input image is given by its projection on the manifold, i.e., the image rendered from the model with the parameter vector corresponding to the image.

In the computation of the depth pattern and the texture pattern, we represent both patterns in terms of some elementary functions called “atoms”. We have chosen to use parametric and smooth atoms, i.e., atoms derived from smooth generating functions, in the construction of the manifold. The usage of smooth and parametric atoms in the model results in a more regular manifold, which facilitates the registration of face images with respect to the manifold, and the ease of efficient storage of the model information.

The studied method for 3D face modeling can be used in pose-invariant face recognition applications, whereas most of the 2D face recognition methods have limited performance in case of pose variation. The recognition process can be assimilated as the minimum manifold distance computation. For instance, a test image is project to the different manifolds obtained from the database, and the image is classified to the one that gives the lowest distance. In the Figure 1.2 is shown the structure.

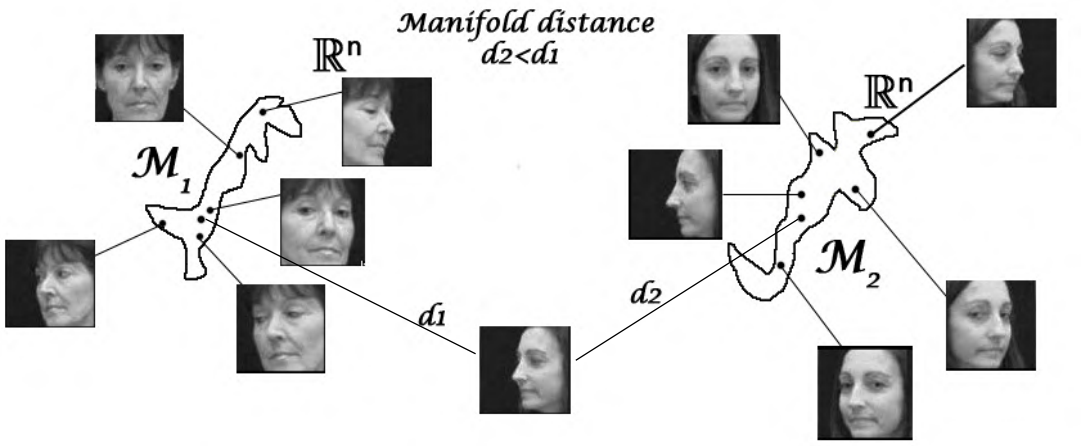


Figure 1.2 If the manifolds of different subjects are provided. Then, the classification process is equivalent to calculate the minimum manifold distance.

1.1. Related Works

1.1.1. Eigenfaces

In (1) Matthew A. et al. present a 2D Face recognition system based on "Eigenfaces". This approach considers statistical features of face data in a new "face space", characterized by eigenvectors, or called also eigenfaces. Mainly, the aim is to find, with Principal Component method (PCA), a set of eigenvectors that can represent adequately the deviations between the images and the average one. Consider a set of N face images $\{I_1, I_2, ..., I_N\}$, defined in an n-dimensional space, where each image belongs to one of c classes $\{f_1, f_2, ..., f_c\}$. The average image is calculated as $I_{av} = \frac{1}{N}\sum_{i=1}^{N} I_i$, and the total covariance as $S_T = \sum_{i=1}^{N}(I_i - I_{av})(I_i - I_{av})^T \in R^{nxn}$. We want to find the unitary matrix $W \in R^{nxm}$ m<n that gives the best approximation rate:

$$Ie_i = W^T I_i \qquad [1.1]$$

in terms of mean square error:

$$e = \sum_{i=1}^{N} \left\| WIe_i - I_i \right\|^2 \qquad [1.2]$$

It has been demonstrated that the optimal matrix is $W_{op} = \arg\max_{W} \left| W^T S_T W \right| = [w_1, w_2, ..., w_m]$, where $[w_1, w_2, ..., w_m]$ are the eigenvectors of the matrix S_T. In this manner, using the largest m eigenvalues and their corresponding eigenvectors, all the face images in the database can be approximated as linear combinations of eigenfaces. Then, for the classification process, we only need to project the test image in the new face space, and compute its distance to projections of the training images in the same space in order to find out the best match.
One drawback of this method is that it maximizes the inter-class distances, but also the intra-class ones, which is undesired for classification purposes. Thus, methods that use PCA will be expected to have a poor classification performance under variation of lighting, and facial expression.

1.1.1. Fisherfaces

Peter N.Belhumeur et. al. proposed another method based on linearly projecting the image space to a low dimensional subspace. They called the developed algorithm “Fisherfaces”, and it is designed to be insensitive to large variation in lighting and facial expression. The algorithm assumes that class labels of the train data are available. It uses a class specific method called Fisher’s Linear Discriminant (FLD) that tries to “shape” the scatter in order to make it more reliable for classification.

In this way, given N face images $\{I_1, I_2, ..., I_N\}$ in an n-dimensional space that belong to one of c classes $\{f_1, f_2, ..., f_c\}$, the objective is to find a transformation matrix $W \in R^{nxm}$, in order to redefine the input samples (using the expression defined in equation [1.1]) that keeps the variation between classes, but reduces the variation of samples in the same class.

Mapping the images defined in an n-dimensional space into an m-dimensional feature space is the task of the transformation. The method selects a matrix W in order to maximize the ratio of the inter-class scatter and the intra-class scatter:

$$S_T = S_B + S_W \quad [1.3]$$

$$S_B = \sum_{i=1}^{c} N_i \left(I_{av_i} - I_{av} \right) \left(I_{av_i} - I_{av} \right)^T \quad [1.4]$$

$$S_w = \sum_{i=1}^{c} \sum_{I_i \in f_i} N_i \left(I_i - I_{av_i} \right) \left(I_i - I_{av_i} \right)^T \quad [1.5]$$

where I_{av_i} is the mean image of the class f_i, and N_i is the number of samples in the class f_i. Then, the optimal transformation matrix is W_{op}, chosen as:

$$W_{op} = \arg\max_W \frac{\left| W^T S_B W \right|}{\left| W^T S_W W \right|} = \left[w_1, w_2, ..., w_m \right] \quad [1.6]$$

The previous formula can be used only if S_w is a nonsingular matrix. In face recognition, the within-class scatter matrix $S_w \in R^{nxn}$ is usually singular, due to the fact that the number of images in the learning set N is much smaller than the number of points in each image I_i. Thus, to overcome this complication, the proposed method, called "Fisherfaces" uses an alternative of the criterion in equation[1.6]. It projects the image set to a lower dimensional space as Eigenface method, using PCA in order to reduce the dimension of the feature space to $N - c$. Then, by applying the defined FLD (equation [1.6]) it reduces once more the dimension to $c - 1$.

$$W_{op}^T = W_{fld}^T W_{pca}^T \quad [1.7]$$

Where:

$$W_{pca} = \arg\max_W \left| W^T S_T W \right| \quad [1.8]$$

$$W_{fld} = \arg\max \frac{\left| W^T W_{pca}^T S_B W_{pca} W \right|}{\left| W^T W_{pca}^T S_W W_{pca} W \right|} \quad [1.9]$$

1.1.2. 3D Face recognition using sparse atoms

R. Sala Llonch, et al. (2) present a method for 3D face recognition with spherical atoms. The method first extracts the face region from the 3D database, and registers all the faces in order to remove the scale and alignment problems. Afterwards, the 3D point cloud of the face region is interpolated to an equiangular spherical grid. The next step is to run simultaneous

sparse Matching Pursuit (see section 2.4) over all the faces, which means to find the best representation of the faces using a set of basis atoms from a redundant dictionary. These atoms are selected such that they can represent salient features like eyes, nose and mouth. Thus, faces are approximately represented by a vector of atom coefficients. With such a scheme, the posterior classification of a test 3D face can be achieved by computing its atom coefficient vector with Matching Pursuit and comparing it with the coefficient vectors of the training images. Then the class label is estimated by identifying the smallest distance between the coefficient vectors.

1.2. Problem Formulation

As we have mentioned in the introduction, we want to represent the manifold with two characteristic patterns, depth and texture. In this section, we are going to introduce the different notations of the following sections and the general framework of the problem.

The input of our algorithm is a set of N 2D gray images $\{I_1, I_2, ..., I_N\}$. And the aim of this project is to compute a representative pattern $p = (p_d, p_t)$ such that its geometric transformations fit the input data. This pattern has the contribution of two components, depth p_d and texture p_t. Due to the variation of the camera viewpoint in the input data, the pattern should have a 3D form, and some projection mechanism from 3D to 2D is needed. We denote the geometric transformations of the pattern as $G_\lambda(p) = G_\lambda(p_d, p_t) \in R^n$, where λ corresponds to the parameters of the transformation. Thus, an estimation of an input image can be written as $I_i = G_{\lambda_i}(p)$, and the total error as:

$$e = \sum_{n=1}^{N} \left| I_i - I_i \right|^2 = \sum_{n=1}^{N} \left| I_i - G_{\lambda_i}(p) \right|^2 \qquad [1.10]$$

The projection mechanism from 3D to 2D is needed in order to get the approximations of the images. Consider an arbitrary point $[x_w, y_w, z_w]$ in the 3D space. Then, its projection (i, j) on the 2D image plane is defined as:

$$[i, j] = T_\lambda([x_w, y_w, z_w]) \qquad [1.11]$$

The intensity of this pixel is shown in the equation

$$I(r, c) = \frac{1}{L} \sum_{i \in I}^{L} A([x_{w_i}, y_{w_i}, z_{w_i}]) \qquad [1.12]$$

I is the set of 3D points, which the projecting point (i, j) belong the square region defined by $(r-1, c-1)$ and (r, c), and operator A give the texture pixels associate to the 3D point $[x_w, y_w, z_w]$.

The problem is that we need the pattern p in order de compute the estimations $\hat{I}_i$, consequently, the first approximation should be obtained from the input images. Afterwards, dimension reduction technics are applied in both texture and depth domain. Particularly, for depth pattern, we use spherical atoms, hence, the initial depth pattern must be expressed in a regular spherical grid. The refinement is the last step in order to reduce the total error.

The general scheme of the algorithm is described in Figure 1.3:

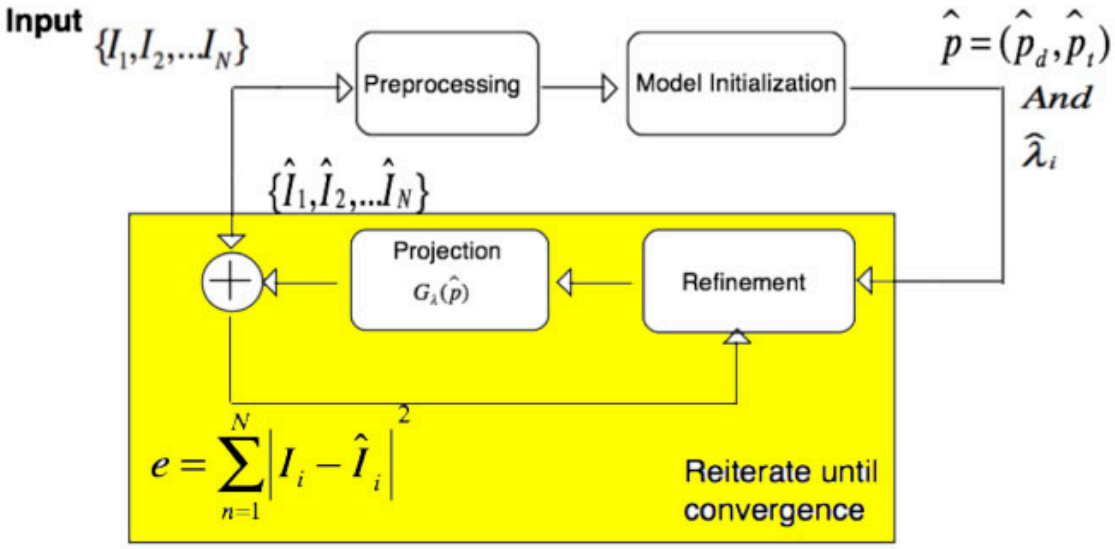


Figure 1.3 General scheme of the algorithm.

1.3. Face Database

The face database used during the entire project is called VidTIMIT database[1] (old version), which contains video recordings of 43 people, reciting short sentences and performing head rotations at the same time. The dataset was recorded in 3 sessions, with an interval of about

[1] http://itee.uq.edu.au/~conrad/vidtimit/

one week between the sessions. There are 10 sentences per person, chosen from the TIMIT corpus.

We are interested only in the head pose variation feature of the database, where the subjects move their head to the left, right, back to the center, up, then down and finally return to center. The recording was done in an office environment using a broadcast-quality digital video camera. The video of each person is stored as a numbered sequence of JPEG gray-scale images with a resolution of 64 x 64 pixels. Some examples of the database are shown in the Figure 1.4.

Ten different poses for each person are picked from the video sequence, and these images are the input for our algorithm to build the face manifold. Some of them are shown in Figure 1.5 and Figure 1.6.

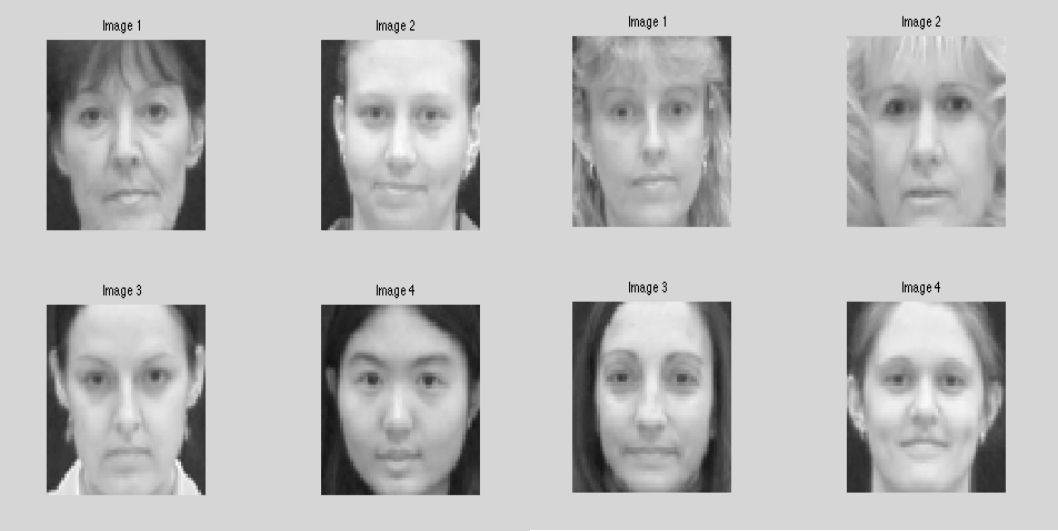


Figure 1.4 Sample images from the VidTIMIT database

Figure 1.5 Examples of inputs to the algorithm

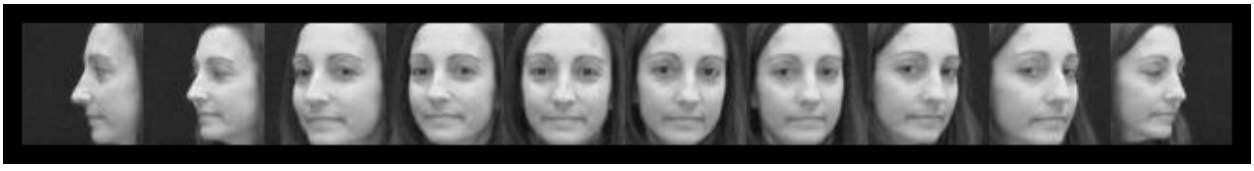


Figure 1.6 Examples of inputs to the algorithm

1.4. Thesis Outline

The thesis has four main chapters. The first one explains the formulation of the problem, reviews of the related works and also the characteristics of the database used in whole project. Chapter 2 contains the model initialization process, where we explain how to build an initial depth and texture pattern. It is divided into four sections. The first section shows the pre-processing steps that have to be performed before the initialization of the model. Then, the second section describes in more detail the 3D to 2D projection mechanism, called Virtual Camera. Finally, the remaining two sections go into detail about how the first depth pattern and texture pattern are obtained, its dimension reduction process via Matching pursuit and the final estimation of registration parameters. The Matching pursuit for depth pattern requires spherical coordinates. Hence, between the initialization of the model and its dimension reduction, some interpolation method is needed. Chapter 3 emphasizes the refinement process of the models achieved in chapter 2. It illustrates the different refinement process and how by alternating it, we can achieve more desirable results. Finally, in the last chapter we present the conclusions and potential future studies.

Chapter 2. MODEL INITIALIZATION

2.1. Preprocessing the input Data

2.1.1. Face Extraction from Background

In the input data, only the face parts of the images are interesting for our purpose. Therefore, some face detection algorithm should be applied as a preprocessing step for our method.

Many algorithms have been proposed for face detection so far. For instance in (3), a face detection algorithm based on skin color detection in the YCbCr[2] color space is presented, where the Cr component is sufficient to realize a proper face separation. Nevertheless, as our database consists only of gray-scale images, a method suitable for gray-scale images needs to be employed. If we observe the input images in more detail, we notice that we can distinguish two main classes in the image: facial regions and non-facial regions (see Figure 2.1). Hence, a classification algorithm can be applied instead of a face detection algorithm for this data. The method chosen for this purpose is the K-means algorithm.

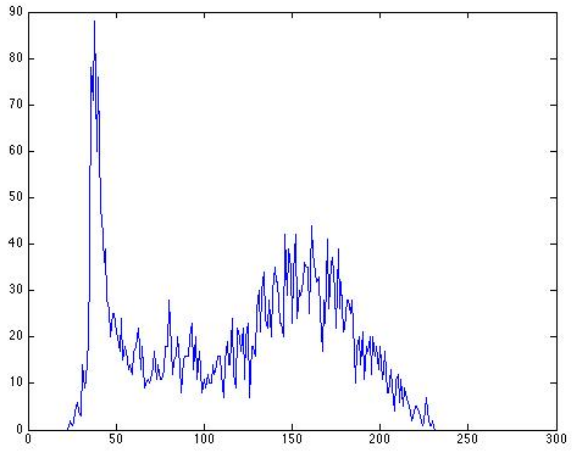


Figure 2.1 Histogram of the image from database. We can perfectly distinguish two classes with Gaussian-like shapes.

[2]Luminance, blue-difference and red-difference chroma.

2.1.1.1. K-means Clustering Algorithm

K-means is a widely-used clustering method based on iterative refinements of the clusters (4) . The initial clusters are set randomly and the number of clusters is defined by user, therefore, is essential to know the number of classes, which is one of drawbacks of this method. Then the clusters are refined by finding the centroid of each cluster and reassignment of the data to the clusters. As the algorithm only computes a local minimum, however has a really low computational complexity, it is common practice to run the algorithm several times with different initializations in order to obtain better clustering results.

The algorithm proceeds by alternating between two steps:

Assignation step: Distribute the points to the nearest mean value, which is set randomly:

$$S_i^{(t)} = \{x_p : \left\| x_p - m_i^{(t)} \right\| \leq \left\| x_p - m_j^{(t)} \right\| \ \forall 1 \leq j \leq k\} \qquad [2.1]$$

Where $S_i^{(t)}$ represents the cluster number i in the instant t, with the set of points x_p, and $m_i^{(t)}$ is the position of the centroid of this cluster.

Update step: Calculate centroids of different clusters, and replace the old means by the centroids.

$$m_i^{(t+1)} = \frac{1}{\left| S_i^{(t)} \right|} \sum_{x_j \in S_i^{(t)}} x_j \qquad [2.2]$$

Where $m_i^{(t+1)}$ is the position of the centroid in the instant $t+1$ of the cluster S_i.

The algorithm terminates when there is no longer change in the distribution of the clusters and the positions of the centroids.

In Figure 2.1 we can see some examples of the classification process. The problem with taking only two classes is that, often, the hair is labeled as a facial region (see Figure 2.1 right image); or some parts of the eye and eyebrow regions are lost, which are important to recover. Then, we complete the whole face extraction process with a post-processing process. We call it ellipse fitting.

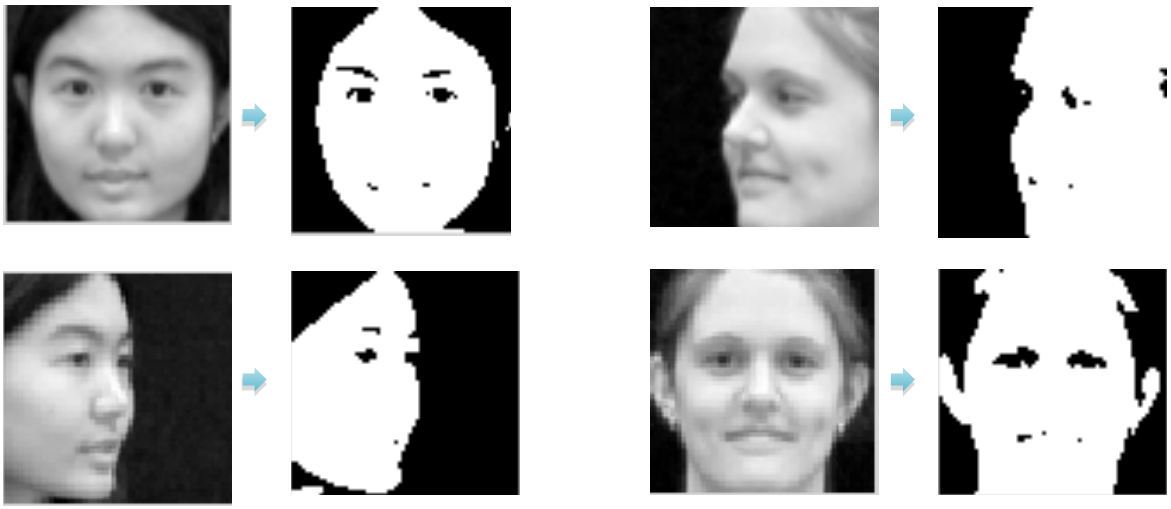


Figure 2.1 Examples from the classification process: In the binary images the white regions correspond to the face component

2.1.1.2. Ellipse Fitting

The idea behind this implementation is that facial regions have ellipse-like shapes. From this point of view, our method tries to fit the best ellipse to the facial region labeled by the k-means algorithm.

Before the application of the ellipse-fitting method, some morphological filters are applied in order to remove small objects in both face and non-face classes. For example, in the facial regions, we fill the holes resulting from the low gray level of the eyes, eyebrows, and the mouth. On the other hand, in non-facial regions, small remaining's of hair or ear are not interesting. Dilation-reconstruction and Erosion-reconstruction filters with square structuring elements are chosen for the purpose. Some results are shown in Figure 2.2

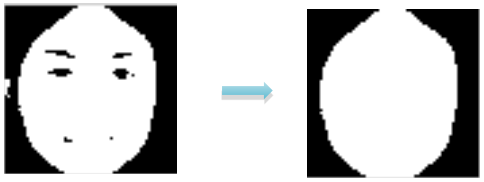


Figure 2.2 Removal of small shapes before ellipse fitting.

The ellipse-fitting algorithm works as follows:

Initialization: First of all, the major and minor axes of the initial ellipse are computed from the positions of first face pixels (white) in vertical and horizontal directions, i.e., the absolute distance between the first row and column where the face pixels begin, and the last row and column where they end. Afterwards, the centers of the ellipse are determined, for instance, horizontal center is computed as the first column of the face pixel plus minor axes over two, and for vertical one first column and major axes are used instead of.

Update: This step is executed only if the number of face pixels inside the ellipse is inferior to 95%. The threshold of 95% was obtained experimentally. When the ratio of face pixels is inferior to this threshold, an ellipse reduction process is performed in order to augment the ratio between face pixels and non-face pixels. We shrink the ellipse progressively by reducing its axis one pixel at each step. This reduction process is terminated when the ratio between the numbers of reduced face pixels and the number of all reduced pixels in the ellipse gets below 70%. For instance, if the initial pixels inside the ellipse are $P_i = P_i^f + P_i^{nf}$, after reduction process we have an ellipse that contains $P_{i+1} = P_{i+1}^f + P_{i+1}^{nf}$ pixels, then, the difference between them is $\Delta P = P_i - P_{i+1} = \Delta P^f + \Delta P^{nf}$. In such manner, $r = \frac{\Delta P^f}{\Delta P} = \frac{7}{10}$ is defined as the lower bound.

In Figure 2.3 the ellipse fitting process is demonstrated, where the difference between the initial ellipse and the updated one can be seen.

Figure 2.3 Ellipse fitting for face extraction

As we can observe in the figure, applying this process, non-face pixels are considerably reduced, but we also lose interesting details such as the extremity of the nose. We will see that the recovery of these details has a considerable effect on the performance of the face manifold construction. We solve this problem by using the last two ellipses before the termination of the

ellipse reduction process. The space between the last ellipse and the second last ellipse, which is easily calculable, is shown in gray color in the fifth and fourth images of Figure 2.4 and Figure 2.5 respectively. Since the second last ellipse is expected to cover a larger portion of the nose tip, we include the pixels labeled as face pixels in the second last ellipse in the final face extraction output, i.e., the final output of the face extraction procedure consists of the combination of all pixels in the last ellipse and the pixels labeled as face pixels in the second last ellipse.

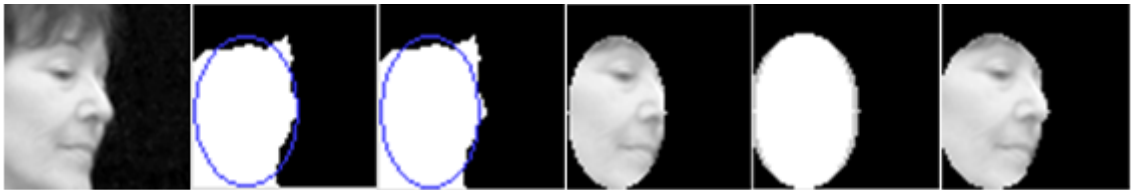


Figure 2.4 Whole face extraction process: The images are shown in the following order: Original face, initial ellipse fitting, final ellipse fitting, the face region extracted with respect to the last ellipse, plot of the last two ellipses, and the face region extracted with respect to the last two ellipses**.**

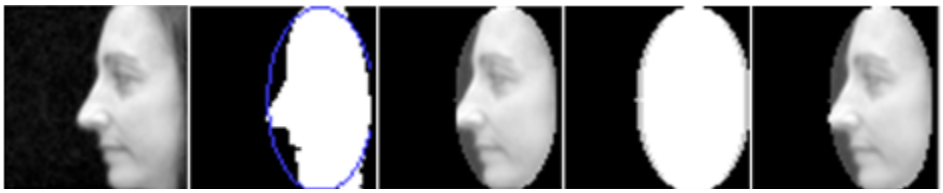


Figure 2.5 Whole face extraction process: The images are shown in the following order: Original face, initial-final ellipse fitting, the face region extracted with respect to the last ellipse, plot of the last two ellipses, the face region extracted with respect to the last two ellipses**.**

2.1.2. Frontal Face Selection

We will build the initial texture pattern based on an image chosen among the input data. The frontal face image encloses both the left and right parts of the face texture. Therefore, it is suitable for the initializing the texture map. In this section we will focus on how to select automatically the frontal image from the database.

The selection is achieved using the nose inclination characteristics. We know that without illumination perturbations, the nose is the most prominent part of the face, which has a high

gray-scale value. With this characteristic feature in mind, if we check the pixels with the highest intensity values, we notice that we can extract the nose inclination (see Figure 2.6 and Figure 2.7). Therefore, applying a threshold on pixel intensity values, we obtain regions with sharp edges that are likely to correspond to the nose part. Then, the rest of the problem can be solved using a line detection algorithm, and the ideal method for this purpose is the Hough transform (5).

Figure 2.6 Original input face sequence

Figure 2.7 Most prominent parts of the face images shown in Figure 3.1

2.1.2.1. Hough Transform

Hough transform is a feature extraction technique, whose purpose is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This procedure is carried out in the parameter space, where object candidates are found as local maxima. We use the Matlab implementation for detecting straight line. However, some lines that do not correspond to the nose might be detected by the Hough transform. Therefore, a post-processing step is implemented to remove these aberrant lines, for example, the horizontal ones and the ones near boundaries. Once we have applied this post-processing step, the remaining lines should characterize the nose inclination. Some results are shown in Figure 2.8. Therefore, in order to find out which one is the frontal image, we only have to calculate the energy on the left and right sides of the straight line. The image having the most vertical line and a balanced distribution of the energy on the two sides corresponds to the frontal face. The energy definition is given in equation [2.3].

$$E = \sum_i \sum_j |I(i,j)|^2 \qquad [2.3]$$

Where $I(i,j)$ is the intensity value of the image, the middle point of the line is taken as the separation limit between the left and right part of the face images.

Figure 2.8 Some results of nose line detection.

2.2. Virtual Camera

In this section, the focus is on the generation of a face image from a specific viewpoint, given the texture pattern, the depth pattern of the face model and the transformation parameters. In the generation of arbitrary views of the face model, we simulate a real camera that captures 2D images of a 3D scene. We call our view generation tool the Virtual Camera.

The simple pinhole camera model is used to achieve this purpose. Due to the wide camera scenes, some restrictions are imposed in order to simplify the acquirement of the image. For instance, we adopt the convention of fixing the camera center in z axis, and instead of moving the camera around the subject, the subject itself is performing rotations. It means that the translation of camera to the viewpoint and its posterior rotation focusing to the subject can be summarized in only performing rotation of the 3D face. In addition, the distance between the camera and the subject is always constant, the focal length parameter model the effect of zoom in and zoom out. Hence, the viewpoints are yielded in sphere around the face model.

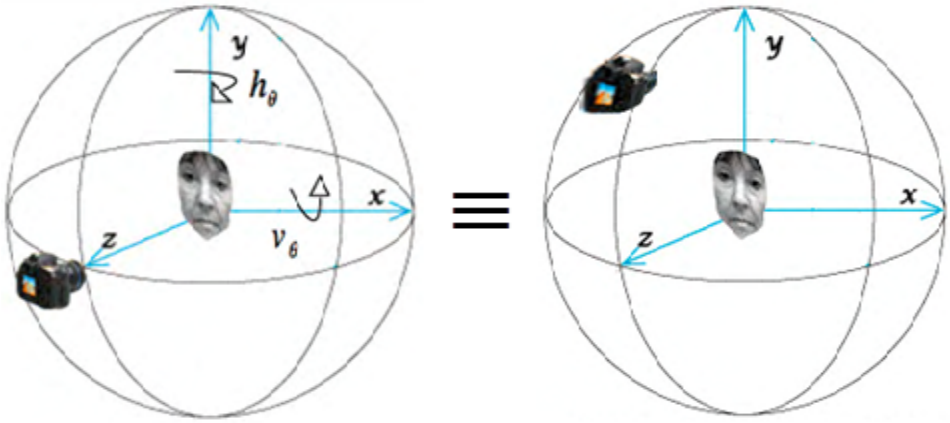


Figure 2.9 Equivalent scenes if we rotate the face or we translate the camera and rotate it to focus.

In Figure 2.9 is shown the general scene.

Once the general framework has been defined, the view generation can be obtained as described with equation [2.4]:

$$I_i = G_{\lambda_i}(p) \qquad [2.4]$$

Where $\lambda = (t_x, t_y, zoom, h_\theta, v_\theta)$, are the transformation parameters, called also registration parameters. t_x, t_y indicates the translation in image plane after projection, h_θ, v_θ are the rotation parameters and $zoom$ the scale parameter.

The estimated image acquirement can be summarized in three steps:

1. The first step is to obtain the camera coordinates of the face model in the required viewpoint. Consider that initial face model is represented in face coordinate system, where the subject is situated at origin and facing through z axis. Then, in order to capture different camera views, as they are situated all in a sphere around the model, we have to turn the subject respect to two spherical angles, h_θ and v_θ. Mathematically , the change of coordinates between the face coordinate system and the camera coordinate system is given by:

$$X_{CAM} = [R, -C] * X_w \qquad [2.5]$$

where $X_{CAM} = [x_{CAM}, y_{CAM}, z_{CAM}]$ and $X_w = [x_w, y_w, z_w]$ are the coordinates of the same 3D point in the camera coordinate system and the face coordinate system. And

$R = R_{h_\theta} * R_{v_\theta} = \begin{bmatrix} \cos(h_\theta) & 0 & \sin(h_\theta) \\ 0 & 1 & 0 \\ \sin(h_\theta) & 0 & \cos(h_\theta) \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(v_\theta) & \sin(v_\theta) \\ 0 & \sin(v_\theta) & \cos(v_\theta) \end{bmatrix}$ is the rotation matrix, and

$C = [0,0,165]$ is the fixed distance between the camera center and the face model on the z-axis.

2. Once we have the camera coordinate representation of the object points, the second step is to project them onto the image plane. We perform the projection using a pinhole camera model (6). The mapping between the 3D point and its 2D projection can be expressed as:

$$(x, y, z) \rightarrow \left(f\frac{x}{z}, f\frac{y}{z} \right) = (j_{nt}, i_{nt}) \qquad [2.6]$$

where f is the focal length parameter given by:

$$f = -\frac{\text{focal length}\,(mm)}{zoom} * \frac{\text{image dimension}\,(\text{pixels})}{\text{sensor dimesion}\,(mm)} \qquad [2.7]$$

To simplify the camera environment, focal length and sensor dimension are set to some typical commercial camera values, specifically, 50mm has been picked for the focal length and 20mm for the sensor dimension. However, image dimension depends on the input images, in our case is set to 64 pixels because of square image shapes with a 64x64 pixel resolution.

Considering that the camera can only be translated around the sphere, the zoom parameter is essential to capture close-up and distant images of the model, and is set by default to one. Values larger than one implies zoom out, and contrarily, values less than one indicates zoom in.

The illustration of the 3D-2D mapping with the pinhole camera model is shown in Figure 2.10.

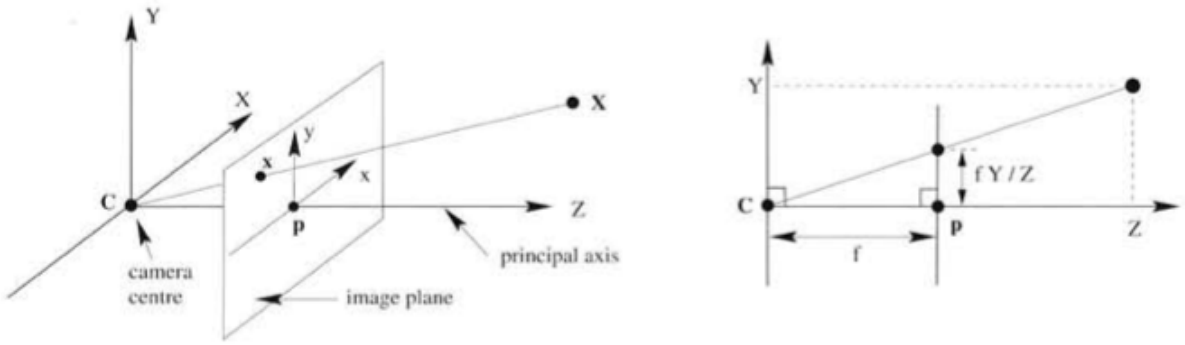


Figure 2.10 Pinhole camera geometry **(6)**. C is the camera center and p the principal point. The camera center is here placed at the coordinate origin. Note the image plane is placed in front of the camera center.

If homogeneous coordinates are used (6), the matrix representation of the projection is obtained as in equation [2.8]:

$$\begin{pmatrix} fx \\ fy \\ z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \qquad [2.8]$$

In the computation of the exact mapping between 3D points and the pixel coordinates of their projections on the image plane, the main difficulty is to handle the occlusions. For

instance, if an image views the right side of a face, the points on the left side of the face should not be represented on the image plane. However, two points from the left and right sides of the face may be seen to project onto the same pixel when the projection is computed. Thus, we apply patch clustering to solve the problem as follows. We divide all 3D points into clusters of size 8x8, such that every patch has 64 points. Such a clustering is straightforward to do in our case, since our 3D points are defined on a regular angular grid. Then, whenever two or more points coming from different patches project on the same pixel, we look at the distance between the camera center and these patches, and eliminate the 3D points that do not come from the patch with the smallest distance to the camera center, which are occluded points. Some results are shown in Figure 2.11.

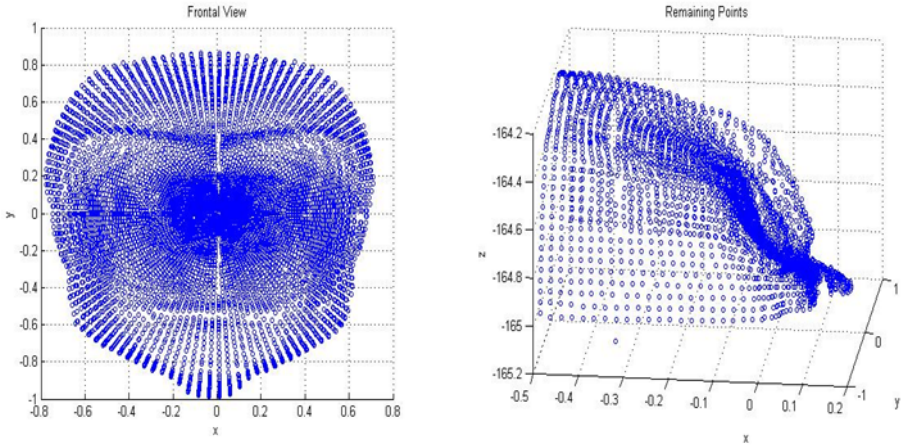


Figure 2.11 Left image: without performing rotation. Right image: performing horizontal rotation with $\frac{\pi}{2}$.

3. As soon as the non-occluded points projecting to each pixel are determined, the assignment of the intensity value of the pixel can be achieved by averaging the texture values associated to these points. However, all the images acquired by this process are centered shapes, because we are not performing translation in 3D space, images translation procedure is implemented by the Matlab function "imtransform" to allow us to move the face through the image plane.

$$(i, j) = (i_{nt} + t_y, j_{nt} + t_x) \qquad [2.9]$$

This whole process can be defined by the expression:

$$[i, j] = T_{\lambda}([x_w, y_w, z_w]) \quad [2.10]$$

Therefore, we define the image value as:

$$I(i, j) = \frac{1}{N} \sum_{i}^{N} T_{\lambda}(x_i, y_i, z_i) \quad [2.11]$$

And the inverse projection, called also back-projection, is given by:

$$I(r, c) = \frac{1}{L} \sum_{i \in I}^{L} A([x_{w_i}, y_{w_i}, z_{w_i}]) \quad [2.12]$$

Where I is the set of 3D points $[x_{w_i}, y_{w_i}, z_{w_i}]$, which the projecting point (i, j) belong the square region defined by $(r-1, c-1)$ and (r, c), and operator A give the texture pixels associate to the 3D point $[x_w, y_w, z_w]$.

In addition, the inverse or back-projection can also be defined as:

$$[x_{w_i}, y_{w_i}, z_{w_i}]_{i \in I} = T^{-1}{}_{\lambda}(r, c) \quad [2.13]$$

This transformation information is stored in two structures in the algorithm implementation, which save the 2D projections of 3D points (the assigned pixels), and the 3D back-projections of 2D pixels.

2.3. Depth and Texture Pattern Initialization

2.3.1. Assignment of Depth Map

As we have mentioned in the introduction, both depth and texture model are necessary to build a consistent face manifold. In general, for 2D images, the recovery of the depth map requires the internal and external calibration of the cameras viewing the scene. However, in our case we have the side information that the scene consists of a face. Therefore, we have chosen to make use of a general adjustable 3D face model, which we adapt to some particular faces from our database. CANDIDE-3 it is a very simple, easily adaptable parameterized mesh model.

2.3.1.1. CANDIDE-3

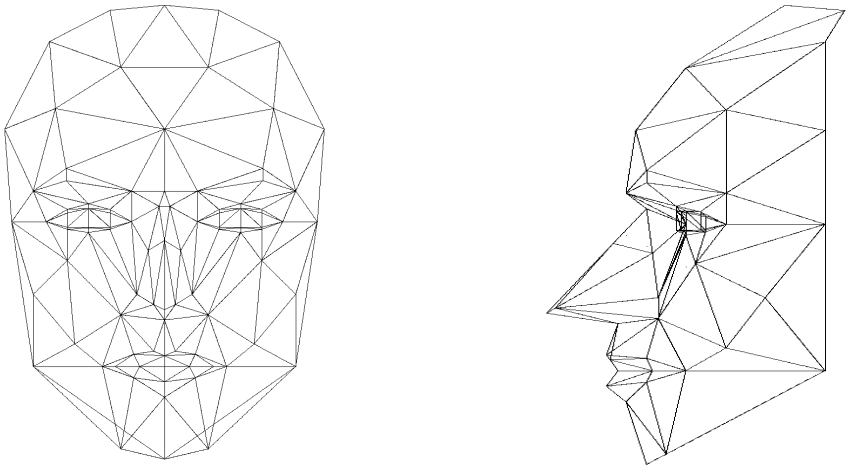


Figure 2.12 Frontal view of CANDIDE model in the left image, and side view in the right image

CANDIDE-3 is specifically developed for model-based coding of human faces. It is a wireframe pattern, which contains several control units. These units are classified basically in two groups: first of them, general action units, administrates the 3D rotations and translations of the model. In other hand, we have local action units, which are composed of shape units and animation units. A shape unit defines a deformation of a standard face towards a specific face, and the animation unit tries to model the movements of face muscles, so that different expressions can be created. For more details of these units see the paper (7).

The effect of all these control units can be incorporated into the single equation

$$\mathbf{g} = \mathbf{R}\mathbf{S}_3\left(\bar{\mathbf{g}} + \mathbf{S}\sigma + \mathbf{A}\alpha\right) + \mathbf{t} \qquad [2.14]$$

Where $\bar{\mathbf{g}}$ contains the vertices coordinates of the original model, and $\mathbf{g}$ contains the vertices of the adapted model. There are three components to control the global motion: $\mathbf{R} = R(r_x, r_y, r_z)$, which is a rotation matrix, $\mathbf{S}_3 = S_3(s_x, s_y, s_z)$ is a scaling matrix through all the three dimensions, and $\mathbf{t} = t(t_x, t_y, t_z)$ a translation vector. Besides, the columns of $\mathbf{S}$ and $\mathbf{A}$ are

the shape and animation units respectively, and thus the vectors σ and α contain the shape and animation parameters.

Thanks to the frontal face detection algorithm, we can use the frontal image for the model adaptation. Then, the rotation matrix in the above formula is not needed.

For simplicity of model control, we have created a graphical user interface in Matlab to reshape the original wireframe model according to our particular face images. A view of the interface is shown in Figure 2.13

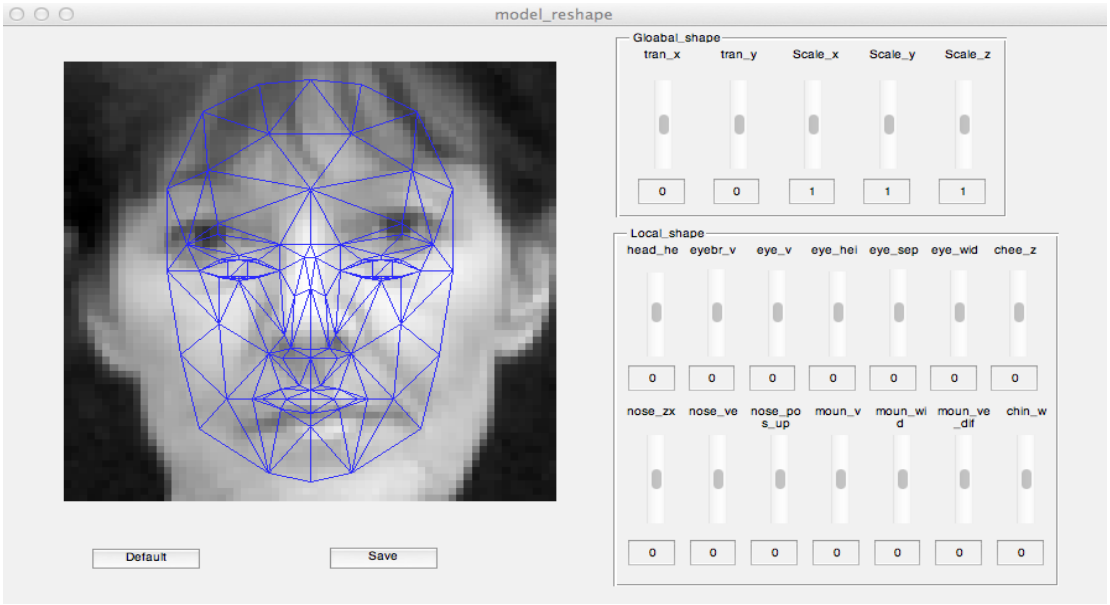


Figure 2.13 User interface for matching the CANDIDE model to a particular face.

Once we fit the general face model to the frontal face image, an interpolation process is required to refine the mesh in order to get a dense depth map, which will later be converted to spherical coordinates. The spherical coordinates are required for posterior dimension reduction with spherical atoms. Section 2.4 is detailed the whole process.

In fact, there are two interpolation processes during the transition from the mesh model to the dense depth map in spherical coordinates. The first one allows us to refine the mesh model, and in the second one we use K-nearest neighbors (K-NN) in order to obtain the spherical representation of our face model. For the mesh refinement process we have used a triangular

mesh with a spline interpolated 4-split method provided by Dirk-Jan Kroon[3]. And the K-NN step is implemented by a Matlab function called "TriScatteredInterp".

The spherical grid has been chosen to have size 128x128 regular grid composed by θ and φ (Appendix A) angles. However, in practice we use only the half of the sphere with positive z values, which means that only the half of the grid points are kept and another half corresponding to the back part of the head are discarded.

There are two main reasons to choose this particular grid size. Firstly, the grid size has to be bigger than our image resolution (64x64) for a reasonable image quality. Also, the implementation of the spherical matching pursuit algorithm that we use requires the grid size to be a power of 2 (see section 2.4 for more detail).

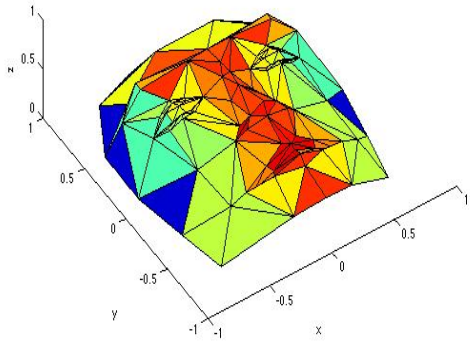


Figure 2.14 Initial mesh model

In the Figure 2.14 and Figure 2.15 provide an illustration of these steps.

[3] http://www.mathworks.com/matlabcentral/fileexchange/16215-triangular-mesh-refinement/content/refinepatch.m

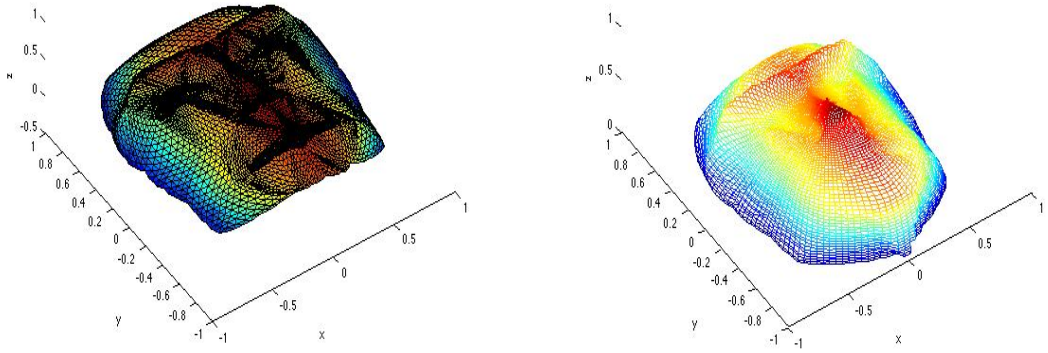


Figure 2.15 Left image, mesh refinement. Right image: representation in spherical coordinates

2.3.2. Assignment of Texture Map

In the previous section, how to build a first depth pattern has been explained. In this section, the objective is to initialize the texture one. Similar to the previous section, the frontal face is chosen to achieve this purpose because it covers the left and right side of face texture. First of all, we associate a constant intensity value to all the points of the initial depth model. It can be any value. Then, using the virtual camera, and setting the registration parameters as the frontal ones, the projected image is obtained. If the associated texture intensities are the right ones, the projected image should correspond to frontal face. Consequently, the mismatch pixels are employed to change the associated intensity value. Mathematically the sequence can be expressed as:

1. $\hat{I}_F = G_{\lambda_F}(\hat{p}) = G_{\lambda_F}\left(\hat{p}_d, \hat{p}_t\right)$
2. $M = \{(r,c) \mid I_F(r,c) \neq \hat{I}_F(r,c)\}$
3. $[x_{w_i}, y_{w_i}, z_{w_i}]_{i \in I} = T^{-1}{}_{\lambda}(r,c)$
4. $A([x_{w_i}, y_{w_i}, z_{w_i}]_{i \in I}) = I(r,c) \quad \forall (r,c) \in M$

The new texture pattern after this inverse-assignation in spherical coordinates is shown in Figure 2.16.

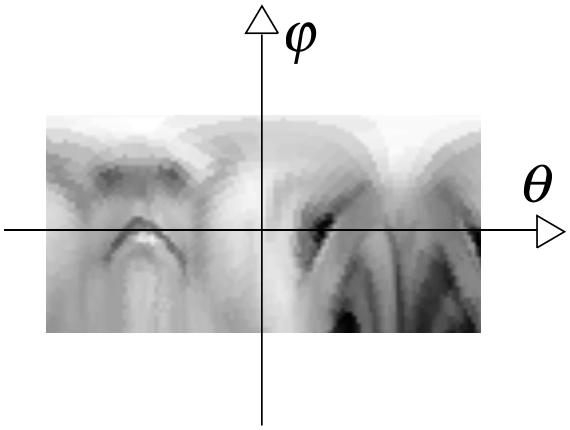


Figure 2.16 Initial texture pattern in spherical coordinates

If the obtained texture pattern is mapped over the depth one, the initial entire pattern is illustrated in the Figure 2.17.

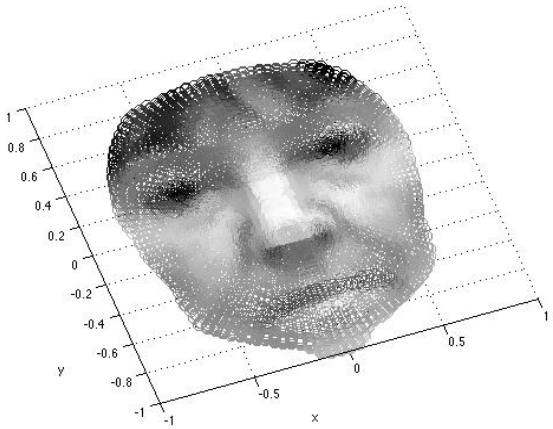


Figure 2.17 Depth pattern combined with texture pattern.

2.4. Depth and Texture Dimension Reduction

2.4.1. Matching Pursuit

Matching pursuit (MP) is an iterative signal approximation algorithm (8), whose aim is to obtain a sparse approximation of a signal in terms of some dictionary elements called atoms. It is

common practice to use a redundant dictionary for a good signal approximation, which means that there are more atoms in the dictionary than necessary in order to span the subspace generated by the dictionary. The matching pursuit algorithm selects atoms iteratively in order to improve the signal approximation progressively. In each iteration, the atom that best matches the residual signal is selected. Then, the residual signal is updated by subtracting the contribution of the selected atom to the signal. The process continues until predefined approximation accuracy or sparsity constraint is reached.

The algorithm requires a predefined dictionary. In our application, we adopt a dictionary model generated by applying a certain set of geometric transformations to a mother function. Let H be a Hilbert space, and g be a mother function in *H*. Then the dictionary D in H is given by the family:

$$D = \{g_\gamma\}_{\gamma \in \Gamma} \qquad [2.15]$$

of unit-norm vectors where each vector g_γ is derived from the mother function *g* by applying a geometric transformation specified by the vector γ. If $Span(D) = H$, we say that the dictionary is complete.

Before explaining how the algorithm works, let us mention some properties of signal approximation in a Hilbert space.

First, we know that if we take an atom g_{γ_0} from a complete dictionary, an arbitrary function *f* in *H* can be written as:

$$f = \left\langle f, g_{\gamma_0} \right\rangle g_{\gamma_0} + Rf \qquad [2.16]$$

where Rf is the residual vector after approximating the function f in g_{γ_0} direction, and $\left\langle f, g_{\gamma_0} \right\rangle$ is the inner product in the Hilbert space. The inner product between two vectors f and g is defined as:

$$\left\langle f, g \right\rangle = \int_x f(x) g(x) dx \qquad [2.17]$$

The norm of a vector *f* is thus given by

$$\|f\| = \sqrt{\left\langle f, f \right\rangle} \qquad [2.18]$$

An interesting observation is that g_{γ_0} and are Rf orthogonal, which is explained in (8). So from equation [2.16] is possible to conclude that:

$$\|f\|^2 = \left|\left\langle f, g_{\gamma_0} \right\rangle\right|^2 + \|Rf\|^2 \qquad [2.19]$$

From equation [2.19] we can clearly see how MP works. A good atom for the approximation of f should give low residual energy, or equivalently, maximize the correlation, or the inner product $\left\langle f, g_{\gamma_0} \right\rangle$ with the function f . Therefore, at each step the atom that has the highest correlation with the residual function is selected. This condition implies that an adequate and efficient dictionary is very important.

Furthermore, writing $f = R^0 f$, the choice of the atom in each iteration can be expressed as follows.

$$g_{\gamma_n} = \max_{\gamma \in \Gamma} \left(\left| R^n f, g_\gamma \right| \right) \qquad [2.20]$$

Where n is the number of the iterations, and Γ is the space of atom parameters, and the n^{th} order residue with $n \geq 0$ can be described as:

$$R^n f = \left\langle R^n f, g_{\gamma_n} \right\rangle g_{\gamma_n} + R^{n+1} f \qquad [2.21]$$

Therefore, in the n^{th} iteration, the decomposition of the function of interest is given by

$$f = \sum_{i=0}^{n-1} \left\langle R^i f, g_{\gamma_i} \right\rangle g_{\gamma_i} + R^n f = \sum_{i=0}^{n-1} \alpha_i g_{\gamma_i} + R^n f \qquad [2.22]$$

where $\alpha_i = \left\langle R^i f, g_{\gamma_i} \right\rangle$ are the computed coeficients.

Moreover, two more properties are proved in (8) under the completeness assumption of the dictionary. The first one is the following. If we take N iterations, as $N \to \infty$, the residual part tends to zero. And the input signal is being decomposed as only a linear combination of the dictionary atoms. Secondly, the exponential decay of the residual function in a finite dimensional space has also been proved.

2.4.2. Texture Matching Pursuit

In this section we explain in more detail how the MP is applied to the initial texture pattern achieved in the section 2.3.2. First of all, texture atoms should be acquired applying geometry transformations to mother function. As has been mentioned in the introduction, smooth and parametric atoms results in a more regular manifold, which facilitates the registration of face images with respect to the manifold. Therefore, the selected mother functions are the Gaussian and Anisotropic refinement function (9). For both functions, the transformation parameters are $\lambda_t = (d_x, d_y, \vartheta, \alpha_x, \alpha_y)$, where d_x and d_y are the motion variables, ϑ performs the atom rotation, and finally α_x, α_y are the x, y scale parameters. The mathematic expressions are shown in equation [2.23] and [2.24].

Consider $x = (r - dy) * \cos(\vartheta) - (c - dx) * \sin(\vartheta)$, and $y = (r - dy) * \sin(\vartheta) - (c - dx) * \cos(\vartheta)$.

$$\phi_{gauss} = \sqrt{\frac{2}{\pi}} e^{-\left(\left(\frac{x}{\alpha_x}\right)^2 + \left(\frac{y}{\alpha_y}\right)^2\right)} \qquad [2.23]$$

$$\phi_{AnR} = \sqrt{\frac{2}{3\pi}} \left(4x^2 - 2\right) e^{-\left(\left(\frac{x}{\alpha_x}\right)^2 + \left(\frac{y}{\alpha_y}\right)^2\right)} \qquad [2.24]$$

Calling the transformed atom as $a_{t_i} = Ut_{\lambda_{t_i}}(\phi_{gauss}$ or $\phi_{AnR})$, the estimation of the texture pattern with K atoms can be expressed as $p_t = \sum_{i=1}^{K} c_{t_i} a_{t_i}$.

Once we have chosen the mother function and the transformation parameters, we should define the dictionary. $N_t = 4000$ points have been chosen in the parameter domain in order to achieve this purpose. The parameter domain is defined as:

$$\Gamma_t = \{d_x \in \left[-\frac{col}{2}, \frac{col}{2}\right], d_y \in \left[-\frac{row}{2}, \frac{row}{2}\right], \vartheta \in [0, 2\pi], \alpha_x \in \left[0, \frac{col}{5}\right], \alpha_y \in \left[0, \frac{row}{5}\right]\} \qquad [2.25]$$

Consequently, we have a total of 8000 atoms in our dictionary, half Gaussian and another half AnR atoms.

Finally MP algorithm is applied to the initial texture model, the pseudo code is shown in the Table 1. The *TexThreshold* is set as $\frac{\langle p_t, p_t \rangle}{350}$, chosen to have a reasonable approximation of the

texture pattern with 200 atoms. The initial and approximated model is compared in the Figure 2.18.

$R^0 f = p_t$

$n=1$

$While(R^n f > TexThreshold)$

$g_{\gamma_{n-1}} = \max_{\gamma \in \Gamma_t} \left(\left| R^n f, g_\gamma \right| \right)$

$n = n+1$

$R^n f = R^{n-1} f - \left\langle R^{n-1} f, g_{\gamma_{n-1}} \right\rangle g_{\gamma_{n-1}}$

end

Table 1 Texture Matching Pursuit.

Figure 2.18 Left image: initial texture pattern. Right image: An approximation of texture pattern with 200 atoms.

2.4.3. Depth Matching Pursuit

The approximate of the initial model as a linear combination of the spherical atoms is also desired. In section 2.3.1 we mention that the pattern should be represented in regular spherical grid, but it was not specified. This grid is obtained by uniformly sampling a number of $N_\theta = N_\varphi = 128$ points to the two spherical angles θ and φ. The Matlab notation of the sampling process is shown in equation [2.26] and [2.27].

$$\theta = 0 : \frac{\pi}{N_\theta - 1} : \pi \qquad [2.26]$$

$$\varphi = -\pi : \frac{2\pi}{N_\varphi - 1} : \pi \qquad [2.27]$$

Similar to the texture MP, spherical atoms yielded in the same grid, are defined as geometric transformations of the mother function. Tosic et. al. (10) propose a progressive coding scheme for 3-D objects, based on overcomplete signal expansions on the 2-D sphere. We chose only the low frequency mother function present in the paper, because is sufficient to approximate a depth face (2). The mathematical expression is shown in the equation [2.28].

$$g_{LF} = e^{(-\tan^2\frac{\theta}{2}(a^2{}_1\cos^2(\varphi)+a_2^2\sin^2(\varphi)))} \qquad [2.28]$$

Some atoms are shown in the Figure 2.19.

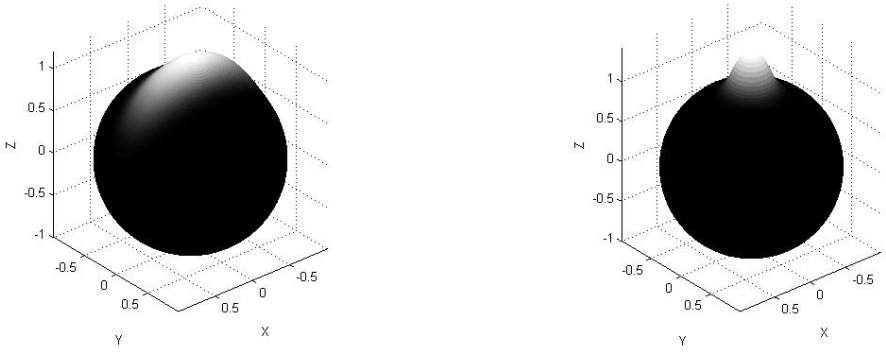


Figure 2.19 Left image: depth atom with $a_1 = 3, a_2 = 8$. Right image: depth atom with $a_1 = 10, a_2 = 10$.

However, all the atoms created in this way are centered in the North Pole, which means $\theta = 0,\ \varphi = \frac{\pi}{2}$ in spherical coordinates. Therefore, motion of the atom in the sphere has to be implemented. In (10) they use three spherical rotation parameters $r_\varphi, r_\theta, m_\varphi$ to solve the problem. The matrix forms of the three parameters to perform the rotation in Cartesian coordinates are shown in [2.29],[2.30] and [2.31].

$$R_\varphi = \begin{bmatrix} \cos(r_\varphi) & \sin(r_\varphi) & 0 \\ -\sin(r_\varphi) & \cos(r_\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad [2.29]$$

$$R_\theta = \begin{bmatrix} \cos(r_\theta) & 0 & \sin(r_\theta) \\ 0 & 1 & 0 \\ -\sin(r_\theta) & 0 & \cos(r_\theta) \end{bmatrix} \quad [2.30]$$

$$M_\varphi = \begin{bmatrix} \cos(m_\varphi) & \sin(m_\varphi) & 0 \\ -\sin(m_\varphi) & \cos(m_\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad [2.31]$$

Hence, if the initial atom is expressed in Cartesian coordinates as b_{np}, the transformed atom is described by the equation [2.32].

$$b_t = R_\varphi(r_\varphi)R_\theta(r_\theta)M_\varphi(r_\varphi)b_{np} \quad [2.32]$$

In this manner, the atoms can be expressed as $a_d = Ud_\lambda(g_{LF})$, where $\lambda = (a_1, a_2, r_\varphi, r_\varphi, m_\varphi)$, and the approximated depth model by L atoms as $p_d = \sum_{i=1}^{L} c_{d_i} a_{d_i}$.

Some rotated atoms are illustrated in the Figure 2.20:

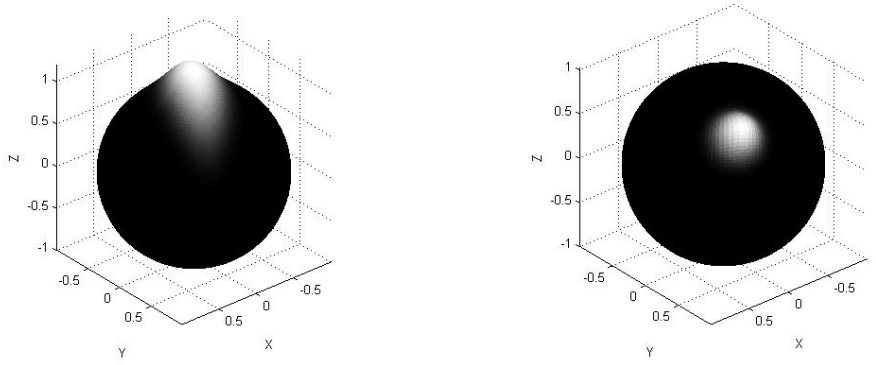


Figure 2.20 Left image: depth atom with $a_1 = 3, a_2 = 8, r_\theta = 0, r_\varphi = \frac{\pi}{3}, m_\varphi = 0.$

Right image: depth atom with $a_1 = 10, a_2 = 10, r_\theta = \frac{\pi}{4}, r_\varphi = \frac{\pi}{3}, m_\varphi = 0.$

After the definition of depth atoms, we should establish the dictionary. Unlike the texture approach, where we take a random grid over parameter domain to construct it. Here, the size of

dictionary is defined by number of sample point in the two scale parameters. Due to computational efficiency, instead of computing the inner product, we make use of the correlation in transform spherical domain provided by YAWTB[4] toolbox. Thanks to the equivalence between the inner product and the correlation in transform spherical domain, giving two atoms situated in North Pole(without applying rotation transformations), the inner product between the rotated version of both can be easily collected by the "sofcorr" (11) function. Suppose *fst*() symbolizes the spherical transformation and $a_{d_1}(r_\varphi, r_\theta, m_\varphi) = Ud_{(a_1, a_1, r_\varphi, r_\theta, m_\varphi)}(g_{LF})$, $a_{d_2}(r_\varphi, r_\theta, m_\varphi) = Ud_{(a_2, a_2, r_\varphi, r_\theta, m_\varphi)}(g_{LF})$ two atoms yield in $128x128$ uniform spherical grid. Defining the sampling of rotation angles in Matlab notation as in equations [2.33],[2.34] and [2.35].

$$r_\varphi = -\pi : \frac{\pi}{N_\varphi - 1} : \pi \qquad [2.33]$$

$$r_\theta = 0 : \frac{\pi}{N_\theta - 1} : \pi \qquad [2.34]$$

$$m_\varphi = -\pi : \frac{\pi}{N_\varphi - 1} : \pi \qquad [2.35]$$

Then, $C_{rr} = softcorr(fst(a_{d_1}(0,0,0)), fst(a_{d_2}(0,0,0)))$ is a $128x128x3$ matrix, where $C_{rr}(i,j,k) \equiv \left\langle a_{d_1}(r_\varphi(i), r_\theta(j), m_\varphi(k)), a_{d_2}(r_\varphi(i), r_\theta(j), m_\varphi(k)) \right\rangle$.

In this project, the logarithmic sampling in scale parameters is taken as:

$$a_1 = 2^{\log_2(1):\frac{1}{2}:\log_2(\frac{N_\theta}{2})} \qquad [2.36]$$

$$a_2 = 2^{\log_2(1):\frac{1}{2}:\log_2(\frac{N_\varphi}{2})} \qquad [2.37]$$

The size of the dictionary is shown in the table Table 2.

N_θ	$N_\varphi \leftarrow r_\varphi$	$N_\varphi \leftarrow m_\varphi$	$size(a_1)$	$size(a_2)$	North pole atoms	Size of dictionary
128	128	128	13	13	169	354418688

Table 2 Size of dictionary

[4] http://sites.uclouvain.be/ispgroup/yawtb/pmwiki.php/Main/Presentation

Once we have constructed the dictionary, the MP is applied to the first depth map. A pseudo code is shown in Table 3.

The number of atoms to represent an initial depth map is chosen to have a reasonable approximation. And with only 20 atoms, we can capture the main shapes of the pattern. In Figure 2.21, the difference between the initial and approximated depth pattern is illustrated.

$R^0 f = p_d$

for k=1:L

for i=1:size(a_1)

for j=i:size(a_2)

$C_{rr} = softcorr(fst(R^n f), fst(a_{d_i}(0,0,0)))$

$C_{\max} = \max_{1<n<N_\varphi, 1<m<N_\theta, 1<p<N_\varphi} C_{rr}(n,m,p)$

$\alpha = r_\varphi(n)$

$\beta = r_\theta(m)$

$\delta = m_\varphi(p)$

$n = n+1$

$g_{\gamma_{n-1}} = a_{d_i}(\alpha, \beta, \delta)$

$R^n f = R^{n-1} f - \left\langle R^{n-1} f, g_{\gamma_{n-1}} \right\rangle g_{\gamma_{n-1}}$

end

end

Table 3 Depth Matching Pursuit.

Figure 2.21 Left image: initial depth model. Right image: approximation of the model with 20 atoms.

2.5. Initialization of Registration Parameters

In Section 2.3, we have explained the computation of approximate 3D face compost by texture and depth pattern. Now, in order to improve the model by refining its texture and depth pattern, we have first to find out the transformation parameters that register each one of the input images with respect to the face model.

The registration process is analogous to the estimation of the transformation parameters that we have discussed in Section 2.2. It can be achieved in the following way.

1) Smooth the detected faces with a low pass Gaussian filter and taking the initial depth model and the approximated texture pattern for the registration process.
2) Estimate h_θ and the image translation parameters from the nose inclination information.
3) Taking a reasonable uniform grid for the zoom parameter, and for each of them, re-estimate the translation parameter near the face region, and choose the one that gives the lowest error.
4) Finally, update the all five parameters with a gradient-descends algorithm.

2.5.1. Smooth the detected faces with a Gaussian filter

Before any registration process, we have to preprocess the input data. In this case, applying low pass Gaussian filter to the detected faces. The reason of such an action is that smooth images present better registration performance. Therefore, the approximated texture pattern is taken instead of the initial one. However, for depth model, we choose the initial one because of the possession of more real shape information.

2.5.2. Estimate Horizontal Angle from Nose Inclination

Now that we have the smooth detected faces as the target images, and the approximated texture pattern combined with initial depth model producing estimated images, the transformation parameters are the goal.

Estimating all the five parameters together is complicate. Therefore, we use the available information to have a first approximation of three of them. We assume that $zoom = 1$, and $v_\theta = 0$. Taking the information of the nose detection algorithm we know that each input image has a group of lines that models the nose inclination. Then, the extraction of parameter t_x, t_y is easy as computing the middle point of the set of lines, and the deviation of this point to the $center = \left(\frac{row}{2}, \frac{col}{2} \right)$ gives the value of t_y, t_x. The horizontal angle h_θ is set as the degree of inclination of each line. Thus, combining both information, for each line, the error expressed in equation [1.10] is computed, and those parameters that gives smallest distance to the target is chosen as the first estimation value.

2.5.3. Fitting and Region Search Process.

After having the estimated value of t_x, t_y, h_θ, we focus the search of the zoom parameter. We take a uniform grid of possible zoom values. Then, for each of them we seek the region near the initial position(± 2pixels) in order to find the best match which is the one that gives the minimum error.

2.5.4. Update the Parameters with Gradient Descends Algorithm.

Finally, all five parameters are updated using gradient descends algorithm. In the is shown the results of the registration process.

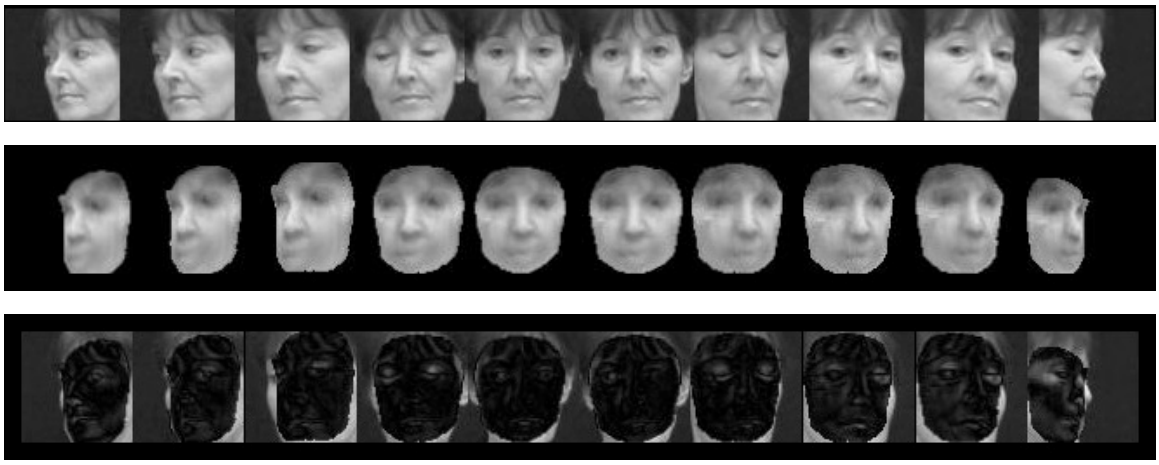


Figure 2.22 From up to dowm, we have the sequence: input images, registered images, difference between input images and registered images.

Chapter 3. MODEL REFINEMENT

3.1. Error reformulation

After the model initialization process, we can reformulate the error expression in the equation [1.10] as:

$$e = \sum_{n=1}^{N} \left| I_i - I_i \right|^2 = \sum_{n=1}^{N} \left| I_i - G_{\lambda_i}(p) \right|^2 = \sum_{n=1}^{N} \left| I_i - G_{\hat{\lambda}_i}\left(\sum_{i=1}^{L} c_{d_i} a_{d_i}, \sum_{i=1}^{K} c_{t_i} a_{t_i}\right) \right|^2 \quad [3.1]$$

Then,

$$e = f(\hat{p}_d, \hat{p}_t, \{\hat{\lambda}_i\}) = f(\{c_{d_i}, \lambda_{d_i}\}, \{c_{t_i}, \lambda_{t_i}\}, \{\hat{\lambda}_i\}) \quad [3.2]$$

In this way, the error can be described by three main constituents. The first component $\{c_{d_i}, \lambda_{d_i}\}$, are the depth coefficient and atom parameters. The second one $\{c_{t_i}, \lambda_{t_i}\}$ represents texture coefficient and atom parameters. And the last component $\hat{\lambda}_i = \left(t_x, t_y, zoom, h_\theta, v_\theta\right)$ is the estimated registration parameters. As all the parameter are estimated, or taken from some grid vector, there still some improvable space. In the next sections, we try to find out the best way.

3.2. Optimization of Texture Atom Parameters

As the error has contribution of three main components. We apply "superposition", which means improving one component keeping the rest constant. We start with texture refinement. The gradient descends algorithm is applied to improve coefficient and texture atom parameters. A plot of error by updating 20 atoms is shown in the Figure 3.1:

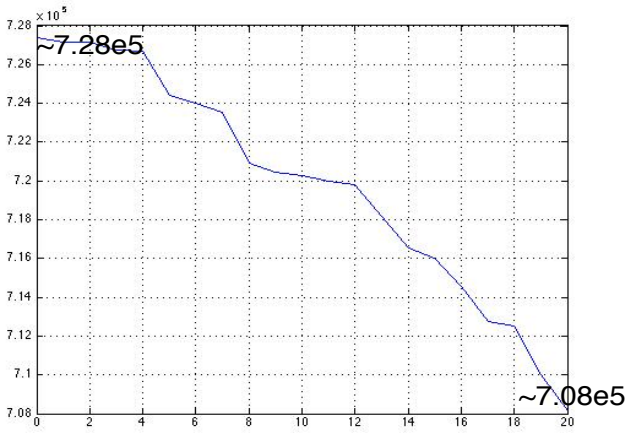


Figure 3.1 Error reduction updating 20 atoms.

As we can check in the Figure 3.1, the error is not significantly reduced. The main reason is that no having a proper registration parameter and depth model, it difficult to change the texture pixels in order to reduce the global error. For instance, the side view has a large error due to the incorrect shape, then, if we try to change the wrong texture pixels in order to get better match, the right ones from the frontal face have to be changed at the same time and maybe has a long error influence.

Also we want to remark that optimization process is enhanced keeping projection information of the 3D points, thus, the 3D to 2D time consuming process is supplied.

3.3. Optimization of Depth Atom Parameters

It has been demonstrated in the previous section that improving only texture atoms the reduction of error is unsatisfactory. This section concerns the improvement of the depth ones:

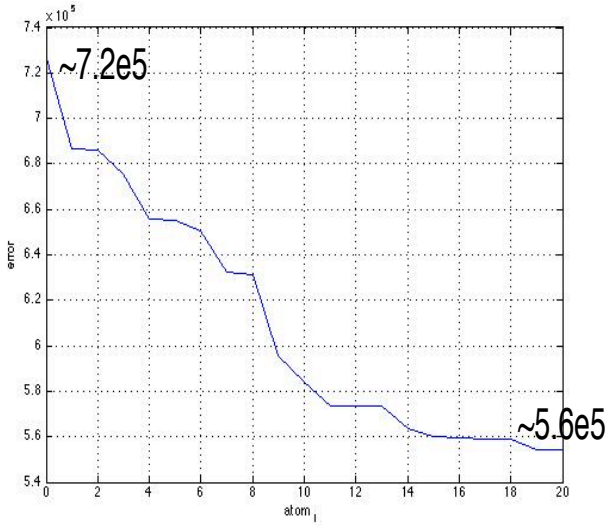


Figure 3.2 Error reduction updating 20 atoms.

In the Figure 3.2 the error behavior is shown, clearly, the reduction of the error is preferable than the texture approach, but the problem here is the high computational rate. Unlike the texture optimization, we cannot keep the projection information, because the depth model is changed every time that some atom is modified. Some improvement can be achieved by projecting only those patches where the atom has high contribution, but is still no comparable with texture optimization in terms of computational rate.

The approximated depth pattern and improved one is shown in Figure 3.3.

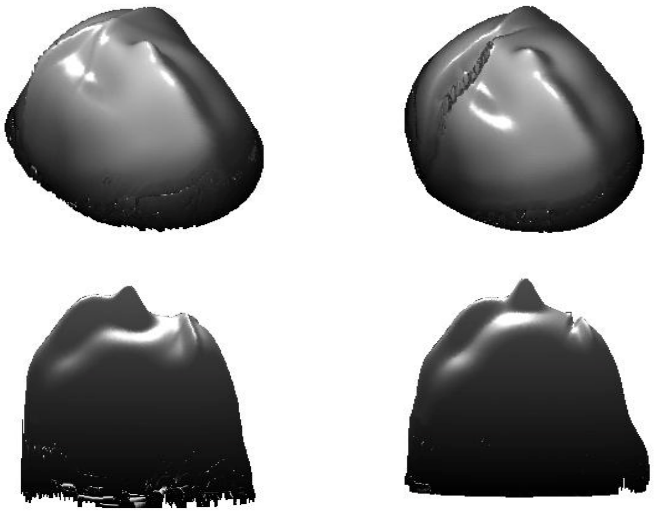


Figure 3.3 Left side: approximated depth pattern. Right side: improved depth pattern.

If we compare the two models, we notice that some wrong shape initialization is correct by the optimization process. For example, the elevation of the nose that was not high enough has been corrected. But also some undesirable changes have occurred as hole part in the eyebrow region. The reasonable argumentation for these unwanted changes is that we do not have the exact texture map and registration parameter.

3.4. Addition of Depth Atoms

We have seen during whole project that the depth pattern is the most problematic point for the good performance of our algorithm. In this section we present the way to extract depth information from the input image thanks to correlation between them.

In section 2.3.1 a CANDIDE model is introduced in order to initialize the first depth map. However, this model is the general wireframe model. Even after reshaping to our particular face, there is not guarantee that it can include all the real shape information. We propose adding new depth atoms to our approximated depth pattern to cover the missing information. Feature detection technics is applied to achieve this proposed, namely, the Affine Invariant Feature Detection(ASIFT) algorithm is employed[5] (12).

[5] http://www.ipol.im/pub/algo/my_affine_sift/

3.4.1. Feature Detection

Given two input images, with relation of affine transform between them, ASIFT computes those matching pixels of the images. Some results can be observed in the Figure 3.4.

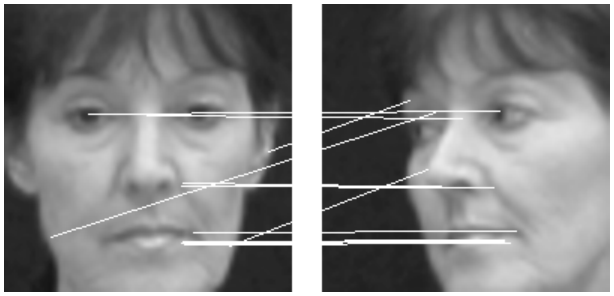


Figure 3.4 Applying ASIFT algorithm to two input images from database.

However, due to inexactly correspondence with affine transformation when the human performs head rotations. ASIFT can give aberrant points. For instance, the marked point in the Figure 3.5.

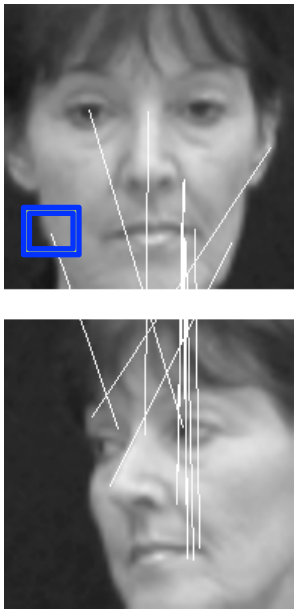


Figure 3.5 Wrong matching point.

Then, some preprocessing step should be applied in order to remove these points. In our case, having a general depth map, this process is immediate. The two filtering conditions are the following:

1. If the feature points are outside of the detected faces, result from face extraction, it means that they could be background points, which are no interesting.
2. For the second condition we need to obtain the estimated images corresponding to the inputs images of the ASIFT algorithm. Afterwards, we check if the distance between the back-projection of matching points is within some reasonable distance interval. Because, if we have a real depth model, these two matching points should back-project to the same 3D points. If the distance is higher than a certain threshold, we just discard these points.

After the preprocessing steps, we have a set of reasonable matching points. The problem now is how to extract depth information from them. The main idea is that if they effectively are describing the same feature of target depth face model, it means that the back-projection 3D points should be the same. If they are not near enough, we know that we have a wrong depth shape in this region. Then, some atoms with contribution in this region can be added in order to correct the shape. The interest of this viewpoint is that we are extracting depth information directly from the input image, here the problem of not having good enough texture model does not exist.

Hence, we know where we should add the atom, but we do not what the kind of atom is the best one. We classify the problem in two situations to deal with the problem:

1. When the back-projection rays meets before arriving to depth model. The atom with positive coefficient should be added in order to create a convex shape. Situation shown in Figure 3.6.

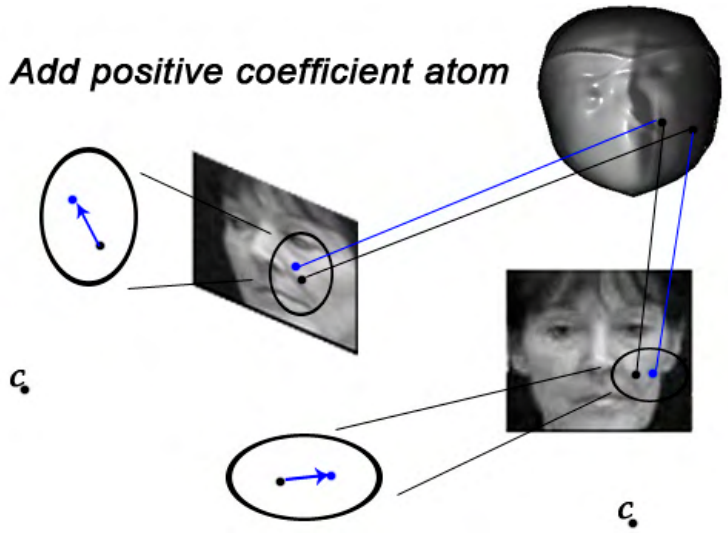


Figure 3.6 Situation where we have add atom with positive coefficient.

2. In contrary, when the back-projecting does not meet after arriving the depth pattern. The atom with negative coefficient should be added in order to formulate a hole in the region for make ray to meet in the surface. Situation shown in Figure 3.7.

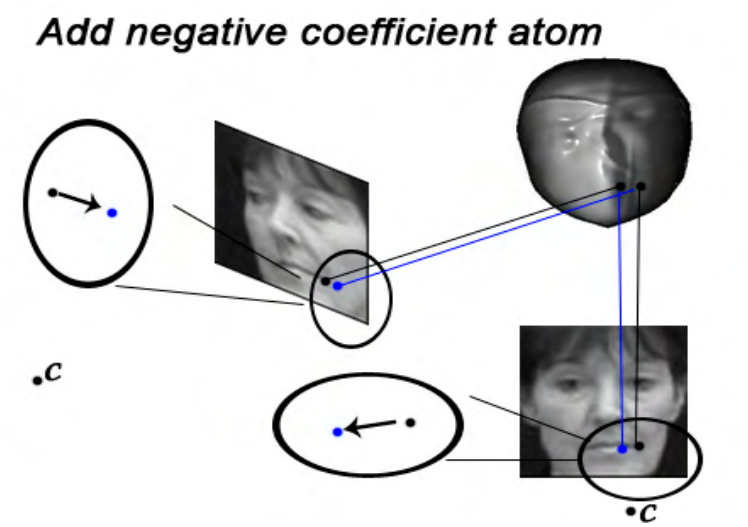


Figure 3.7 Situation where we have add atom with negative coefficient.

Once we have decided the problem scheme, gradient descends algorithm is applied to obtain the best atom. Some aberrant atoms, such as the one that the contribution goes outside the limits of the face model (unit-sphere), are removed. Defined (r_1, c_1) and (r_2, c_2) as the matching point in the input images I_1 *and* I_2, the error condition is set as in equation [3.3].

$$e = d(T^{-1}{}_{\lambda}(r_1, c_1), T^{-1}{}_{\lambda}(r_2, c_2)) + \frac{I_1(scatter_1) - I_2(scatter_2)}{\lambda} \qquad [3.3]$$

Where $scatter_1$ and $scatter_2$ is the projection region of the corresponding path to the back-projection points. And λ is the compensation parameter because of different domain of both error components.

3.5. Alternating Optimization of Depth Pattern and Texture Pattern

It has been seen that separate optimization of the different contributions has or small reduction performance or high computational rate. In this final step, we consider the optimization of alternating the three main components: depth map, texture map and registration parameters:

The sequence of the optimization is as following:

1. Depth optimization with all the depth atoms in order to get a general depth map.
2. Parameter optimization.
3. Texture optimization with only five atoms.
4. Add new depth atoms choosing two random input images. If they do not have any matching points, change the input images until some matching points are gotten.
5. Texture optimization with only five atoms, different from the point 2.
6. Restart the sequence $2 \rightarrow 3 \rightarrow 4 \rightarrow 5$ until convergence.

One sequence error plot is shown in the Figure 3.9, and the estimated images in Figure 3.8.

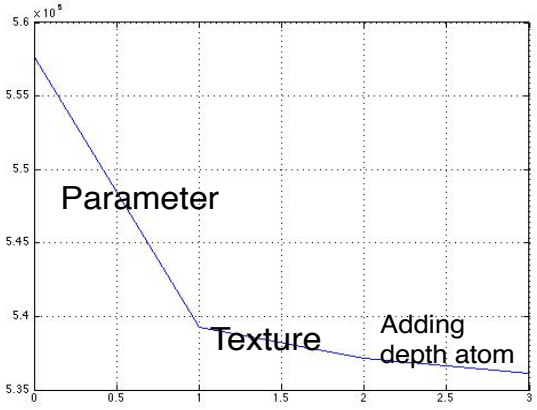


Figure 3.9 Error plot.

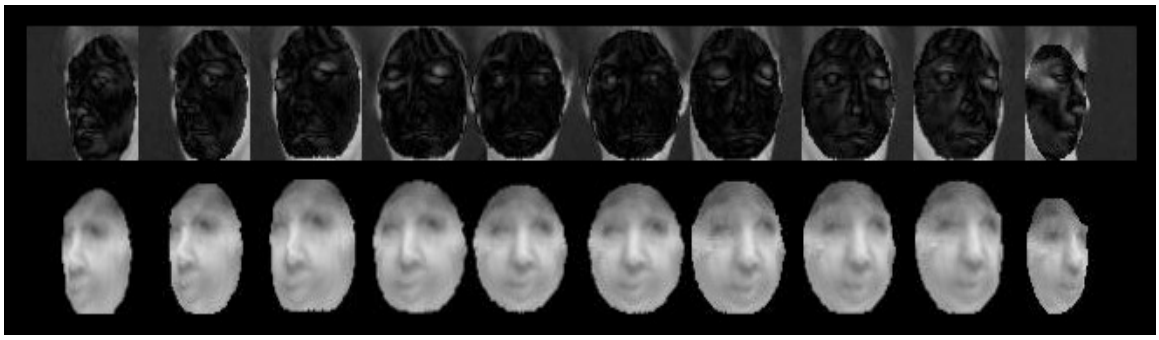


Figure 3.8 Estimated images and difference with input images.

As we can see in the Figure 3.8 , the estimated faces are more face-like than the ones before the optimization process, which are close to the wireframe model.

Chapter 4. CONCLUSIONS AND FUTURE WORK

4.1. Conclusion

We have studied the problem of approximating human faces using an analytical model. Although the proposed method based on initialization of the model, and their posterior optimization alternating registration parameters, depth and texture patterns is still lightly far for face recognition purpose. However, the viability seems very promising.

Thus, it can be used in pose-invariant face recognition applications, whereas most of the 2D face recognition methods have limited performance in case of pose variation.

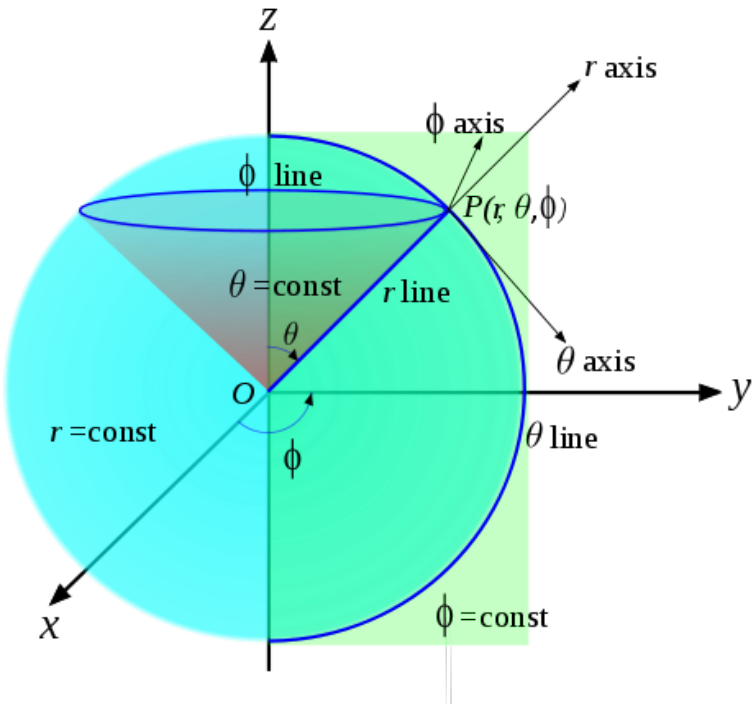
Also, the usage of smooth and parametric atoms in the model results in a more regular manifold, which facilitates the registration of face images with respect to the manifold, and the ease of efficient storage of the model information as the coding scheme.

4.2. Future work

- Joint utilization of multiple faces in model refinement.
- Compensation for illumination changes in the virtual camera model.
- Application of the method in face recognition.

Appendix A

Spherical coordinate system:



Reference

1. **M.A. TurkA.P. Pentlandand.** Face Recognition Using Eigenfaces.

2. **R. Sala Llonchal.et.** 3D face recognition with sparse spherical representations, Pattern Recognition. 2009.

3. **Y. ShengSadka and A. M. KondozA.H.** “Automatic face segmentation for 3D model-based video coding” Proceedings of IEE International Conference on Visual Information Engineering, Guildford, U.K. pp. 274-277. 2003.

4. **Tapas KanungoM. Mount,Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, Y. WuDavid.** An Efficient k-Means Clustering Algorithm: Analysis and Implementation.

5. **Richard O. DudaE. HartPeter.** Use the hough transformation to detect lines. 1971.

6. **ZissermanHartley and AndrewRichard.** Multiple View Geomatry in Computer Vision Second Edition, Cambridge University Press. 2004.

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

8. **Stephane G.MallatZhangZhifeng.** Matching pursuit with time-frequency dictionaries.

9. **FrossardVural and PascalElif.** Approximation of pattern transformation manifold with parametric dictionaries.

10. **I.TosicP.VandergheynstP.Frossard,.** Progressive coding of 3D objects based on overcomplete descompositions.

11. **TeamYAW ToolboxThe.** Yet Another Wavelet Toolbox Reference Guide, http://sites.uclouvain.be/ispgroup/yawtb/pmwiki.php/Main/Documentation.

12. **J.M. Morel and G.Yu.** ASIFT, A new framework for fully affine invariant image comparison. .

13. **P. PhillipsGrother, R. Michaels, D. Blackburn, E.Tabassi, and M. BoneP.** Face recognition vendor test: evaluation report. http://www.frvt2002.org. 2002.

14. **Y. ChanLin, and S. KungS.-H.** Video indexing and retrieval. In Multimedia Technology for Applications. B.J. Sheu, M. Ismail, M.Y. Wang, and R.H. Tsai, editors. Wiley, New York. 1998.

15. **P.N.BelhumeJ. P. HespanhaJ. KriegmanD.** Eigenfaces vs. Fisherfaces.