Diploma thesis

Štěpán Obdržálek
xobdrzal@fel.cvut.cz

Topic

Compensation of illumination effects on facial appearance using a 3D model.

In facial image analysis tasks, such as facial feature extraction, face verification and identification, the face illumination has crucial effect on exactness and credibility achieved.

The goal of this work is to develop and implement a method for the recovery of a 'canonical illumination' of facial image, where the effects of the unknown illumination conditions are minimized.

The quality of the recovery should be assessed at least using a simple, pixelwise evaluation, and, if possible, in conjunction with some face identification or face verification system.

Impact of the 3D model accuracy on results achieved, and possibility to use only one universal 3D model for all faces, should be also investigated.

Contents

Abstract

In this work a method for recovery of a canonical illumination of a facial image is presented, assuming that a 3D model of the face is available.

Having an arbitrarily illumined facial image, and a 3D model of the face available, the global scene illumination characteristics are estimated so that the 3D model is rendered to an image as similar to the to-be-compensated image as possible. The facial image is then compensated for the estimated illumination, to appear like an image acquired under frontal or omnidirectional illumination.

The recovery quality is evaluated using a simple eigen-face recognition system, where, on a test set of illuminated facial images, the recognition rate increases from 33% for original images to 94% for compensated images.

Experiments are made to investigate dependency of the compensation results on the 3D model precission. The possibility to substitute all individual facial models with a generic one is also studied.

1 Introduction

Computer vision systems processing facial images must often solve the problem of unknown environmental conditions. Images of faces acquired in real-time applications usually vary in face position, orientation, facial expression and face illumination.

The lighting variability includes not only intensity, but also direction and number of light sources. As is evident from figure 1, the same person, with the same facial expression and seen from the same viewpoint, can appear dramatically different when light sources illuminate the face from different directions.

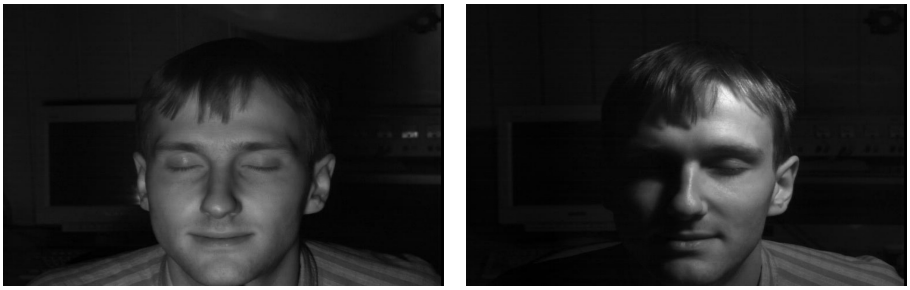

Figure 1: The same person seen under different lighting conditions can appear significantly different: In the left image the dominant light source is nearly frontal, in the right image, the dominant light source is above and to the right.

As a typical application, let us consider a face recognition system. Given a database of face images labeled with person's identity (the learning set), the task is then, for a facial image acquired in different environmental conditions,

to identify the person in the image using the image database. For a survey on face recognition methods see [CWS95], [SI92].

Variations in illumination can be addressed in two ways. First, the image database may contain several images of the individuals, taken under different illumination conditions. Under some idealization of the face surface, all images of the same face taken from the same viewpoint but under varying illumination forms a three-dimensional linear subspace in the image space (see [BHK97]). Search for the most similar face image in the database then becomes a search for the closest linear subspace. Or just simply for the closest image if the coverage of different illuminations of every single face is dense enough.

The second possibility stands for evaluators independent on illumination conditions. So each face is in the database stored only once. Either the face description is built invariant to changes in the illumination, ie. using only shape feature descriptors, or the acquired facial image is somehow compensated for the changes in the lighting before evaluation.

In this work, the problem of compensation of a facial image for an unknown illumination is addressed, assuming that in addition to 2D images of individual faces, also 3D models are available. Variations in face pose and expressions are not considered at all. Furthermore, an assumption has been made that the face had been located and aligned within the image, as there are numerous methods for finding faces in scenes.

To compensate the illumination influence, illumination characteristics are first approximated. This is done minimizing the difference between the acquired image and an image rendered from a textured 3D model of the face, while varying the 3D scene illumination. Then, the acquired image is compensated according to the best illumination approximation found.

Section 2 gives a description of 3D facial models, and of two 3D acquisition systems used to obtain the models. Next sections report methods for determination of the illumination characteristics, for the compensation for the approximated illumination, and for rendering of the 3D models. Sections 7 and 8 describe the eigen-face recognition system and the face database used to qualitatively evaluate the effect of the illumination compensation. Finally, conclusions and suggestions for future work are given.

2 3D Models

Two different systems are used to obtain the 3D models. For further use, models from both systems are converted to a common file format, which is described in appendix A.1.

2.1 Quadrifocal Stereo Vision System (QFSVS) Models

Some of the 3D models were acquired using a quadrifocal stereo vision system (QFSVS) developed at Center for Machine Perception (CMP), Czech Technical

University (CTU) Prague. This system is interoperated into the Matlab programming environment [Matlab], and as it is in development stage, 3D data are provided as Matlab variables. The following information is provided:

- $matrix[v][3]$ - 3D coordinates of mesh vertices
- $matrix[v][3]$ - surface normals at each vertex
- $matrix[t][3]$ - triangles, as indices of vertices forming the triangle
- $4 * matrix[3][4]$ - four projection matrices, one for each camera
- $4 * matrix[x][y]$ - four photo images, one from each camera

where v stands for number the of vertices, t for the number of triangles, and x and y are the dimensions of images captured by the cameras.

To obtain a texture for the model, the same camera system is exploited. Choosing the image from one of the cameras as the texture, texture coordinates for each vertex are computed transforming the vertex's 3D coordinates with the projection matrix of the selected camera.

2.2 Orientation and Correction of QFSVS Input Data

Data provided by the QFSVS 3D reconstruction system is just an unorganized set of triangles. Before further use, two steps should be performed:

1. Orientation of triangles, so all of them are oriented the same way (either clockwise, or counterclockwise).
2. Removal of such triangles, that causes the surface to be physically nonsensical. Moebius strip is an example of such a pathological structure which can occur.

Example of an original model (without texture) is on figure 2.

All the triangles are traversed by an algorithm, which in a breadth-first-search way assigns orientation to those triangles, for which the orientation was not set yet. As the 3D mesh can consist of multiple isolated parts, all of these parts must be processed. The algorithm runs as follows:

While there is any not-yet-oriented triangle (thus any not yet oriented isolated part of the mesh) do:

1. Pick any not-yet-oriented triangle
2. Assign it the desired orientation (clockwise or counter-clockwise)
3. Remember it as the only member in the list of open triangles
4. While there is any triangle in the list of open triangles do:

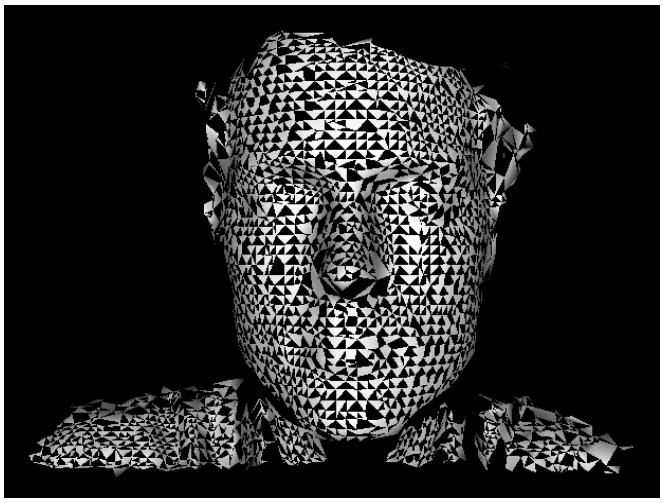

Figure 2: Original QFSVS 3D model, visibility of a triangle depends on its orientation

(a) pop triangle T from the beginning of the list

(b) find all triangles from the mesh neighboring with T

(c) for each of them, three possibilities can occur:

i. The triangle has no orientation assigned yet. It is then oriented in accordance with T, and then put at the end of the list of open vertices.

ii. The triangle is oriented, and its orientation is conforming with the orientation of T. It means that the triangle was already processed before T. No action is taken then.

iii. The triangle is oriented in contrary to the orientation of T. As this can not occur on correct surface, it means structure similar to Moebius strip occur, and the mesh must be broken at this place. That is why T is discarded from the mesh.

An example of an oriented model without texture is in figure 3, an example of a texture for the 3D model is in fig. 4.

2.3 3D Model from the ShapeSnatcher System

Another system used to provide the 3D models was the ShapeSnatcher, developed by Eyetronics Inc. This system uses only one camera to acquire an image of a scene where a regular square grid was projected from a slightly different angle. From the deformation of the grid observed by the camera, the 3D shape is reconstructed. Figure 5 shows an example of such a scene. Texture for the

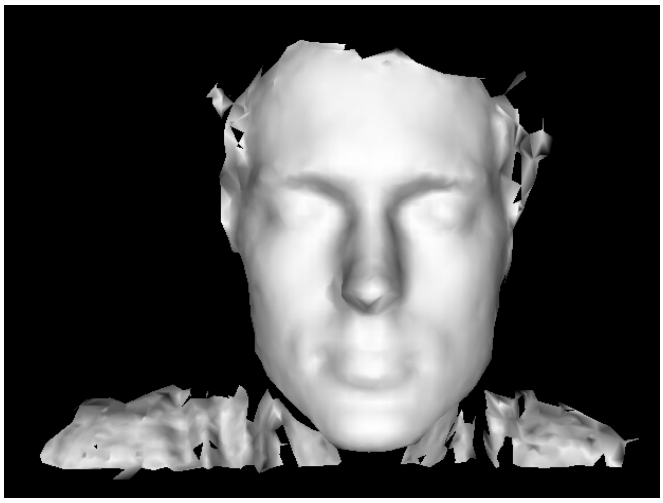

Figure 3: An oriented QFSVS 3D model

3D model is obtained from the same image, removing the grid. The result of the 3D reconstruction is the texture along with an two dimensional array of vertices placed on the nodes of the grid. For each vertex V of the grid following information is provided:

- 3D coordinates $\begin{pmatrix} x & y & z \end{pmatrix}$ of V
- texture coordinates $\begin{pmatrix} x_{tex} & y_{tex} \end{pmatrix}$ of V, ie. coordinates of the grid node corresponding to V in the image in fig. 5
- identifiers of the four neighboring vertices (N, E, W, S) on the grid, see figure 6.

Information missing but required for further processing, that is to be approximated:

- normal vectors at vertices, necessary for Gouraud shading (see section 6).
- projection matrix mapping the 3D coordinates to the texture, necessary for registration of the 3D model to the acquired image (see section 3).

Vertex normals are computed as an average of normals of the neighboring triangles. According to figure 6, a normal of triangle NEV is computed as a cross-product of vectors $N - V$ and $E - V$. Similarly for triangles ESV, SWV, and WNV. The V's normal is then the sum of all four these normals (if V lies on the 3D mesh boundary, there are less of them), normalized to unit length.

Calculation of the model's projection matrix is a task similar to the registration of the 3D model to the acquired image (see section 3). The sought projection matrix in homogenous coordinate system has four rows, three columns

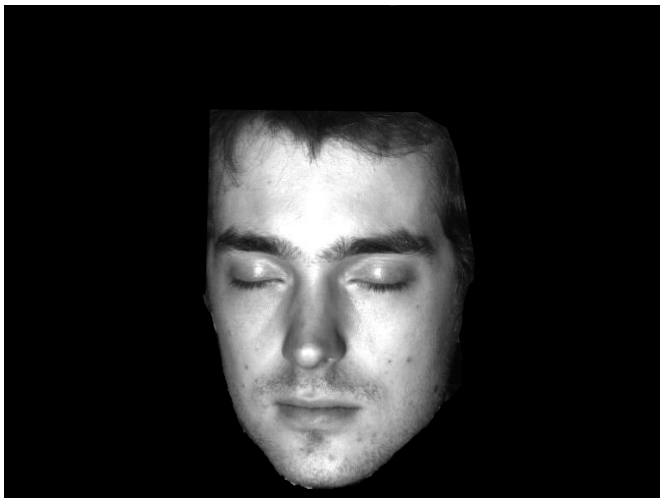

Figure 4: A texture for a QFSVS 3D model

and contains eleven independent variables (see eq. 2). The model consists of several thousands of vertices (typically $3000 - 5000$), for each of them the texture coordinates are known. Setting two linear equations for each vertex, a very overdetermined set of several thousands of equations of eleven variables is constructed (see equation 3). The approximate solution, which minimalizes the overall square error, is then used to form the projection matrix. See section 3 for detailed explanation.

3 Registration of 3D Shape to 2D Image

To estimate the illumination characteristics, and later to eliminate the illumination influence, it is necessary to render the 3D shape to the coordinate system of the acquired image; both the rendered and the acquired face should occupy the same region of the 2D image. In other words, the projection matrix used to render the 3D shape should be determined from the size, position and orientation of the face on the acquired image. Using homogenous coordinates, the equation

$$\begin{pmatrix} x & y & z & w \end{pmatrix} \cdot \mathbf{P} = \begin{pmatrix} x' & y' & w' \end{pmatrix} \tag{1}$$

should be ideally valid for each point on the face, which on the 3D shape has coordinates $\begin{pmatrix} \frac{x}{w} & \frac{y}{w} & \frac{z}{w} \end{pmatrix}$, and on the acquired image is projected to image coordinates $\begin{pmatrix} \frac{x'}{w'} & \frac{y'}{w'} \end{pmatrix} = \begin{pmatrix} x'' & y'' \end{pmatrix}$. $\mathbf{P}$ is a projection matrix of four rows

Figure 5: An example of a scene figuring as an input to the ShapeSnatcher reconstruction system

and three columns, having eleven independent variables

$$\mathbf{P} = \begin{pmatrix} a & e & i \\ b & f & j \\ c & g & k \\ d & h & 1 \end{pmatrix} \tag{2}$$

For each point on the face, following equations are set (let's assume that w equals to 1)

$$x' = ax + by + cz + d$$
$$y' = ex + fy + gz + h$$
$$w' = ix + jy + kz + 1$$

so

$$x'' = \frac{x'}{w'} = \frac{ax + by + cz + d}{ix + jy + kz + 1}$$
$$y'' = \frac{y'}{w'} = \frac{ex + fy + gz + h}{ix + jy + kz + 1}$$

then

$$ax + by + cz + d - ixx'' - jyx'' - kzx'' = x''$$
$$ex + fy + gz + h - ixy'' - jyy'' - kzy'' = y''.$$

This forms a set of two linear equations of eleven variables for each point. To determine all eleven variables, five and a half point is necessary (only one coordinate, x or y, of the sixth point on the acquired image).

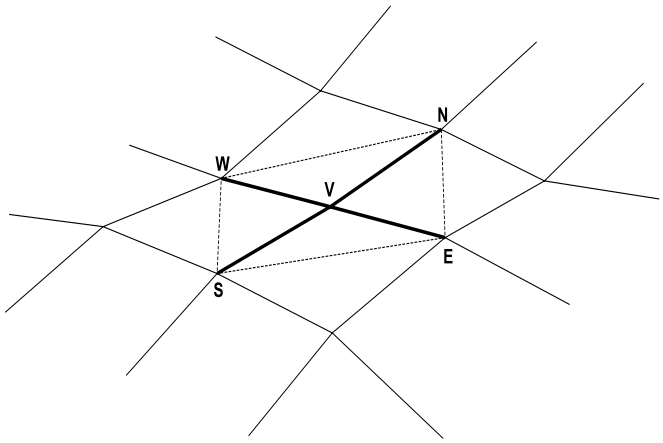

Figure 6: Structure of 3D mesh provided by the ShapeSnatcher

In matrix notation,

$$\begin{pmatrix} x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -x_1x_1'' & -y_1x_1'' & -z_1x_1'' \\ 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & -x_1y_1'' & -y_1y_1'' & -z_1y_1'' \\ x_2 & y_2 & z_2 & 1 & 0 & 0 & 0 & 0 & -x_2x_2'' & -y_2x_2'' & -z_2x_2'' \\ 0 & 0 & 0 & 0 & x_2 & y_2 & z_2 & 1 & -x_2y_2'' & -y_2y_2'' & -z_2y_2'' \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \\ j \\ k \end{pmatrix} =$$

$$= \mathbf{Q} \cdot \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \\ j \\ k \end{pmatrix} = \begin{pmatrix} x_1'' \\ y_1'' \\ x_2'' \\ y_2'' \\ \vdots \end{pmatrix} \tag{3}$$

so the vector of variables

$$\begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \\ j \\ k \end{pmatrix} = \mathbf{Q}^{-1} \cdot \begin{pmatrix} x_1'' \\ y_1'' \\ x_2'' \\ y_2'' \\ \vdots \end{pmatrix} \tag{4}$$

To increase the stability of the solution, more than eleven equations could be used, so the equation set will become overdetermined. The solution sought is then the one, which is the best approximation of the exact solution in the least-squares sense.

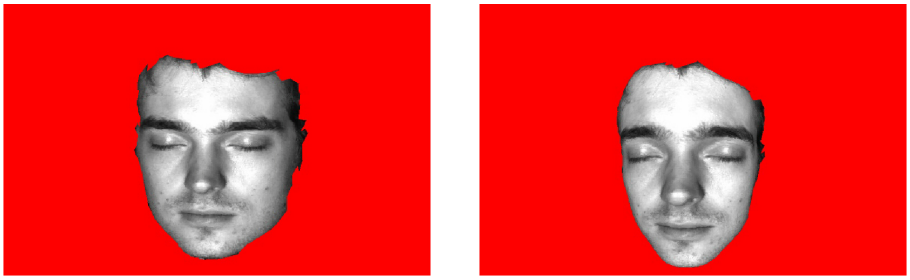

Figure 7: The original face and an image rendered from the 3D model using a projection computed from seven feature points

Unfortunately, the set of linear equations tends to be ill-conditioned, even for more feature points. The reason is that almost all easily specifiable points on the face lay in a plane, so the matrix $\mathbf{Q}$ is close to singular. On figure 7 is shown a typical 3D model rendered using a projection matrix found from seven feature points (four for the eye corners, two for the mouth corners and one for the nose tip). Left image shows the original face, right the rendered image of the same face. The right image seems to be of a different person. Such a behavior is definitively not desirable within a system aimed at face processing tasks.

There are two possible ways out of this problem:

- Increase of the number of the feature points.
- Decrease of the number of the independent variables in the projection matrix.

In general, it is complicated to define more than seven feature points on a face, such that they would be easily and precisely detectable. Even in the feature point set mentioned above, mouth corners are not very suitable as their position vary with facial expressions, and the tip of the nose cannot be positioned very precisely.

So it leaves only the possibility to decrease the variability of the projection matrix. Indeed, all projection matrices possibly encountered in practice would not span uniformly across the whole eleven-dimensional space, but would group together in only a less-dimensional subspace.

The subspace can be bound setting some of the matrix elements to a constant, likely to zero, so the matrix might look like

$$\mathbf{P} = \begin{pmatrix} a & e & 0 \\ b & f & 0 \\ 0 & 0 & k \\ d & h & 1 \end{pmatrix} \tag{5}$$

where only seven independent variables occur. Such a projection would work quite well for 3D shapes aligned so that the z-axis is parallel with the camera-to-screen direction, thus the z-coordinate having role only in the perspective distortion, while the x- and y-cordinates not affecting it. Unfortunately, there is no presumption of such an alignment.

On the other hand, each 3D model comes along with a projection matrix used to map 3D coordinates to the texture coordinates (if not explicitly expressed, can be deduced, see section 2.3). So as long as the texture for the 3D model is obtained with similar camera as the images for compensation will be, this projection matrix can be used as an origin of the less-dimensional subspace of projections. The desired projection matrix is than computed

$$\mathbf{P} = \mathbf{P_t} \cdot \mathbf{T} \tag{6}$$

where $\mathbf{P_t}$ is the projection matrix 4×3 mapping 3D homogenous coordinates of the 3D model to its texture, and $\mathbf{T}$ is a 3×3 matrix standing for a 2D (in homogenous coordinate system) transformation between texture and the acquired image. If the transformation is constrained to be affine, $\mathbf{T}$ would be in form

$$\mathbf{T} = \begin{pmatrix} a & d & 0 \\ b & e & 0 \\ c & f & 1 \end{pmatrix} \tag{7}$$

containing only six variables. Affine transformation comprehends most of the possible alterations, as long as the camera acquiring the image for compensation has similar optical properties as the camera which acquired the texture. These alterations include changes in the face size, face rotations (in tilt sense), different pixel aspect ratios, and different face locations within the image. Moreover, this approach requires location of feature points only on 2D images (on the texture, and on the acquired image), not on the 3D shape, as it was the case of the previous approach. The problem of extraction of feature points from facial

images is well described in literature, still, in the current implementation, these points are identified manually.

The transformation $\mathbf{T}$ maps texture coordinates to the image coordinates

$$\begin{pmatrix} x & y & 1 \end{pmatrix} \cdot \mathbf{T} = \begin{pmatrix} x' & y' & w' \end{pmatrix} \tag{8}$$

where each point $\begin{pmatrix} x & y \end{pmatrix}$ on the texture transforms to the point $\begin{pmatrix} x'' & y'' \end{pmatrix} = \begin{pmatrix} \frac{x'}{w'} & \frac{y'}{w'} \end{pmatrix}$ on the acquired image. So, from the equation 7

$$x' = ax + by + c$$

$$y' = dx + ey + f$$

$$w' = 1$$

For each point, two linear equations are set:

$$x'' = \frac{x'}{w'} = ax + by + c$$

$$y'' = \frac{y'}{w'} = dx + ey + f$$

With six variables, six independent equations are necessary, which requires three feature points which do not lie in one line to be located. Again, for more stable solution more than three points can be used, and the solution of the resulting overdetermined equation set is found as the one with the least-square error.

To summarize, the mapping of the 3D shape to the acquired image is represented by a 4×3 projection matrix $\mathbf{P}$ in homogenous coordinate system. $\mathbf{P}$ is computed according to equation 6, $\mathbf{P_t}$ being the matrix of the known projection from the 3D shape to its texture, and $\mathbf{T}$ being a computed 2D transformation matrix between the texture and the acquired image.

4 Determination of Illumination Characteristics

With a textured 3D model of a face taken in known environmental conditions, and another image I_a acquired in unknown environment, the task is to find an approximation of light sources affecting the image I_a.

The principle is simple. If the 3D model is rendered so that the resulting image $\bar{I}_a$ is as similar to I_a as possible, then the configuration of light sources used for the rendering can be denoted as the best approximation of the unknown illumination.

4.1 Illumination Model

The appropriate illumination model must be chosen first. For the sake of simplicity, three assumptions were made:

- **Assumption 1.** All the light sources are infinitively far away, so the light rays from one source are mutually parallel.
- **Assumption 2.** The face surface is Lambertian, ie. the intensity of the reflected light depends only on the albedo of the surface (the texture), and cosine of the angle between the light direction and the surface normal. No specular reflection is considered.
- **Assumption 3.** There is no self-shadowing on the face shape, ie. all light sources affect entire face surface.

These assumptions lead to a simplification, that all light sources can be substituted with only one.

The used lighting model is very simple. Intensity I of each pixel is computed:

$$I = (A + L\delta cos\omega) * \Lambda \tag{9}$$

where I represents the emitted intensity, A is the ambient energy, δ is 1 if the vertex is visible from the light source and 0 otherwise, ω is the angle between the incident light direction and the surface normal vector, Λ is the surface albedo, and L is the intensity of the directional light. See figure 8 where the situation is depicted. Would the third assumption be always fulfilled, δ will be always equal to one.

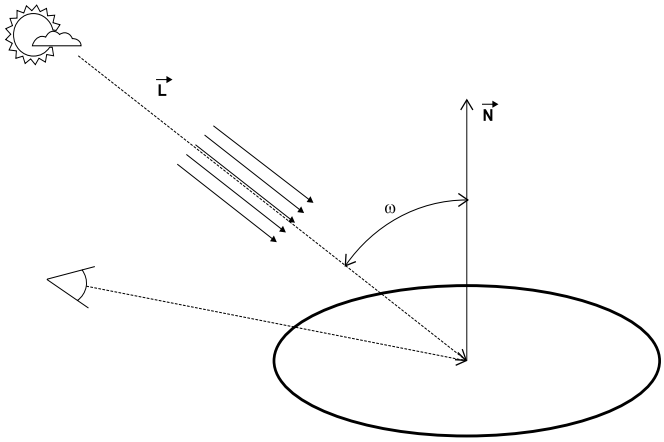

Figure 8: Scheme of the illumination model

Surface albedo Λ is given by the texture, and surface normals are specified within the 3D model. δ and ω depend on the light direction and surface normals. Remaining variables in the model, the ambient and directional light intensities, are free. Thus the rendering of the face can be parametrized by four scalar values, two for the intensities of ambient and directional illumination, and two for the directional light direction. In current implementation, the light direction is

given in spherical coordinates, parametrized by two angles, φ being the rotation around the y axis, and ψ being the rotation around the z axis (see fig. 9). To be represented as a three-dimensional vector $\begin{pmatrix} x & y & z \end{pmatrix}$, individual components are computed this way:

$$x = \sin(\varphi) * \cos(\psi)$$

$$y = \sin(\psi)$$

$$z = \cos(\varphi) * \cos(\psi)$$

Setting the light intensity L to zero and the ambient intensity A to one, the rendered image would be identical to the texture of face model if observed from front.

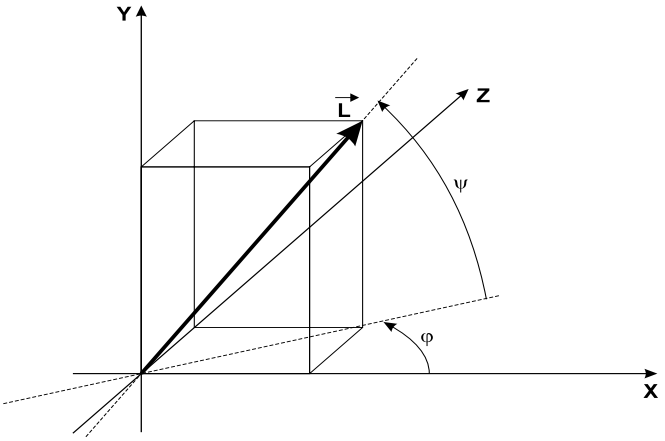

Figure 9: Parametrization of the light direction

4.2 Cost Function

To find the best approximation of illumination, the quality of the approximation must be evaluated.

First, the 3D textured model is rendered into image $\bar{I}_a$, rendering parametrized by the four variables described above. Having the acquired image I_a as a reference image, the approximation quality is determined by pixel-wise differentiation between images I_a and $\bar{I}_a$. Only pixels rendered from the 3D model, ie. these having color other than background on $\bar{I}_a$, are considered.

This can be expressed as a scalar function of four variables, representing the four rendering parameters, function value being the approximation error. The function is then provided to a minimization algorithm as a cost function. Finding the minimum of the function will supply the rendering parameters of the best estimation of the illumination.

4.3 Optimization Algorithm

The algorithm used is the Nelder-Mead simplex search described in [Simplex] or [NR92]. It is a direct search method that does not require gradients or other derivative information. If n is the dimension of the search space, four in this task, a simplex is characterized by the $n + 1$ distinct vectors which are its vertices. In two-space, a simplex is a triangle; in three-space, it is a pyramid, etc. At each step of the search, a new point in, or near the current simplex is generated. The function value at the new point is compared with the function values at the vertices of the simplex and, usually, one of the vertices is replaced by the new point, giving a new simplex. This step is repeated until the diameter of the simplex is less than a specified tolerance and the function values of the simplex vertices differ from the lowest function value by less than another specified tolerance.

It is not very efficient method in terms of the number of function evaluations that it requires. That is the effect of lack of information about the function derivatives. But only about 100 function evaluations are usually required to reach the minimum.

5 Compensation for the Detected Illumination

Once the best approximation of the illumination is known, the acquired image can be compensated for it.

The 3D model is rendered twice more, only without texturing, ie. rendered according to the equation

$$I = (A + L\delta cos\omega) \tag{10}$$

The first of the rendered images, $I_{detected}$, is rendered with parameters of the best approximation, while the second, $I_{desired}$, with parameters corresponding to the desired illumination of the face. For example, setting the ambient intensity to 1 and the directional light intensity to 0 will produce image evenly illuminated (see fig. 10), or setting the ambient intensity to 0, the directional light intensity to 1 and the light direction to be along the z axis will generate image with only a frontal illumination (see fig. 11).

To obtain a compensated image I_c, intensity of each pixel i in the acquired image I_a is multiplied by the ratio of intensities of corresponding pixels in $I_{desired}$ and $I_{detected}$.

$$I_c(i) = I_a(i) * I_{desired}(i)/I_{detected}(i)$$

Samples of compensated images are shown in bottom rows of images in figures 34 and 35.

Figure 10: 3D model without texture rendered with $A = 1$ and $L = 0$

5.1 Cost Function Revised

While finding the best approximation of the illumination, the impact of the error on the approximation quality was constant all over the face. During compensation, increasing the intensity of dark parts of the image I_a will proportionally increase the intensity errors in these parts. For example, let's look at the acquired image lighted from the right side (see fig 12). The intensity map along a horizontal scanline could look like the middle red line in fig. 13a. Let the surrounding lines depict the error range, where intensities of the approximated image would lay. After illumination compensation, when the dark areas of the image were brightened, the error range is expanded (see fig. 13b), yielding more inaccurate results.

Therefore, dark areas of the acquired image should be approximated more precisely than bright areas. To address this, the evaluation of the cost function expressing the approximation error is done this way:

1. image $\bar{I}_a$ is rendered from the textured 3D model according to the rendering parameters (parameters of the cost function)
2. image $I_{detected}$ is rendered with the same parameters, but without texture
3. image $I_{desired}$ is rendered with parameters corresponding to the desired illumination, again without texture
4. The error accumulator is cleared,
   $error = 0$
5. For each non-background pixel i in the rendered image do:

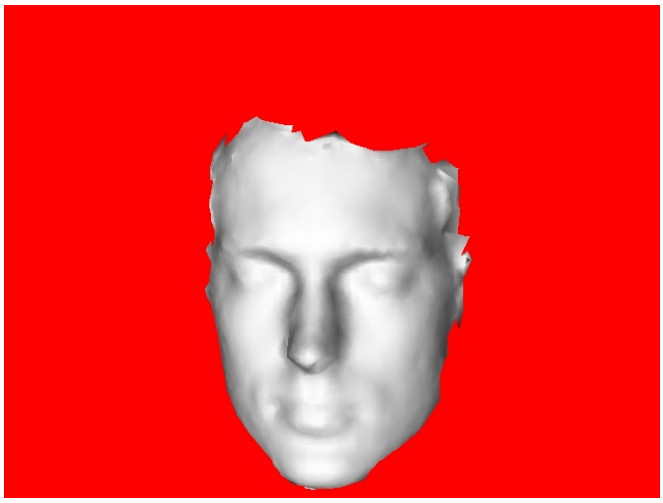

Figure 11: 3D model without texture rendered with $A = 0$ and $L = 1$

   (a) compute the difference between the acquired and the approximated image
   $diff = abs(I_a(i) - \bar{I}_a(i))$
   (b) multiply the difference by the same ratio that would be used later in the compensation phase
   $diff = diff * I_{desired}(i)/I_{detected}(i)$
   (c) increase the square error,
   $error = error + diff^2$

6. The error is normalized,
   $error = error/number\ of\ pixels\ in\ the\ rendered\ image$

7. *error* represents the approximation quality, and is returned as a value of the cost function

So the error of individual pixel is weighted proportionally to the amount of later brightening of the pixel.

6 Rendering

The 3D model is rendered by the Gouraud shading method, triangle by triangle. Each triangle T is rendered in following steps:

1. 3D coordinates $\begin{pmatrix} x & y & z \end{pmatrix}$ of each of the three triangle's vertices are transformed to 2D image coordinates $\begin{pmatrix} x_{im} & y_{im} \end{pmatrix}$ by a projection matrix (see section 3 for how it is obtained).

Figure 12: Sample acquired image

2. Intensity of the light reflected from each vertex is computed according to equation

$$I = (A + L\delta cos\omega) \tag{11}$$

ie. the equation 9 of the illumination model, with the surface albedo ignored.

3. For each pixel p of the rendered image, occupied by triangle T, following values are linearly interpolated from values known only for vertices:

   - intensity I computed in equation 11
   - texture coordinates x_{tex} and y_{tex}
   - 3D coordinates of the point on the model's surface represented by pixel p

4. The final color of pixel p is then given as a multiplication of the illumination intensity I and the texture albedo at pixel's texture coordinates $\begin{pmatrix} x_{tex} & y_{tex} \end{pmatrix}$. The z component of the 3D coordinates associated with the pixel is used as the Z-buffer, ie. if there was, on the image coordinates $\begin{pmatrix} x_{im} & y_{im} \end{pmatrix}$, previously rendered any pixel with larger z value, pixel p is not rendered at all. In addition, each pixel has associated an unique identifier of the triangle from which it was rendered. If the model is rendered with shadows, the remaining components of the 3D coordinates (x and y) are also exploited (see section 6.2).

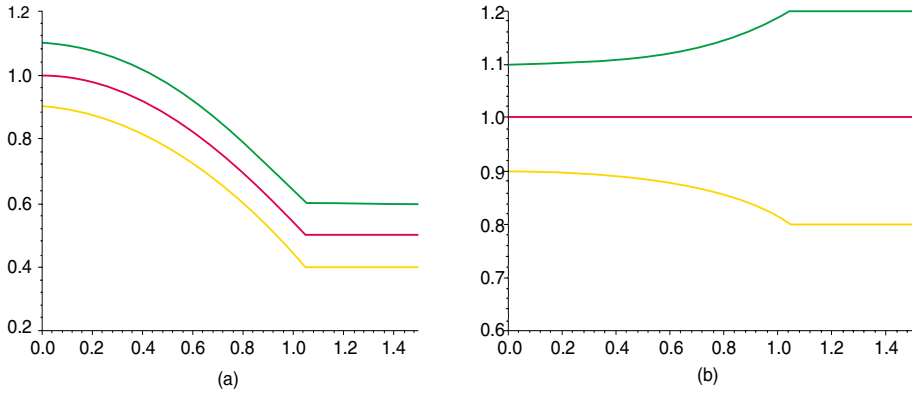

Figure 13: image intensities and error ranges before compensation (a), and after compensation (b)

6.1 Shadows

The existence of shadows is in contrary to **Assumption 3**, and their consideration assumes that the acquired image I_a was influenced by only one light source, ie. not by several lights approximated by the only one.

Shadows can, however, spread over a significant part of the face, especially shadows from the nose (if illumined from side) or from the brows (if illumined from top). The effect of not accounting for shadows is twofold. First, the shadowed parts of the acquired image are not compensated correctly. As the model predicts an illumined area where there is a shadow on the image, the area will remain dark after compensation. Second, more grievous consequence, is a deformation of the cost function, so the best illumination approximation found is inaccurate, and, depending on the image, the optimization process might converge to another local minimum.

In figure 14 is an example of an image acquired with a light coming from right, and there is small, but clearly distinguishable shadow cast by the nose toward the left eye. On figure 15 the best illumination approximation for this image is shown, when it is rendered without shadow. Based on this approximation, the compensation result is shown in figure 16. The only corrected part of the shadowed area is where the surface normal directs away from the incident light.

6.2 Rendering Shadows

Because the rendering takes place in every evaluation of the cost function, the shadows are to be computed as quickly as possible. The implemented method first renders those triangles, that are either completely in the shadow, or completely out of the shadow. The remaining triangles, ie. those only partially

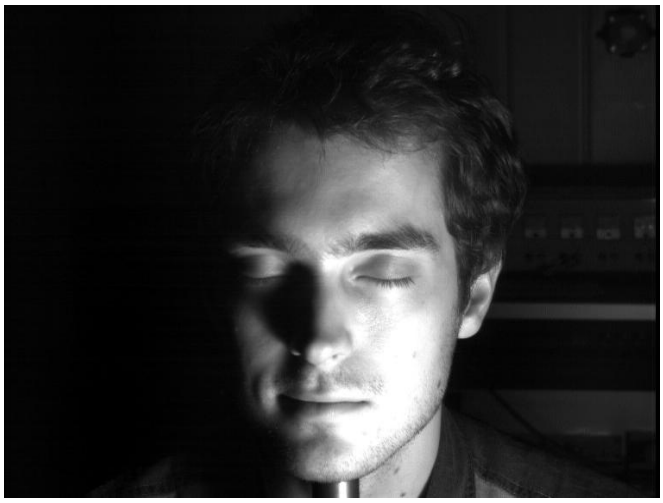

Figure 14: An acquired image with a shadow of the nose

shadowed, are rendered pixel by pixel, all the time checking whether the pixel lies in the shadow or not. The algorithm runs in following steps:

1. **The 3D model is rotated** so that the z-axis is aligned with the light direction, given by two angles, φ and ψ (see fig. 9). The coordinates are transformed with matrix

$$\mathbf{T_L} = \mathbf{Rot_y}(\varphi) \cdot \mathbf{Rot_x}(\psi) =$$
$$= \begin{pmatrix} cos(\varphi) & 0 & -sin(\varphi) & 0 \\ 0 & 1 & 0 & 0 \\ sin(\varphi) & 0 & cos(\varphi) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\psi) & sin(\psi) & 0 \\ 0 & -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (12)$$

   For each vertex the new coordinates are

$$\begin{pmatrix} x_t & y_t & z_t \end{pmatrix} = \begin{pmatrix} \frac{x'}{w'} & \frac{y'}{w'} & \frac{z'}{w'} \end{pmatrix}$$

   where, if the original vertex's coordinates are $\begin{pmatrix} x & y & z \end{pmatrix}$,

$$\begin{pmatrix} x' & y' & z' & w' \end{pmatrix} = \begin{pmatrix} x & y & z & 1 \end{pmatrix} \cdot \mathbf{T_L} \quad (13)$$

2. **The rotated model is rendered.** Size of the image, into which is the model rendered, can be chosen arbitrarily. Larger size increases precision, smaller size increases speed. Currently, the image size is set to be 500×500 pixels. The projection matrix transforming the model's 3D coordinates to the 2D image coordinates is calculated so that the face spans over as large

Figure 15: Approximation of the illumination in fig. 14, ignoring shadows

area of the image as possible. The light is assumed to come in parallel rays, supposedly from an infinitively distant light source (**Assumption 1**), so the projection matrix is to represent a parallel projection with an appropriate scale and translation. First, the bounding box of the rotated model is computed, in $x_t y_t$ plane only. The z_t coordinate does not affect the image appearance, and is used for the Z-buffer. Let the x_{min}, x_{max}, y_{min}, and y_{max} stand for the minimal resp. the maximal coordinate of the rotated model on the x, resp. the y axis, and W and H stand for the size of the rendered image (by choice, both equals to 500). The projection matrix is then

$$\mathbf{P_L} = \begin{pmatrix} \frac{W}{maxX-minX} & 0 & 0 \\ 0 & \frac{H}{maxY-minY} & 0 \\ 0 & 0 & 0 \\ -minX\frac{W}{maxX-minX} & -minY\frac{H}{maxY-minY} & 1 \end{pmatrix} \tag{14}$$

scaling the model to fit the image in size, with a translation to move the projection of $\begin{pmatrix} minX & minY \end{pmatrix}$ to the image coordinates $\begin{pmatrix} 0 & 0 \end{pmatrix}$. An example of an image rendered this way is in fig. 17, for the same light configuration as in the figures 14, 15, and 16.

3. **Each vertex is checked whether it is shadowed.** During the rendering, unique identifier of currently rendered triangle is assigned to each pixel written to the image (see section 6). To determine whether a vertex V is in the shadow, the triangle identifier is picked up from the just rendered image of the rotated 3D model, from the pixel position where V was

Figure 16: Compensation of fig. 14, ignoring shadows

projected to. If V is one of the triangle's vertices, it was not overpainted by another triangle, so it is visible from the light direction, and thus not shadowed.

4. **Triangle overlap map is generated.** A special triangle overlap map is generated during the rendering from the light direction, which for every pair of triangles holds the information whether they both share at least one pixel of the rendered image. So for every triangle T, it enumerates only such triangles, that might cast shadow over T, or T might cast shadow over them. The map is implemented as a two-dimensional array of $number\_of\_triangles \times number\_of\_triangles$ booleans. The indication is set whenever a pixel of a triangle is to be rendered over a previously rendered pixel (whether it is really written depends on the Z-buffer look-up). Again, information of the triangle's identifier associated with a rendered pixel is exploited.

5. **The image from the camera's view is finally rendered.** The image is rendered in two steps. First, only those triangles that are completely in shadow (all three of their vertices were marked as shadowed in step 3), or completely lit (all three of their vertices were marked as lit) are rendered. See fig. 18, where only these triangles are displayed. The decision based only on shadowing of the vertices is not exact (see fig. 19), but as on the face shape do not occur sharp edges, the error can be neglected. In the second step, the remaining, partially shadowed, triangles are rendered. The check for the shadow is performed for every individual pixel of the triangle. While rendering the pixel p, the three dimensional coordinates of point P of the shape surface represented by p are interpolated (see

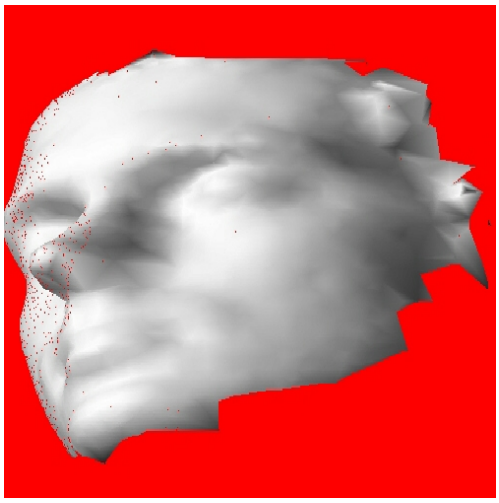

Figure 17: A 3D model rendered from the light direction, dots represent positions of hidden (shadowed) vertices.

section 6). The triangle overlap map says, which of the triangles might cast shadow over point P, and only these are checked whether they do so. To check it, P is rotated to the coordinate system of the light, using the transformation $\mathbf{T}_L$ from the equation 12. Then, if the 2D projection of the point to the $x_t y_t$ plane lays inside any of the triangles from the triangle overlap map, these again projected to the $x_t y_t$ plane, and the triangle is nearer to the light source (according to the rotated z_t coordinate), the point is shadowed by the triangle. Pixel p is then rendered only using the ambient component of the light ($\delta = 0$ in the illumination model equation 9). If there is no triangle placed over the rotated point P, pixel p is rendered using both ambient and directional component of the light ($\delta = 1$).

6.3 Shadow Rendering Results

An example of the detected illumination, rendered including shadow, is shown in figure 20. Compensation based on this approximation is in figure 21. Two kinds of defect are observable. First, the shape of the shadow predicted from the 3D model shape does not match exactly the real shadow. The consequence is, on figure 21, the extremely bright part of the left eye on the shadow boundary (here the 3D model predicted shadow where it was not), and the dark strip leading from the left corner of the left eye down and toward the nose (here the 3D model did not predict a shadow, but it was there). The second defect is caused by sharp edges of the modeled shadow. In real, shadows are soft

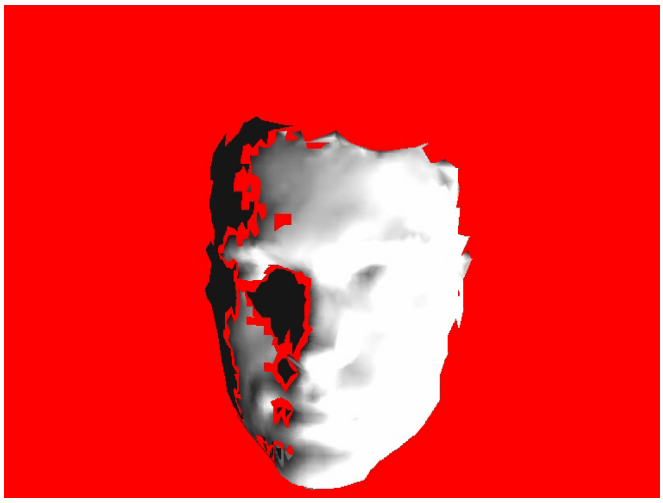

Figure 18: Only wholly lit or wholly shadowed triangles rendered

due to optical properties of the air. Modeling such soft shadows would require more advanced rendering techniques, which are not affordable for this task for their computational complexity. The easiest way to simulate the soft shadows is to blur the rendered image. Simple intensity averaging is used to do so. On figure 22 is an example of such a blurred illumination approximation, and on the figure 23 is the result of the sequent compensation. The effect of the mispredicted shadow shape, either caused by inexactness of the 3D model, or by inexact approximation of the illumination direction, still remains. The image in figure 23 shows the result of the whole compensation process of the image in figure 14.

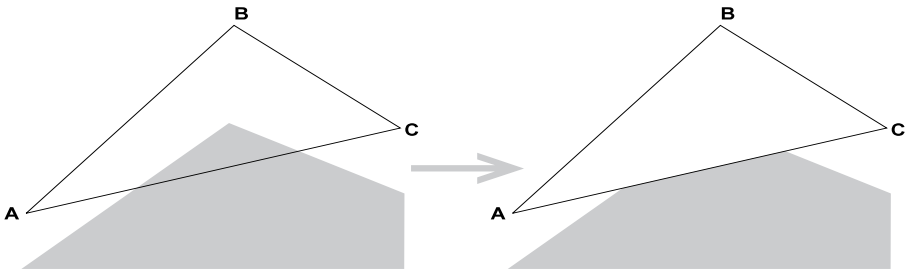

Figure 19: A situation, where all three triangle vertices are lit, but a shadow is still cast over the triangle (left). Such a triangle is wrongly rendered as if completely lit (right).

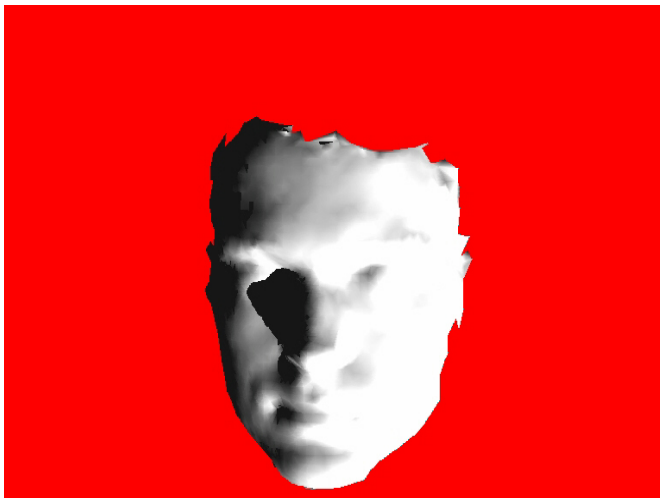

Figure 20: Illumination approximation rendered with shadows

7 Eigen-faces for Face Recognition

To evaluate the quality of the illumination compensation, an elemental face recognition system was implemented. The selected method – eigen-faces – was described by Turk and Pentland in [TP91]. The scheme is based on an information theory approach that decomposes face images into a small set of characteristic feature images called "eigen-faces", which may be thought of as the principal components of the initial training set of face images. Recognition is performed by projecting a new image onto the subspace spanned by the eigen-faces ("face space") and then classifying the face by comparing its position in the face space with the positions of known individuals.

In the language of information theory, the idea is to extract the relevant information in a face image, encode it as efficiently as possible, and compare one face encoding with a database of models encoded similarly. A simple approach to extracting the information contained in an image of face is to capture the variation in a collection of face images, independent of any judgement of facial features, and use this information to encode and compare individual face images.

In mathematical terms, the principal components of the distribution of faces, or the eigenvectors of the covariance matrix of the set of face images are found, treating the image as a point (or a vector) in a very high dimensional space. The eigen vectors are ordered, according to their associated eigenvalues, each one accounting for a different amount of the variation among the face images. These eigenvectors can be thought of as a set of facial features that together characterize the variation between face images.

Every individual face from the training set can be represented exactly in terms of a linear combination of the eigen-faces. Each face can also be ap-

Figure 21: Compensation based on approximation rendered with shadows

proximated using only the "best" eigen-faces — those that have the largest eigenvalues, and which therefore account for the biggest variance within the set of face images.

The system is initialized with following operations:

1. Acquire an initial set of M face images (the training set).
2. Calculate the eigen-faces from the training set, keeping only M' eigen-faces that correspond to the highest eigenvalues. These M' images define the face space.
3. Calculate the corresponding distribution in M'-dimensional weight space for each known individual, by projecting its face image(s) onto the face space.

Having the system initialized, following steps are used to recognize new face images:

1. Calculate a set of weights based on the input image and the M' eigen-faces by projecting the input image onto each of the eigen-faces.
2. Determine if the image is a face at all by checking to see if the image is sufficiently close to the face space.
3. If it is a face, classify the weight pattern as either a known person or as a unknown, according to its distance to the closest weight vector of a known person.

Figure 22: Blurred illumination approximation, rendered with shadows

7.1 Calculating Eigen-faces

Let a face image $I(x, y)$ be a two-dimensional X by Y array of intensity values. An image may also be considered as a vector of dimension $X \times Y$, so that a typical image of size 256 by 256 pixels become a vector of dimension $65,536$, or, equivalently, a point in $65,536$-dimensional space. An ensemble of images, then, maps to a collection of points in this huge space.

Images of faces, being similar in overall configuration, will not be randomly distributed in this huge image space, and thus can be described by a relatively low-dimensional subspace. The main idea of the principal component analysis (or Karhunen-Loeve expansion) is to find the vectors that best account for the distribution of face images within the entire image space. These vectors define the subspace of face images, called face space. Each vector is of length $X \times Y$, and is a linear combination of the original face images.

Let the training set of images be $\mathbf{\Gamma}_1, \mathbf{\Gamma}_2, \ldots, \mathbf{\Gamma}_M$. The average face of the set is defined by

$$\mathbf{\Psi} = \frac{1}{M} \sum_{n=1}^{M} \mathbf{\Gamma}_n$$

Each face differs from the average by vector

$$\mathbf{\Phi}_i = \mathbf{\Gamma}_i - \mathbf{\Psi}$$

This set of large vectors is then subject to principal component analysis, which seeks a set of M orthonormal vectors $\mathbf{u}_1 \ldots \mathbf{u}_M$, which best describes the distribution of the data. Vectors $\mathbf{u}_1 \ldots \mathbf{u}_M$ and scalars $\lambda_1 \ldots \lambda_M$ are the

Figure 23: Compensation based on blurred approximation. The noise in the brightened areas was present in the original image as well, only with a small magnitude. In this image, the noise is magnified by the illumination compensation

eigenvectors and eigenvalues, respectively, of the covariance matrix

$$\mathbf{C} = \frac{1}{M}\sum_{n=1}^{M} \mathbf{\Phi}_n \cdot \mathbf{\Phi}_n^T = \mathbf{A} \cdot \mathbf{A}^T$$

where the matrix $\mathbf{A} = [\mathbf{\Phi}_1, \mathbf{\Phi}_2, \ldots, \mathbf{\Phi}_M]$.

For description of the training set see section 8. The average face $\mathbf{\Psi}$ of this training set is shown in figure 24. Sixteen most significant eigen-faces (with highest eigenvalues) of the training set, and four least significant ones, are shown in figure 31.

To obtain a weight vector $\mathbf{\Omega}$ of contributions of individual eigen-faces to a facial image $\mathbf{\Gamma}$, the face image is transformed into its eigen-face components (projected onto the face space) by a simple operation

$$\omega_k = \mathbf{u}_k^T(\mathbf{\Gamma} - \mathbf{\Psi})$$

for $k = 1, \ldots, M'$, where $M' \leq M$ is the number of eigen-faces used for the recognition.

The weights form vector $\mathbf{\Omega} = [\omega_1, \omega_1, \ldots, \omega_{M'}]$ that describes the contribution of each eigen-face in representing the face image $\mathbf{\Gamma}$, treating the eigen-faces as a basis set for face images.

The original image might be reconstructed

$$\bar{\mathbf{\Gamma}} = \mathbf{\Psi} + \sum_{k=1}^{M'} \omega_k \mathbf{u}_k$$

Figure 24: Average face $\mathbf{\Psi}$ of the training set, corresponds to the origin of the face-space.

where $M' \leq M$ is the number of eigen-faces used. If $M' < M$, the reconstructed image is only approximation of the original face, if $M' = M$, the reconstruction is exact.

The impact of the number of eigen-faces on the reconstructed face appearance is demonstrated in figure 25. In the upper left corner, an original image from the face database is shown. Then, approximations of the face using different number of eigen-faces with highest eigenvalues are depicted. Using all eigen-faces (bottom right corner), the face reconstruction is identical to the original image, except for round-off errors.

7.2 Face Recognition

Provided an unknown facial image, the recognition process first involves transformation of the facial image to the position common for all faces from the training set. This transformation is based on a facial feature detection, and is described in section 8. Then, the image is projected onto the face space in the same way as the images from the training set (see section 7.1), resulting into a weight vector $\mathbf{\Omega}$.

Finally, the face most similar to the unknown one is sought in the training set. The simplest method for determining which face provides the best description of an unknown input facial image is to find the image k that minimizes the

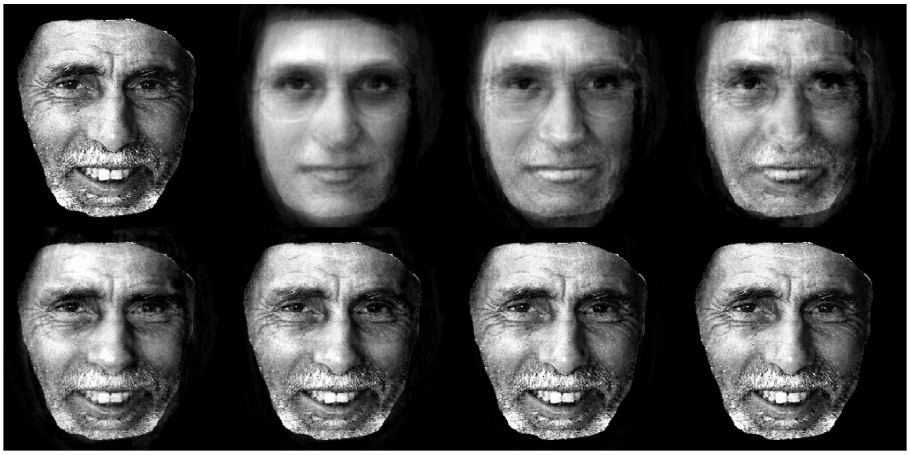

Figure 25: Reconstruction of a face image from the eigenfaces. From left to right, top to bottom: the original image, approximation using first 10, 20, 30, 40, 50, 75 eigen-faces, reconstruction using all the eigen-faces.

euclidean distance

$$\epsilon_k = ||(\mathbf{\Omega} - \mathbf{\Omega}_k)||^2$$

where $\mathbf{\Omega}_k$ is a weight vector describing the kth face from the training set. A face is classified as belonging to person k when the minimum ϵ_k is below some chosen treshold Θ_ϵ, otherwise, the face is classified as unknown.

Usually, implementations of this method use more than one image of each individual in the training set. Different images are taken under varying illumination conditions and under varying pose. The recognition becomes less dependent on these variations, since the acquired image is allowed to match any of the images of the person.

For use in this work, however, only one image of each individual, frontal view with frontal illumination, is present in the training set. The reason is that variations in face pose are not addressed at all, and variations in the illumination are what is to be compensated in this work. The eigen-faces method stands for evaluation of the compensation quality, so any improvements regarding the illumination variability are not desired.

8 Face Database

One hundred of facial images, originated from different sources, were used to initialize the eigen-face recognition system. All the images were first converted to grayscale images. To eliminate the effect of different background, from every image was the background manually removed, so that only the facial part remained. See fig. 26 for an example.

Figure 26: An original image from the face database, after background removal, and converted to grayscale

Being of different origin, the images vary in face size, position, orientation and intensity levels. For the eigen-face identification algorithm functionality, it is essential to have all the faces equally aligned in the image space. So all the images were transformed to a common eigen-face image space:

1. Dimension of the eigen-face images I_{ef} was chosen, currently to 256 by 256 pixels.

2. Seven feature points $p_1 \dots p_7$ were defined to be used for the face position alignment, four for the eye corners, two for the mouth corners and one for the nose tip.

3. On each of the 100 facial images, coordinates of all of the seven feature points $p_{o1} \dots p_{o7}$ were identified (currently, the identification is manual).

4. A desired position $p_{ef1} \dots p_{ef7}$ of these feature points was defined within the eigen-face image space.

5. An affine transformation in form

$$\mathbf{T} = \begin{pmatrix} a & d & 0 \\ b & e & 0 \\ c & f & 1 \end{pmatrix}$$

was found for each image such that the overdetermined equation set

$$\begin{array}{ccccc} p_{ef1} & \cdot & \mathbf{T} & = & p_{o1} \\ \vdots & & \vdots & & \vdots \\ p_{ef7} & \cdot & \mathbf{T} & = & p_{o7} \end{array}$$

was satisfied with the least square error.

6. Transformation $\mathbf{T}$ was then used to resample the original image. Intensity of every pixel p_{ef} in the eigen-face image I_{ef} was set to the intensity of pixel $p_o = p_{ef} \cdot \mathbf{T}$ in the original image, or, in the case when the p_o was outside the original image, to the intensity of the background.

The result was a facial image resampled to the eigen-face image space, geometrically aligned to the position given by feature points $p_{ef1} \ldots p_{ef7}$. One of the final images, as were used to compute the eigen-face subspace, is shown in figure 27. The whole face database is in fig. 32.

Figure 27: The face image after transformation. Images in this form are used to compute the eigen-faces.

9 Measurements

The quality of the compensation was evaluated using the eigen-face recognition system, as was described in section 7. 36 images of five different persons were

used for the measurements. The face database consisted of one hundred different faces (see section 8) to make the recognition task not so trivial as it would be with only five persons.

The compensation process assumed that the appropriate person's identity was known prior to the compensation, so his/hers 3D model could be used. This corresponds to tasks like face verification or evaluation of facial expressions, where the person's identity can be assumed. For the recognition, however, it would be necessary to compensate the illumination individually, according to the 3D model of every person in the database, before the recognition is applied.

The set of 36 images, which were to be recognized, was strongly influenced by illumination. All the images were taken under one directional light, varying in position and direction. On most of the images, only part of the face was perceptible. See figures 1, 12, 14, 34, and 35 for examples of such input images.

When these original images, ie. the images before the compensation, were provided as an input to the eigen-face recognition algorithm, the recognition rate was 33%; only 12 out of the 36 images were recognized correctly. After the illumination compensation, however, the recognition rate increased to 94%, ie. 34 out of the 36 images were recognized as of the correct person.

9.1 Using a Generic Face Model

The possibility to use only one generic 3D face model was investigated. The motivation was to allow tasks where the person's identity can't be assumed before the compensation takes place, as it is for example in the face recognition task. This way, the necessity of multiple compensations of one input image, each time for every individual possible face, would be eliminated. Moreover, there would be no need to have the 3D models of individual persons in the database. It would even allow the compensation of images of unknown, never-seen-before persons, so the compensation could be connected to another set of computer vision tasks, such as human gender or age detection.

The 3D model used as the generic one was the average head shape of US NAVY guy. As the texture for the face model, the average face of the face database from section 8 was used (see figure 24). The projection matrix used to render the 3D model was obtained as described in section 3, so the rendered image was allowed to scale and skew differently in each axis to match the acquired image. The generic 3D model is shown in figure 28.

The recognition rate of images compensated using this average face model was however bad. Only 47% (17 out of 36) of the compensated images were recognized correctly. For the not-recognized images, the illumination approximation was misjudged, ie. the optimization process converged to wrong spot. Yet, it is still an increase in the recognition rate, as the original, not compensated images, yielded recognition rate of 33%.

The recognition rate of images compensated using the generic model depends on the person in the image, probably in relation to the similarity between the person's face and the average face. For individual persons, the recognition rate ranged from 12.5% to 87.5%.

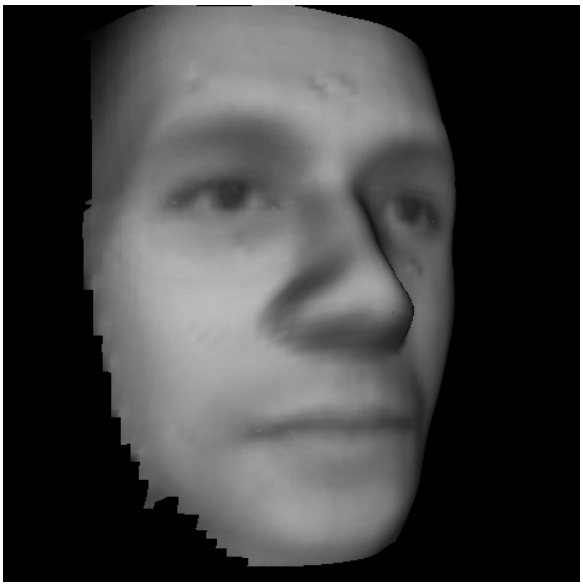

Figure 28: Rendered generic 3D model with the average face from the face database as the texture.

9.2 Dependency on the 3D Model Precision

The same set of 36 images was used to investigate dependency of the recognition rate on the 3D model precision. A white noise was added to the original facial models, ie. their vertices were randomly moved in each axis's direction. The noise deviation was gradually increased in units of percents of the face size. If the size of the modeled face was, for example, 15 cm, the deviation of 1% of the face size corresponded to 1.5 mm. Normal vectors of the face surface were then recalculated to reflect the changes in the shape, so the illumination of every surface triangle changed.

In figure 29 are shown changes in the recognition rate with increasing noise deviation. How the noise deforms the 3D face model is depicted in figure 33, where different rows represent increasing noise deviation. In the left column, deformed textured 3D models are shown, in the right column, images compensated using these models are depicted.

The result is, that the recognition rate remains good up to noise deviation about 1% if the face size. Over 2%, the compensation results are not satisfactory.

9.3 Measurements of the 3D Model Precision

To contemplate dependency on the 3D model quality, the precision of the model provided by the 3D reconstruction system should be first estimated. The estimation was done only for the ShapeSnatcher system. A rigid model of a head

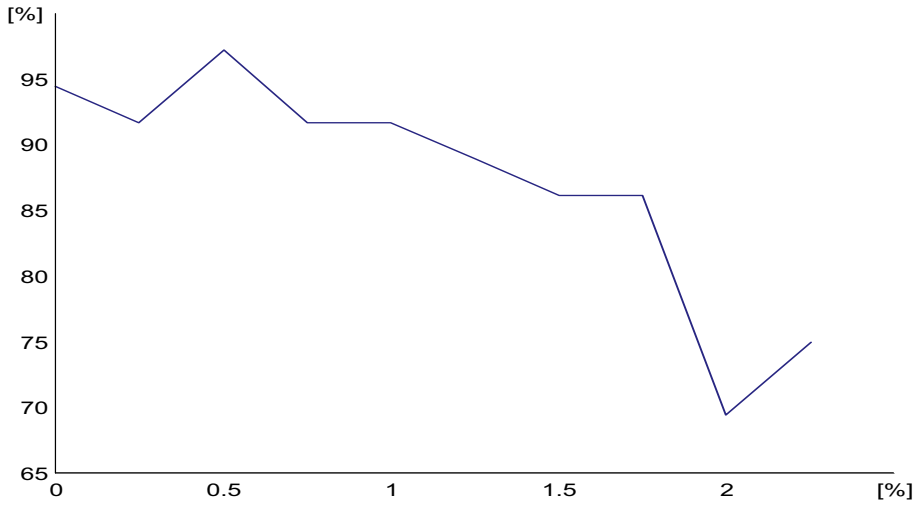

Figure 29: Recognition rate (in percents) in dependency on the 3D model noise deviation (in percents of the face size)

with 44 markers put on the surface was reconstructed for this purpose. The ShapeSnatcher was used to acquire 24 3D patches of the head model, each from a different angle. The all-around 3D model of the head was then generated by the ShapeMatcher tool, combining all the 24 patches into one large 3D model, consisting of more than 180,000 triangles. See figure 30, where the head model is shown.

The real 3D coordinates of the markers on the head's surface were obtained using a high precision 3D scanner.

To get the 3D coordinates of the markers on the reconstructed model's surface, their 2D coordinates were first identified in the texture image. The transformation from the texture coordinates to the 3D space is done in two steps:

1. The mesh triangle which contains the marker is found. Each vertex has associated its texture coordinates, ie. 2D coordinates in the texture image. This way every triangle of the mesh forms a triangle in the texture. All triangles are successively traversed to see if the marker's 2D position lies inside the triangle's projection to the texture. Only one such a triangle should be found, more only for markers laying on edges of triangles – it does not matter then, which one is used.

2. The exact 3D coordinates of the marker are calculated with a bilinear interpolation of the 3D coordinates of the three vertices of the triangle found in step 1.

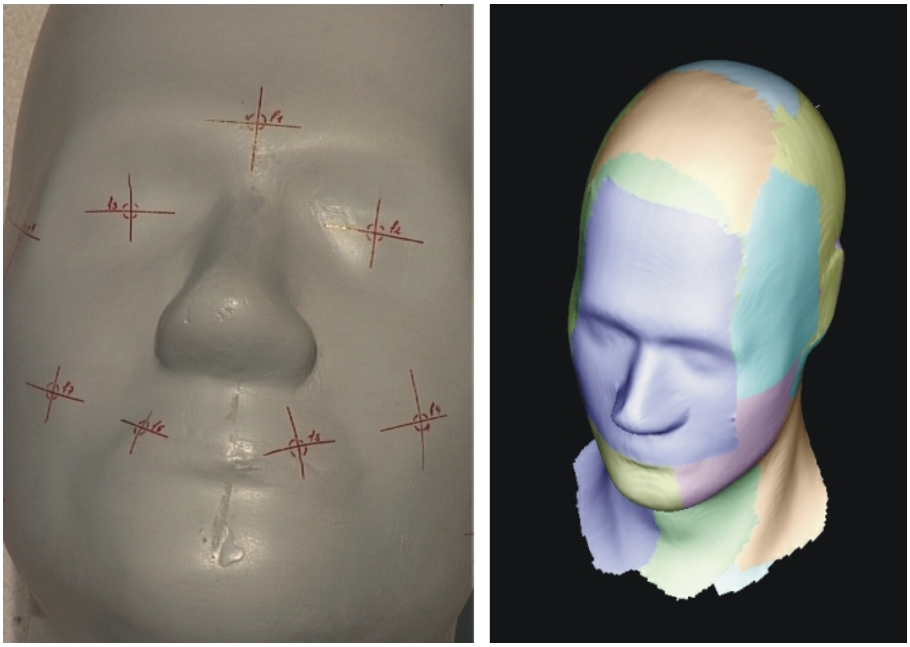

Figure 30: The head model. Left: a photo of the head with markers, right: the complete 3D model combined from 24 different patches

10 Conclusion

A method for recovery of a canonical illumination of a facial image, where the effects of the unknown illumination conditions are minimized, were developed and implemented, assuming that a 3D model of the face was available.

First, the global scene illumination characteristics were estimated so that the 3D model rendered to an image as similar to the to-be-compensated image as possible. Then, the image was compensated in conformance to the illumination estimation found. If the estimation was wrong, so was the compensation.

The quality of the compensation was evaluated using the eigen-face recognition system, where the compensation caused the recognition rate to increase from 33% to 94%. The evaluation was made on a set of 36 images of 5 different persons; the faces were sought in a face-database of 100 individual faces. Only 17 out of the 36 original facial images were recognized correctly, while 34 of them were recognized if the illumination compensation was applied beforehand.

The person's identity was to be known prior to the compensation, in order to use the right 3D model. Experiments with an universal 3D model shown that it is not a sufficient substitution for individual face models.

Images of four compensation processes are shown in figures 34 and 35. Individual columns stand for different acquired images. In the first row, the ac-

quired images I_a are shown. In the second row are images $\bar{I}_a$ rendered as the best approximations of I_as. Third row contains images $I_{detected}$, ie. images of intensities of the detected lights. These are obtained as the $\bar{I}_a$s rendered without the texture. Finally, the fourth row holds compensated images I_c.

The overbrightened areas on the images are where the direction of the incident light is nearly parallel to the surface. In contrary to the **assumption 2**, the face surface is not Lambertian. The intensity of the light reflected from the surface at grazing angles of the incident light direction is higher. See [Cornell] for description of the human skin reflectance properties. The model predicts lower light intensity in these areas. They are brightened accordingly to the model, thus too much for the real image.

The noise perceptible in brightened areas of the reconstructed faces was present in the original images too, but the image intensities were too low there for the noise to be noticeable. During the illumiantion compensation, the noise was magnified along with the image intensity increase.

11 Future Work

The problem of the illumination compensation is far from thoroughly examined. Here, a list of suggestions for further investigation is stated:

- To extend the method to work with colored models and with lights of arbitrary colors. The simplest solution is to split the task to three independent subtasks, each handling a separate RGB channel.
- To study the impact of multiple light sources on the compensation quality, for until now all the test images were taken under only one dominant directional light source.
- To examine the possibility of parametrization, of some kind, of the generic 3D model, so to increase the compensation quality for persons for which the compensation using the generic 3D model yields bad results. Significant dependency of the results on the person's identity suggests, that would the generic 3D model be adjusted to the person who is on the to-be-compensated image, likely according to some feature detection, the results may improve. Other possibility is to use more than one generic 3D model, and to choose one, hopefully the one most similar to the person's face shape, again according to some kind of feature detection in the image before compensation.
- To try out other algorithms for the cost function minimization. The advantages of the simplex downhill method, which is used now, is its robustness, and that it makes no special assumptions about the minimized function. The disadvantage, on the other hand, is the number of the function evaluations required to reach the minimum. Recalling that one function evaluation means complete rendering of the 3D model, followed

by pixel-by-pixel difference calculation between the rendered and the acquired image, decreasing the number of evaluations will cause significant gain in the overall speed of the compensation process.

- To try to use some more advanced rendering technique, such as raytracing or radiosity rendering, even exploiting information about the human skin BRDF (the bidirectional reflectance distribution function). See [Cornell] for description of human skin BRDF measurement. Such an advanced rendering method is expected to slow down the compensation process.
- To use hardware accelerated rendering, which is widely available now, to speed up the compensation process.
- To implement automation for tasks which are done manually now. These are the face location in the acquired image, and the facial feature point detection. Computer vision literature is full of methods addressing these problems.
- To do further testing and evaluation of the achievements of the compensation. More input data should be used, some reproducibility tests be done, and also achievements in cooperation with tasks other than the eigen-face recognition should be investigated.

References

[Br97] R. Brunelli: Estimation of pose and illuminant direction for face processing. *Image and Vision computing*, Vol. 15 (1997), 741-748.

[TP91] M. Turk and A. Pentland: Eigenfaces for recognition *Journal of Cognitive Neuroscience*, vol. 3, no. 1, 1991.

[NR92] Numerical Recipes In C: The Art Of Scientific Computing (ISBN 0-521-43108-5). *Cambridge University Press*, 1992. http://www.nr.com

[Matlab] Matlab *The Mathworks, Inc.* http://www.mathworks.com

[Cornell] Stephen R. Marschner, Stephen H. Westin, Eric P. F. Lafortune, Kenneth E. Torrance, Donald P. Greenberg: Reflectance Measurements of Human Skin *Technical report PCG-99-2, Program of Computer Graphics, Cornell University, Jan 99*
http://www.graphics.cornell.edu/pubs/1999/MWL+99a.html

[Simplex] J. A. Nelder and R. Mead: A Simplex Method for Function Minimization. *Computer Journal*, Vol. 7, 308-313

[CWS95] R. Chellappa, C. Wilson, and S. Sirohey: Human and machine recognition of faces: A survey. *Proceedings of the IEEE*, vol. 83, no. 5, pp. 705–740, 1995. http://www.cfar.umd.edu/ftp/TRs/CVL-Reports-1994/TR3339-Chellappa.ps.Z

[SI92] A. Samal and P. Iyengar: Automatic recognition and analysis of human faces and facial expressions: A survey. *Pattern Recognition*, vol. 25, pp. 65–77, 1992.

[MAU94] Y. Moses, Y. Adini, and S. Ullman: Face recognition: The problem of compensating for changes in illuminant direction. in *European Conf. on Computer Vision*, 1994, pp. 286-296. http://dlib.computer.org/tp/books/tp1997/pdf/i0721.pdf

[BHK97] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman: Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711-720. http://dlib.computer.org/tp/books/tp1997/pdf/i0711.pdf

[OF] Olivier Faugeras: Three-Dimensional Computer Vision, A Geometric Viewpoint. *The MIT Press, Massachusetts Institute of Technology, Cambridge, Massachusetts 02142.* FAUTH 0-262-06158-9

[BK92] Paul J. Besl and Neil D. McKay: A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, February 1992.

Appendices

A File Formats

In this appendix, description of file formats created for use within this work is given. They are all ASCII text files of identical structure, which is essentially a set of different sections. Section could be either a single-line, consisting of a keyword followed by a list of values, or a multi-line section, where a keyword is followed by an opening brace, then by a list of values, and finally by a closing brace.

A.1 The 3D Model File

The 3D model file contains all information related to a 3D model of the face. All sections are used only as an input, ie. they are not created or modified during calculations.

vertices Multi-line section, first value is the number of vertices, then for each vertex eight numbers follow: x, y, and z coordinate of the vertex, x, y, and z component of the normal vector of the surface at the vertex, and finally x and y texture coordinates. The texture coordinates are given in image space, ranging from zero to width resp. height of the texture image in pixels.

triangles Multi-line section, first value is the number of triangles, then, each triangle is described with three zero-based indices of the vertices forming the triangle.

texturesize Single-line section with two values – width and height of the texture in pixels.

texture Single-line section with a single value – the file name of the texture image in JPEG format. No spaces are allowed in the file name.

projection Multi-line section, three times four values forming the projection matrix projecting the 3D coordinates of the vertices onto their texture coordinates. Used together with feature points to compute a projection from the 3D model to an acquired image (see section 3).

features Multi-line section, seven times two values representing x and y texture coordinates of seven feature points identified on the face. Currently, these feature points are: left corner of the left eye, right corner of the left eye, left corner of the right eye, right corner of the right eye, left corner of the mouth, right corner of the mouth, and the tip of the nose.

An example of the 3D model file:

```
vertices
{
2706
109.82  -13.3613 -104.103 0.5782 -0.05584 0.8139 586.008 547.074
   .       .         .     .       .       .        .       .
   .       .         .     .       .       .        .       .
   .       .         .     .       .       .        .       .
 63.801 -41.0729  211.935 0.655   0.01442 0.7554 530.641 441.93
}
triangles
{
4267
1136  1137  1149
 .     .     .
 .     .     .
 .     .     .
2631  2632  2649
}
texture texture.jpg
texturesize 760 484
projection
{
 2.003829   -0.08618332  0.6314119  367.1147
-0.02423601 -1.781183   -0.03714525 462.8707
-0.00013448 -0.00014599  0.00091631   1
}
features
{
 307 284
 365 278
 444 282
 502 291
 333 379
 446 388
 391 318
}
```

A.2 The Compensation Infomartion File

This file contains all information related to the illumination approximation and compensation. The first three sections are inputs only, the last two are both results of computations and inputs for other tasks.

model Single-line section, single value: a name of the 3D model file, as described in section A.1.

image Single-line section, single value. File name of the acquired image, ie. the image which is to be compensated for the illumination effect, in JPEG format. No spaces are allowed within the file name.

features Multi-line section, seven times two values, x and y image coordinates of seven feature points identified on the face. The order of feature points must match feature points within the 3D model, ie. currently these feature points are: left corner of the left eye, right corner of the left eye, left corner of the right eye, right corner of the right eye, left corner of the mouth, right corner of the mouth, and the tip of the nose.

projection Multi-line section, three times four values forming the projection matrix projecting the 3D model onto the **image** image. Calculated from the projection matrix of the 3D model and from feature point locations on both the 3D model's texture and the acquired image (see section 3).

light Single-line section, four values of the illumination approximation. Computed during iterative search for the best illumination approximation (see section 4.3), used during illumination compensation (see section 5). The values stand for the ambient light intensity, the directional light intensity, and φ and ψ angles defining the directional light direction (see section 4.1).

An example of the file:

```
model face2
image face2im4.jpg
light 0.0536281 1.194 -0.917901 -0.273749
projection
{
 1.97515    -0.00641959 0.641879  359.921
-0.0314425  -1.83432   -0.0466655 469.044
-0.00013448 -0.00014599 0.00091631  1
}
features
{
282 263
340 258
398 260
460 261
309 380
423 376
348 330
}
```

A.3 The Face Image Information File

This file format describes an image of a face which is to be used within the eigen-faces recognition algorithm. So this file is created for every image from the learning set, and also for every image which is to be recognized. The first two sections are inputs only, the other two are both results of computations and inputs for other tasks.

image Single-line section, single value – a file name of the face image in JPEG format. No spaces are allowed in the file name.

features Multi-line section, seven times two values, x and y image coordinates of seven feature points identified in the face image. These are used to calculate an affine transformation of the image to a common eigen-image space, where all faces are aligned in a fixed image position (see section 8).

transformation Multi-line section, three times three values forming a 2D transformation matrix, which, computed from feature point locations, transforms the image into a common eigen-image space (see section 8).

coefficients Multi-line section, represents the weight vector Ω of the image's projection into the eigen-space (see section 7.1). The first value is the length of the vector (ie. M'), then, $\omega_1, \ldots, \omega_{M'}$ follows.

An example of the file:

```
image face2im4.jpg
transformation
{
  354.468 -5.12437 0
  -17.0473 322.624 0
  206.839 142.949 1
}
features
{
  282 263
  340 258
  398 260
  460 261
  309 380
  423 376
  348 330
}
coeficients
{
39
  -0.0919629
    .
```

```
    .
    .
  -0.0854766
}
```

A.4 The Eigen-face File

Eigen-faces are vectors of floating point values, with size equal to number of pixels in face images, which is currently set to $256 \times 256 = 65536$ pixels. These vectors are stored in raw binary format, every element in IEEE standard 8-byte floating point format. So the length of the eigen-face file is $256 \times 256 \times 8 = 524288$ bytes. These files are generated during initialization of the eigen-face recognition system (see section 7.1), and then used during recognition, when an unknown image is projected onto the eigen-space (see section 7.2).

A.5 The Eigen-face Set File

This file format describes both the learning set of the eigen-face recognition system and the computed eigen-space.

imageset Multi-line section, represents a list of images of the learning set used to build the eigen-space. First value is the number of face images (M in section 7). Then follows a list of file names of the Face Image Information files, which are in format described in section A.3.

averageimage Single-line section, value is the file name of the average image of the learning set. The format of the file is described in section A.4.

eigenimages Multi-line section, represents a list of the base vectors of the eigen-space, ie. a list of individual eigen-faces. First value is the number of eigen-faces (M' in section 7). Then follows a list of M' pairs, where the first value is the file name of the eigen-face file (described in section A.4), and the other value is the eigenvalue corresponding to the eigenvector represented by the eigen-face.

directory Optional single-line section, the only value identifies the directory where the files from sections **imageset**, **averageimage**, and **eigenimages** are located.

An example of the file:

```
directory s:\eigenfaces\
imageSet
{
100
  learningset1.FI
  .
  .
```

```
  .
  learningset100.FI
}
eigenImages
{
39
  Set1_00.dbl 85704.4
  .               .
  .               .
  .               .
  Set1_38.dbl 773.061
}
averageImage set1_avg.dbl
```

A.6 The Measurement Info File

This file format is used for conversion from texture coordinates to 3D coordinates of corresponding points on the 3D shape surface (see section 9.3).

meshtype Single-line section, value specifies the 3D mesh type. Allowed values are "SS3D" for decompressed meshes generated by the ShapeSnatcher tool, and "TOP" for decompressed topology files generated by the ShapeMatcher tool.

mesh Single-line section, value is the file name of the 3D mesh file in format specified by the **meshtype** section.

texsize Single-line section, two values stand for the texture width and height.

2D Multi-line section, represents a list of 2D points in texture image space. First value is the number of points, then list of coordinate pairs follows. Coordinates are in range $[0 \ldots \textit{texture width}] \times [0 \ldots \textit{texture height}]$, where *texture width* and *texture height* are values of the **texsize** section.

3D Multi-line section, represents a list of points in the 3D model space. First value is the number of points, then list of coordinate triplets follows. This section is an output of the texture to 3D mapping process, where points from the **2D** sections are the input. So the amount and ordering of the 3D points correspond to the amount and ordering of the points in the **2D** section.

An example of the file:

```
meshtype SS3D
mesh ph702.ss3d
texsize 640 480
2D
{
```

```
7
  151 165
  232 169
  300 171
  382 172
  198 371
  350 370
  257 277
}
3D
{
7
  12.263 25.3016 271.646
  15.4597 25.3289 270.336
  18.1515 25.3434 269.724
  21.4577 25.4063 269.948
  14.1481 33.1198 270.347
  20.1946 32.9795 269.531
  16.2736 29.1068 266.696
}
```

B Used Symbols

- I_a - an image acquired for compensation
- I_c - a compensated image
- $\bar{I}_a$ - an approximation of I_a rendered from a textured 3D model
- $I_{detected}$ - an approximation of I_a rendered without texture, showing only the incident light intensity
- $I_{desired}$ - a rendered 3D model without texture, showing only the intensity of the desired illumination
- A - an intensity of the ambient light
- L - an intensity of the directional light
- ω - an angle between the light direction and a surface normal vector
- Λ - the surface albedo (texture)
- φ, ψ - the light direction parametrization
- $\mathbf{P}$ - a matrix of a projection mapping a 3D model to a 2D acquired image I_a
- $\mathbf{P}_t$ - a matrix of a projection from a 3D model to its texture

- $\mathbf{T}$ - a matrix of a 2D affine transformation
- $\mathbf{T}_L$ - a matrix of a 3D transformation rotating the 3D model into the light direction, used during rendering of shadows
- $\mathbf{P}_L$ - a projection matrix of the 3D model rotated to the light direction, used during rendering of shadows
- M - number of images in the eigen-face recognition system's training set
- M' - dimensionality of the eigen-space, ie. number of eigenvectors used for the face recognition
- $\mathbf{\Gamma}_n$ - an image from the eigen-face recognition system's training set
- $\mathbf{\Psi}$ - the average face image
- $\mathbf{\Phi}_n$ - the difference between a face image $\mathbf{\Gamma}_n$ and the average face image $\mathbf{\Psi}$.
- $\mathbf{C}$ - the covariance matrix of the face images from the training set
- $\lambda_1 \ldots \lambda_{M-1}$ - nonzero eigenvalues of $\mathbf{C}$
- $\mathbf{u}_1 \ldots \mathbf{u}_{M-1}$ - eigenvectors of $\mathbf{C}$ corresponding to nonzero eigenvalues
- $\mathbf{\Omega} = [\omega_1 \ldots \omega_{M'}]$ - the weight vector of a face image, ie. a projection of the image onto the eigen-face space
- $\bar{\mathbf{\Gamma}}$ - an approximation of a face image reconstructed from the eigen-faces

Figure 31: Sixteen eigen-faces with the highest eigenvalues (upper four rows), and four eigen-faces with the smallest eigenvalues (bottom row).

Figure 32: The face database used to compute eigen-faces.

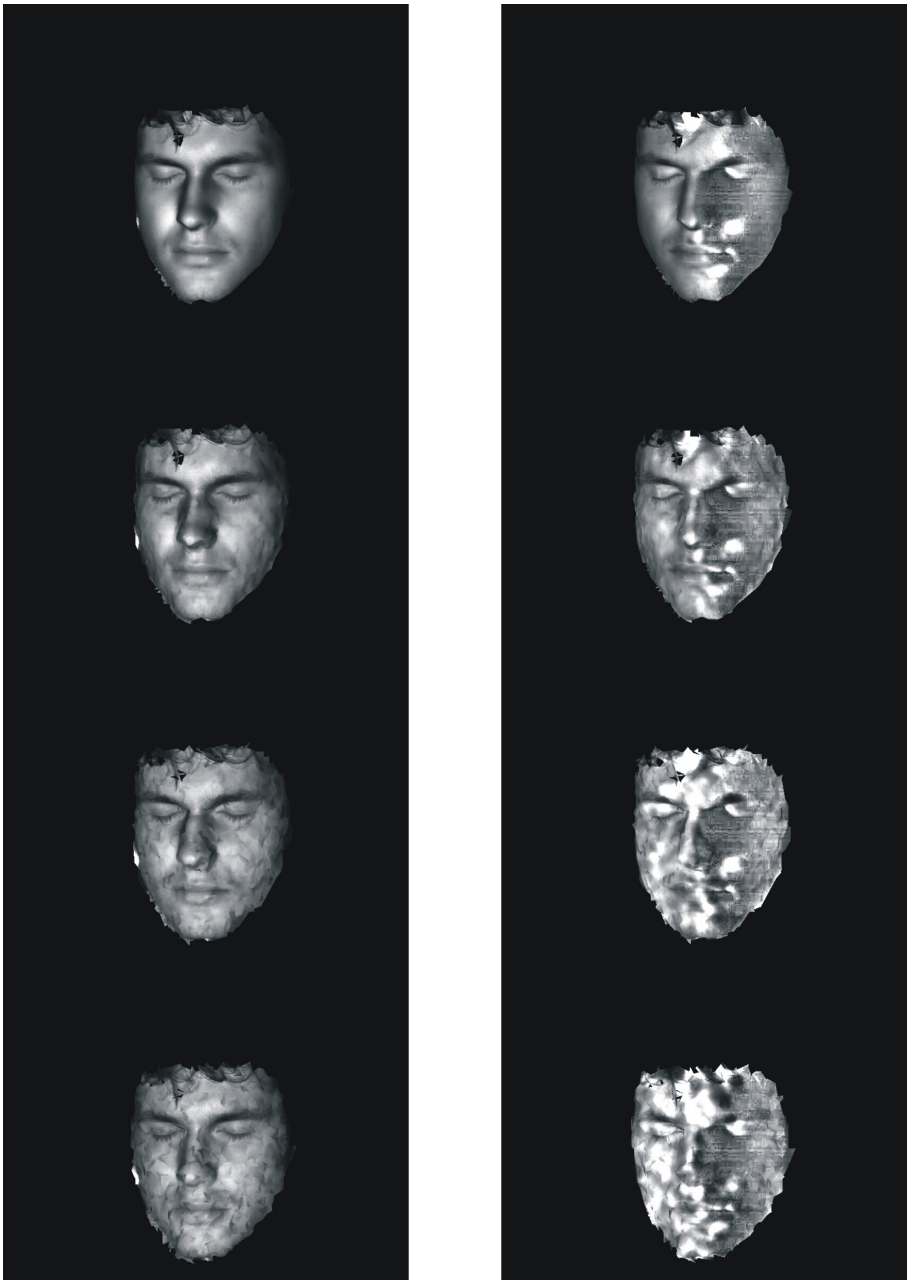

Figure 33: A 3D model with an additional noise. Left column – rendered 3D model, right column – a compensated image based on the model. Upper row – no noise, lower row – a white noise with the deviation of 2% of the face size.

Figure 34: Examples of compensation

Figure 35: Another examples of compensation