Giovanni Dainese, Marco Marcon, Augusto Sarti,
Stefano Tubaro
Dipartimento di Elettronica e Informazione,
Politecnico di Milano

Krzysztof Kucharski, Władysław Skarbek
Faculty of Electronics and Information Technology,
Warsaw University of Technology
K.Z.Kucharski@elka.pw.edu.pl

Abdul H. Sadka, Yun Sheng
Centre for Communication System Research,
University of Surrey

Kraków, June 17, 2005

TOWARDS 3D FACE RECOGNITION

Abstract: **3D face recognition framework is analyzed with its basic counterparts presented i.e. face localization, facial feature extraction, 3D face model reconstruction from a few 2D images and 3D eigenface. The main advantage of the proposed framework is its fast processing time. The most time consuming step that is the 3D face model building takes less than 3 seconds at Pentium 4 3GHz 1GB RAM for image size about one milion pixels.**

1 INTRODUCTION

Scientists and researchers all over the world have been pursuing an automatic face recognition (FR) for about two decades. The potential applications include surveillance of the airports, railway stations, stadiums, etc. what becomes increasingly important these days but also browsing of the multimedia content such as films, video shots or streaming video. Face recognition involves face localization process to state if there is a face in the analyzed image and to precisely distinguish it from the background. The recognition task itself can be verification of a human identity given the database of various persons photos. Though, more often some number of the most similar facial images is needed what is always the case when the database is huge and thus impossible to be browsed manually.

A great number of different approaches to face recognition have been proposed in the last twenty years – a good overview of the first decade can be found in [1]. The common approach is to learn a feature set in course of processing two dimensional face images as holistic patterns given that the training set covers sufficiently broad range of face appearance variability. Unfortunately, these classic approaches suffer from major drawbacks like illumination dependance of the recognition, lack of some face information due to a non-frontal pose acquisition of the camera, confusing the recognition engine because of a planar image of the person, etc.

Many of these problems can be either reduced or avoided using a 3D approach where different spatial models of the acquired faces are directly compared. For this reason a classic approach for depth-maps reconstruction that produces a detailed 3D texture-mapped mesh of the acquired face from a few of its 2D images is adapted. A face has to be first localized within the 2D image and crucial facial features describing eyes, mouth, chins etc. extracted. These points are then used during depth map reconstruction to apply normalization in such a way that a 3D model is rotated and scaled to obtain a frontal view with a constant distance between the two eyes. The transformation also normalizes all the facial feature points to provide them for a recognition module which is based on 3D eigenfaces method.

We report in this paper the advanced state of research on the 3D face recognition framework. In section 2 the face localization by the AdaBoost cascade is described, in section 3 the most important facial feature extraction methods are presented; section 4 contains the main part of the paper that is a 3D face model building algorithm based on a global energy minimizer derived from graph-cuts techniques. In section 5 the 3D eigenfaces method is sketched and section 6 concludes the paper pointing out the future research directions.

2 FACE LOCALIZATION

Among many of face detection and localization methods [2] the AdaBoost cascade classifier introduced by Viola and Jones [3] is proven to be one of the most successful both in terms of accuracy and speed. Their really novel approach has shown how local contrast features found in specific positions of the object can be combined to create a strong face detector. AdaBoost

is known from the late 1980s as a multi-classifier and a training procedure for a collection of weak classifiers, e.g. having the success rate about 0.5, to boost them by suitable voting process to very high level of performance. Viola and Jones applied an early and suboptimal heuristics of Shapire and Freund for AdaBoost training algorithm [4].

The Adaboost cascade algorithm for object detection may be described as nested three-level process [5]. On the lowest level parameters of the best single weak classifier are found, that on the higher level is incorporated into the set of such weak classifier forming the strong classifier. The third and the highest level creates serial connections between strong classifiers to maintain the trade-off between crucial detection performance measures, false acceptance rate and false rejection rate. The resulting final structure is called the AdaBoost cascade.

2.1 The Weak Classifier

For each object window o, the weak classifier elaborates a decision $\delta_\omega(o) \in \{-1, +1\}$ on the basis of membership of the object o to one of two classes labeled by -1 (a negative) and +1 (a positive) (Figure 1).

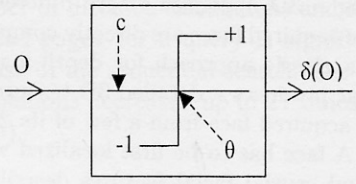

Figure 1: *Diagram of the weak classifier*

In face and eyes detection region contrasts $c(R)$ are used as weak classifiers, where R is the sub-window of the image window o:

$$c(R) = \sum_{(x,y)\in R^+} o(x,y) - \sum_{(x,y)\in R^-} o(x,y)$$

The regions used to evaluate contrast in our implementation of AdaBoost cascade, which were taken directly from [3], are presented in Figure 2. The positive subregion R^+ is drawn in white whereas the negative subregion R^- is drawn in black.

region	type A	type B	type C	type D
width	2a	a	3a	2a
height	b	2b	b	2b

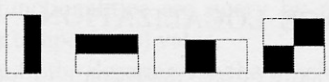

Figure 2: *Types of regions used to determine the weak classifier*

During the training process the optimal choice of threshold θ and parameter sequence (x, y, a, b, t) is made, where a pair (x, y) is the anchor point of the region defined by the type $t \in \{A, B, C, D\}$ and size a and b in the object window o. The training is performed with respect to two labelled datasets consisting of P positives and N negatives respectively. The examples referring to the case of face detection are illustrated in Figure 3.

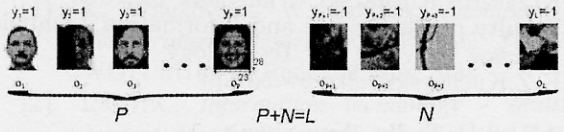

Figure 3: *Positive examples (faces) on the left and negative examples (non-faces) on the right used in the process of the weak classifier training*

2.2 The Strong Classifier

For each object window o, the strong classifier conducts a procedure of weak classifier weighted voting where each weight is called the cost of a decision $\gamma_t(o)$ of the corresponding weak classifier i.e. the error it causes when applied to the training set (Figure 4).

In our version of the AdaBoost algorithm [6], [7] it is assumed that the cost $\gamma_t = \beta_t$ of the negative decision $\delta(o) = -1$ is k_t times greater than the cost $\gamma_t = \alpha_t$ of the positive decision $\delta(o) = 1$.

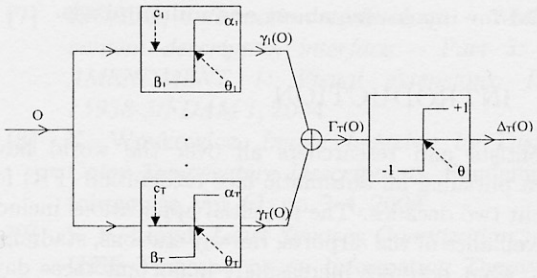

Figure 4: *Diagram of the strong classifier*

The fact that the weak classifiers within the strong classifier are different follows from the main property of the AdaBoost theory, namely the step of changing the training examples weights after every single weak classifier was found. The weights are uniformly initialized i.e. $w_{i,1} = \frac{1}{L}, i = 1, \ldots, L$ and are modified due to the formula:

$$w_{i,t+1} = \frac{w_{i,t} e^{-\gamma_i(o_i) y_i}}{\sum_{i=1}^{L} w_{i,t} e^{-\gamma_i(o_i) y_i}}$$

The assumption about the relation $\gamma_t = k_t \alpha_t$ with $k_t = 3$ leads to the strong classifier that consists of eight weak classifiers and has a false rejection rate $fr = 0.0005$, false acceptance rate $fa = 0.48$ on the training set composed of $P = 4000$ faces and $N = 8000$ non-faces. The respective contract regions overlayed on a positive example are shown in Figure 5.

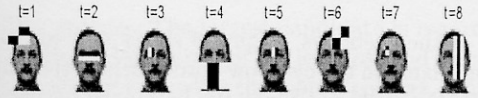

Figure 5: *Components of the strong classifier obtained in course of AdaBoost training process applied to the faces.*

2.3 The Cascade

A stage of the cascade is built by the AdaBoost algorithm i.e. the process of finding the strong classifier, in which the termination condition is satisfied when the false acceptance rate fa is below a threshold, while the false rejection rate fr does not exceed a given level. The specific values of these quantities have to be chosen depending on the properties of the object that is to be detected. For instance, the false acceptance threshold for both face and eyes detection was set to 0.5 whereas the false rejection level was set to 0.005 in case of face detection and 0.01 for eyes detection.

The termination condition for the cascade consisting of K stages is fulfilled if the total false acceptance rate $FA = fr_1 \times fr_2 \times \cdots \times fr_K$ drops below a threshold which we assumed to be 0.000005 for face detection and 0.001 for eyes detection. Value between 10^{-6} and 10^{-5} in face detection case is widely reported in literature [3], but in case of the eyes it turned out that these data differ too little between various examples from the training set to reach such an extreme value of total false acceptance. Hence the conclusion is that eyes detector cannot be used to detect eyes in the whole input image but only within the image of face found earlier with the face detector.

In face localization case, the output AdaBoost cascade structure we trained consists of 17 stages and 972 weak classifiers, subsequent stages having respectively 8, 11, 18, 19, 26, 38, 48, 48, 62, 68, 65, 67, 91, 84, 100, 112, 107 weak classifiers. In eyes detection case output structure consists of 8 stages and 596 weak classifiers, subsequent stages having respectively 8, 16, 26, 45, 87, 99, 142, 173 weak classifiers. The much more rapid growth of cascade stage size in eyes detector may be noticed, it confirms that there is less discriminatory information contained in eyes images. Also, about one of every 100 object windows can pass through first 6 stages of the cascade, therefore high numbers of weak classifiers in the further stages basically do not affect speed of the detection process.

2.4 Face Localization Results

The localization process by the AdaBoost cascade algorithm is simply a scanning the input high-resolution image, anchoring the object window in every single pixel of the image and classifying it to one of two classes i.e. object or non-object, using the AdaBoost cascade structure. The cascade is completely described with the parameter values of all its strong and weak components thus forming the object, e.g. face, eyes detector.

In spite of a very low level of false acceptance rate, the detector has to be prepared for the situations where false alarms occur. Moreover, in the case of correct detection we can expect for sure the more that one alarm because more than one bounding box surrounds the object, e.g. a face or an eye pair. The way of tackling such a problem is based on assigning weights being level of alarms to the objects that pass sufficient number of the cascade stages. The higher stage is achieved the greater weights is assigned. An analysis of the $3x3$ neighbourhood of every pixel in so-formed alarm image reveals if the given pixel is an anchor point of an actual face area. It is always the case when the average alarm level in the neighbourhood exceeds a specific threshold.

The threshold used to exclude some alarm points during the face detection is selected heuristically, basing on the fact how small faces the designed detector is to deal with. Actually, the faces in smaller scales have smaller number of alarms than larger faces. Therefore, lowering the threshold for small scale can help to detect small faces but, on the other hand can include also the false rectangles into the set of results. Example of such a situation is presented in Figure 6.

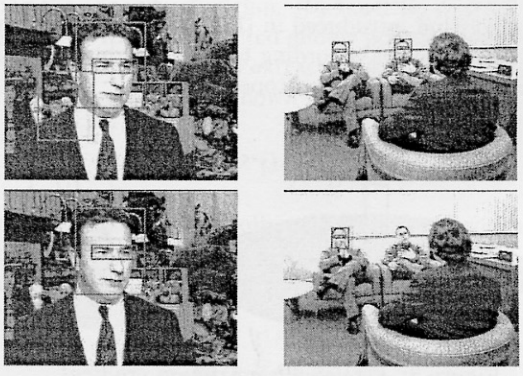

Figure 6: *Detection results in presence of smaller (upper row) and greater (bottom row) thresholds.*

3 FACIAL FEATURE EXTRACTION

The extracted facial features serve as landmarks, so as to obtain the specific facial information. Too many features would increase computational complexity, whereas too few would produce incomplete facial information. Targeting variant facial features, the existing facial feature extraction approaches can be categorized into brightness-based and edge-based algorithms. Bright-ness-based algorithms, typically template matching [8] and region-growing search [9] etc, employ the knowledge of the geometrical topology and the brightness characteristics of facial features, such as the eyebrows, eyes and mouth, to find the approximate position of facial features. Edge-based algorithms, typically Hough transform [10], active contour model or Snakes [11], and deformable template matching [12], target contours of the mouth, eyes and chin, usually

based on gradient images. Before these edge-based algorithms are applied, the position of facial features must be found first. Hence, brightness-based and edge-based algorithms are viewed as mutually complementary in facial feature extraction. This section elaborates an automatic algorithm to extract eight points of the eyebrows and contours of the eyes, irises, mouth lips and chin. These features are high-lighted in Figure 7.

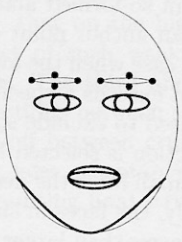

Figure 7: *Illustration of extracted facial features.*

3.1 Extraction of Eyes and Eyebrows

The algorithm introduced in [13] is used to locate the eyes and eyebrows according to their brightness characteristic and geometrical topology.

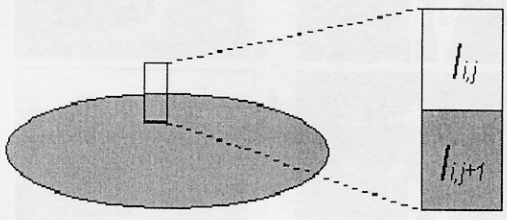

Figure 8: *Pixels with maximum gradient.*

It is found that the conventional deformable template matching algorithms used in eye contour extraction are limited to those images with low resolution. Thus, instead of using deformable template matching algorithm, we proposed a solution that only employs the idea that the eye contours can be described as a combination of several parabolas. After four feature points, i.e. the left and right corners, and the upper and lower middle points of the eyelids, are in turn extracted from the located eye using the method published in [13], the eye contours can be extracted with four fixed parabola templates. In addition, the irises can be extracted using the Hough trans-form when the eyes are open [10].

Due to relatively low intensity of the eyebrows, an automatic eyebrow segmentation approach is presented as follows. Let $I_{i,j}$ denote the intensity of a certain pixel within the narrowed search area, where the subscripts i and j indicate the horizontal and vertical coordinates of the image plane, respectively. The downward intensity gradient at $I_{i,j}$ can be formulised as $g_{i,j} = I_{i,j} - I_{i,j+1}$. Then we seek the maximum gradient value G_i on the current column because the place where the maximum gradient arises must be on the border of the eyebrows. Figure 8 shows two vertically adjacent pixels on the eyebrow border, where the ellipse indicates the eyebrow. Once the boundary position of the current column is discovered, we can determine the threshold value of the column by averaging the intensities of the two adjacent pixels as:

$$T_i = \frac{1}{2}(I_{i,j} + I_{i,j+1})$$

To normalize the threshold, the mean of the determined thresholds on all the columns of the eyebrows is finally calculated:

$$T = \frac{1}{n}\sum_{i=i_0}^{i_n-1} T_i$$

where n denotes the number of calculated columns, normally set to the width of the corresponding eyebrow, and T is the finalized threshold value used to segment the eyebrows.

3.2 Mouth Extraction

Compared with other methods, the deformable template matching algorithm is one of the most effective ways to locate mouth lip contours due to its the continuity and flexibility, nevertheless it still has some challenges, such as template positioning, criterion of mouth being open or closed, and selection of strength constraints. To solve these problems, we address an automatic mouth extraction algorithm with Staged Deformable Templates (SDTs).

In this algorithm, mouth corners are used as two reference points for placing the mouth templates. This raises a problem of how to segment the mouth, and locate the corners. For mouth segmentation, the search area can be narrowed into the lower half of the face. Besides, the mouth lip region is rather sensitive to the chrominance component, Cr, and always has local peak value in Cr because mouth lips appear red (top left image of Figure 9). Thus, based on the above knowledge, a heuristic thresholding scheme [14] is adopted:

1. Select an initial estimate for T
2. Segment the image using T. This will produce two groups of pixels: G_1 consisting of all pixels with gray level values $> T$ and G_2 consisting of pixels with values $\leq T$
3. Compute the average Cr values μ_1 and μ_2 for the pixels in regions G_1 and G_2
4. Compute a new threshold value: $T = (\mu_1 + \mu_2)/2$
5. Repeat steps 2 through 4 until the difference in T in successive iterations is smaller than a predefined parameter T_0.

Initially T_0 is set to zero, and the estimate for T is chosen as a median of the search area, since the desired mouth is small compared to the background of the search area. To remove noise, a morphological opening process is applied to the binarized image, followed by a

labelling process. The largest candidate is regarded as the mouth, middle points of the leftmost and rightmost of which are considered as the mouth corners. The bottom left image of Figure 9 shows the extracted mouth corners with a carphone video frame.

Figure 9: *Mouth extraction.*

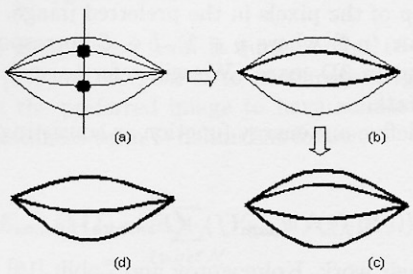

Figure 10: *Staged mouth extraction.*

Discrimination of whether the mouth is open or closed generally affects the selection of the mouth templates. The possible appearance of teeth usually interferes with discrimination, so our proposed criterion is based on the chrominance component, Cr, because of its sensitivity to the red-appearing mouth lips, and its insensitivity to the teeth. On the Cr image, if the mouth is open, there is a valley inside (top left image of Figure 9). But when the mouth is closed, the valley disappears. Therefore, according to this characteristic on the Cr image, a pixel $P(x, y)$ with minimum intensity value on the perpendicular bisector of the mouth corners is explored with $x_m < x < x_M$, where x_m and x_M are extrema of vertical coordinates of the segmented mouth. Our criterion is to compare $P(x, y)$ with a predefined threshold T. If $P(x, y)$ is less than T, the mouth is considered open; otherwise, the mouth is closed.

To suppress the occurrence of mismatch especially if one of the mouth contours fails to be extracted, the procedure of mouth contour extraction is broken down to three stages, as shown in Figure 10 (a), (b) and (c) in a mouth-open state: (a) Middle point extraction of inner lip contours: To avoid the effect of teeth when the mouth is open, a method of calculating only gradients of the pixels on the perpendicular bisector of mouth corners is developed, which is unlike the existing methods of taking edge in-formation along the inner lip

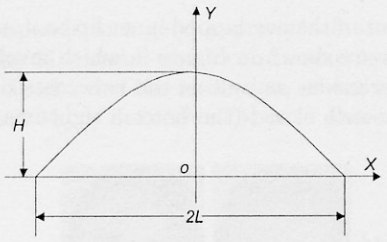

Figure 11: *Parabola template.*

contours into account. As stated before for Cr, since a valley emerges if the mouth is open, the calculation will be based on Cr. However, as the mouth is closed, the valley disappears in Cr, whereas another valley rises between the lips in luminance. So the calculation will be based on the luminance. From the top of the mouth to the pixel before $P(x, y)$, the gradients between the current and the next pixels along the perpendicular bisector of mouth corners are evaluated. The pixel with a maximum gradient value is treated as the middle point of the inner upper lip contour. Similarly, the middle point of the inner lower lip contour can be extracted along the opposite direction. (b) Inner lip contour extraction: With the extracted middle points, each of the inner lip contours can be extracted, and formulated in terms of a parabola, as shown in Figure 11:

$$Y = H \times (1 - (\frac{X}{L})^2)$$

where H and $2L$ symbolize the high and width of the parabola, respectively. (c) Outer lip contour extraction: Only two elaborately selected strength constraints are required:

$$E_t = E_e + E_r$$

where E_e and E_r are the edge strength and the region strength, respectively, and given as:

$$E_e = -\frac{1}{2L} \int_{2L} e(x, y) ds \quad (1)$$

$$E_r = -\frac{1}{A} \int_{A} C(x, y) dr \quad (2)$$

In equation (1), $2L$ is the length of arc; $e(x, y)$ denotes the edge intensity; s indicates the trajectory of facial feature contours. In equation (2), A stands for the area of the regions, $C(x, y)$ indicates chrominance intensity; r represents the scanned area.

To eliminate the impact of teeth, the edge image is derived from the chrominance Cr by using gradient-based edge detection with Sobel gradient operators. For the same purpose, calculation of the region strength is also based on Cr. The outer lip contour template deforms outwards so that the best-fit outer lip contours are found by minimizing the strength equation (3.2).

For the closed mouth, a mouth-closed template is similarly defined in Figure 10 (d), with the fact that the point $P(x, y)$ is directly regarded as the unique, shared

middle point of the overlapped inner lip contours. Several results are shown in Figure 9, which involve some typical expressions as well as the two cases of mouth open and mouth closed (the bottom right image).

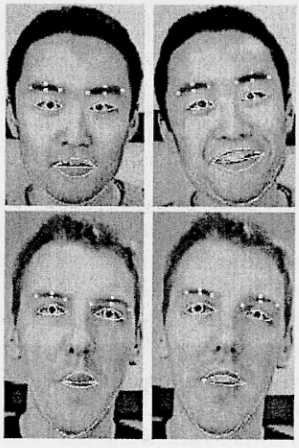

Figure 12: *Facial feature extraction results.*

3.3 Chin Extraction

Since there is a narrow shadow appearing between the face and the neck, the chin contour can be extracted using deformable template matching, as shown in Figure 7. The technical detail of the deformable template match-ing for chin extraction can be found in [15].

Figure 12 shows some facial feature extraction results.

4 3D face model reconstruction

In this section we show how to accurately reconstruct the depth-map of a face from a set of images. In order to do so, we started from the well-known *graph cuts* approach [16, 17, 18, 19], and we adapted it and optimized it to the problem of depth-map reconstruction.

In what follows we provide a brief description of the energy minimization approach that the graph cuts method is based on. After then we will show how to formulate the problem of depth map reconstruction in term of energy minimization.

4.1 Energy Minimization Approach

It is well known that the problems of depth map reconstruction and image restoration can be elegantly approached in terms of energy minimization [17, 18], with extremely appealing results. In the past few years powerful energy minimization algorithms have been developed based on graph cuts [17, 20, 21]. These methods are fast enough to be of practical interest, but unlike other methods such as simulated annealing, the solutions based on graph cuts cannot be applied to arbitrary functions. In this paper we will use some recent results [18] on graph construction, in order to extend the method to quite a general class of energy functions.

The energy minimization formalism exhibits several advantages. It allows a detailed description of the problem to be solved. Moreover, energy minimization naturally enables the use of soft constraints, such as spatial coherence and a global smoothness term. This allows us to avoid ambiguities with spatially smooth answers that preserves discontinuities.

4.2 Problem Formulation

Let us assume that n calibrated images of the same scene are taken from different viewpoints (or at different times). Let us choose a reference camera and let $\mathcal{P}$ be the set of pixels of the corresponding image. A pixel $p \in \mathcal{P}$ corresponds to a ray in 3D-space. Consider the first intersection of this ray with an object in the scene. Our goal is to find the depth of this point for all the pixel of the preferred image. We thus want to find a labelling $f : \mathcal{P} \rightarrow \mathcal{L}$ where $\mathcal{L}$ is a discrete set of labels corresponding to increasing depths from the preferred camera. Equivalently, we want to obtain the *depth map* of the pixels in the preferred image.

A pair $\langle p, l \rangle$ where $p \in \mathcal{P}$, $l \in \mathcal{L}$ corresponds to some point in 3D-space. We will refer to such points as *3D-points*.

We define our energy function as consisting of two terms:

$$E(f) = E_{data}(f) + E_{smooth}(f)$$

In their work, Kolmogorov and Zabih [16] formulate the problem of scene reconstruction in a slightly different fashion, which allowed them to obtain a depth map for every image in the input set by an energy minimization approach. This leads to a computational expensive algorithm whose result is a unorganized clouds of point representing the surface of the visible part of the scene to reconstruct. Moveover, in order to achieve an effective reconstruction from the input set, a further energy term (called *visibility term*) must be accounted for, in order to avoid mutual intersections of re-projected rays coming from different cameras (see [16] for more details). Whereas with our definition we can treat a very large number of camera configurations without these further limitations.

Notice also that in our approach it is no long necessary to define the visibility term like in [16]. In fact, assuming that the set of label corresponds to the increasing depths from the preferred camera, there cannot exists occluding pixels in the same image. As a consequence, a visibility term is no longer necessary. The other terms are also quite different. Our data term, for example, is defined as follow:

$$E_{data}(f) = \sum_{p \in \mathcal{P}} D(p)$$

where $D(p)$ is a non-positive value which results from the differences in intensity between corresponding pixels. $D(p)$ is computed for every pixel of the preferred image (we indicate this image with the index j) by this steps:

1. from p, we get the corresponding 3D-point by back-projecting it from the reference camera center of projection with the selected depth and then we project this 3D-point on each other calibrated image obtaining a set of $n-1$ corresponding pixels $\{q_1, q_2, \ldots, q_i, \ldots, q_n | i \neq j\}$;

2. on every non-reference image we compute the SSD (Sum of Square Difference) using a square window centered on q_i and the one centered on p, obtaining the set of values $\{d_1, d_2, \ldots, d_i, \ldots, d_n | i \neq j\}$;

3. finally, we have

$$D(p) = min(0, \sum_{\substack{i=1 \\ i \neq j}}^{n} d_i - K) \tag{3}$$

where K is a positive constant that is large enough to capture significant variations of the SSD function (a typical value is $K = 30$).

The smoothness term is quite similar to the one used in [16] and its goal is to encourage neighboring pixels in the preferred image to have similar depths. The smoothness term is defined as follow:

$$E_{smooth}(f) = \sum_{\{p,q\} \in \mathcal{N}} V_{\{p,q\}}(f(p), f(q)) \tag{4}$$

This term involves the notion of neighborhood: we assume that there is a neighborhood system on pixel

$$\mathcal{N} \subset \{\{p,q\} \mid p,q \in \mathcal{P}\}$$

This can be the usual 4-neighborhood system: pixels $p = (p_x, p_y)$ and $q = (q_x, q_y)$ are neighbors if they are in the same image and $|p_x - q_x| + |p_y - q_y| = 1$.

In [16], the function $V_{\{p,q\}}$ takes on the following form:

$$V_{\{p,q\}}(l_p, l_q) = \begin{cases} U_{\{p,q\}} & \text{if } l_p \neq l_q \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where the $U_{\{p,q\}}$ is the following non-decreasing function:

$$U_{\{p,q\}} = \begin{cases} 3\lambda & \text{if } \Delta I(p,q) < 5 \\ \lambda & \text{otherwise} \end{cases} \tag{6}$$

In order to obtain a smooth reconstruction that preserves discontinuities, we chose to follow a particular strategy in the use of the smoothness term. In fact, it is well-known that graph cuts techniques often yields flat and blocky results. This may not be important for disparity maps, but it is crucial for shape reconstruction. In order to avoid this problem, we make a first cycle of the reconstruction algorithm with a limited set of labels, in order to rapidly reach a value of the energy that is close to the local minimum that could be reached at convergence with the original algorithm. This corresponds to a good approximation of the position of the 3D-points, which can be improved with a second cycle at twice the resolution, where we change the function $V_{\{p,q\}}$ defined in (5) with this new function:

$$\hat{V}_{\{p,q\}}(l_p, l_q) = \begin{cases} U_{\{p,q\}} & \text{if } |l_p - l_q| > z\_threshold \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

In fact, this function relaxes the penalty mechanism of the smoothness term, giving a 0 penalty not only to the neighboring pixels that lie at the same depth but also to the ones that stay sufficiently close to each other. The idea is supported by the fact that after the first cycle of the algorithm, only some of the pixels are approximatively well positioned in 3D-space by the consistency measure given by the data term, while the other pixels' locations are only decided by the smoothness term. This term, in fact, forces them to lie at the same level of the neighboring pixel, resulting in flat blocks. Thus, relaxing the constraint imposed by the first smoothness term, neighboring pixels have greater chance to occupy adjacent depths correctly.

4.3 Graph cuts Algorithm

Thanks to our energy redefinition the results obtained from the standard graph cuts algorithm (as defined in [16]) are much more accurate. As shown in the next paragraph, further depth map optimization guarantees high fidelity in the reconstructed data.

4.4 Depth map optimization

Even though the graph cuts algorithm is able to reconstruct an accurate depth map, it works only with a limited set of depths and, therefore, it introduces a considerable quantization error in the positioning of each one of the 3D points. in order to overcome this problem, it is necessary to adopt an optimization step which produces more regular depth maps. The output of this process is a new depth map, where the discontinuities are preserved while the other parts turn out to be smoother.

In order to do so, we consider the depth map as a functions of two variables defined on the preferred image and we apply a series of 2D filters to it. In particular, we start with a median filter to eliminate possible outliers and then we apply a dithering technique: some white noise is added to the depth function and, then, a low pass filter is used to reduce depth quantization error and obtain a smoother map. In order to preserve discontinuities, the 2D low-pass filter keeps the information needed from the neighbors of a pixel only if the depth distance is below a certain threshold. The size of the filter windows and this threshold are empirically chosen on the basis of the current reconstruction.

4.5 Mesh triangulation

From the previous section we learnt how to compute a depth map from a set of images of the interested object. We also said that every map can be seen as a 2D function defined on the preferred image. Starting from this point, we can easily implement a triangulation algorithm that produce a mesh from a depth map on the basis of the neighboring pixels. Consider four neighboring points and the six possible connection shown in figure 13:

Figure 13: *Six possible configurations for the creation of triangles from four neighboring points.*

when two neighboring pixels have depths differing by more than some threshold, there is a step discontinuity. The threshold is determined directly by the human operator, as the maximum depth difference which has to be considered a surface discontinuity. If a discontinuity is present, a triangle should not be created. Therefore, for four neighboring pixels, we only consider 3D-points that are not along discontinuities. If three of them satisfy this condition, a triangle will be created in one of the last four style in figure 13. If none of the four are along a discontinuity, two triangles will be created, and the common edge will be the one with the shortest 3D distance, as shown in the first two styles in figure 13.

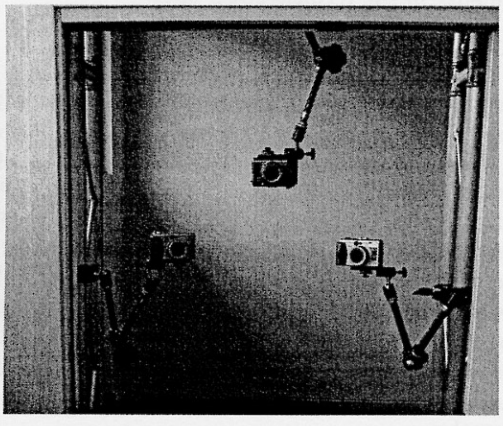

Figure 14: *The acquisition system. Three cameras placed around a gate.*

Repeating these steps for every mesh will lead to a volumetric function whose zero level set locates the object surface. The resulting object can be seen as a sort of convex hull obtained by linking together the meshes and taking only the part of the 3D space contained in their intersection.

To get the real time savings from the filtering the additional features should be:

- fast in extraction;
- fast in their usage for filtering condition computing;
- pre-computed in the database

If several filtering features are proposed then they are used in a cascade manner.

Denote by $f(x)$ the filtering feature computed as a function f for feature x.

As a rule the additional feature $f(x)$ is a scalar feature. In the vector case the bounding of features is performed in vector intervals instead of ordinary intervals.

Let $f_{min}(c,h), f_{max}(c,h)$ denote the lower and the upper bounds of filtering features $f(x)$ for all original features x which belong to the interval $[c, c+h)$.

Let also $f_0(c_{query}), f_1(c_{query})$ denote the lower and the upper bounds for $f(x)$ for all x from intervals which successfully passed filtering performed so far in the searching process for elements located near to c_{query}.

Feature rejection condition for $x \in \mathbb{R}^d$ has the form:

$$f(x) \notin [f_0(c_{query}), f_1(c_{query})],$$

Interval rejection condition for $[c, c+h)$:

$$[f_{min}(c,h), f_{max}(c,h)] \cap [f_0(c_{query}), f_1(c_{query})] = \emptyset$$

Remarks:

- The bounds $f_{min}(c,h)$ and $f_{max}(c,h)$ are joined to interval id dictionary when the identifier is inserted into the dictionary.
- The filtering feature $f(x)$ is joined to the feature vector x when x is inserted into the system.
- The updating technique for the acceptance interval $[f_0, f_1]$ depends strongly on the filtering function f and will be specified for each introduced f.

4.6 Experimental Results

The proposed algorithm has been applied to a variety of test images (of faces) acquired with a calibrated trinocular camera system. The acquisition system is based on three synchronized cameras Powershot G3 from Canon as shown in figure 14. The acquired images are in 2272×1704 JPEG format and, after face segmentation, the area that is actually useful for 3D reconstruction uses about 1MPixel of the 4 that are available.

In figure 15 we show the three segmented facial images and the final 3D model. The reconstruction time is about 3 seconds on a Pentium IV 3GHz. In particular, using the described approach we can obtain a very good depth estimation of difficult facial zones were uniform skin color and highly non-lambertian reflectance generate ambiguity in depth estimation. The parameters K of equation (3) and λ of equation (6) are determined heuristically from a set of facial images. Anyway, we observed that the estimated values gives good values for every facial image analyzed. The

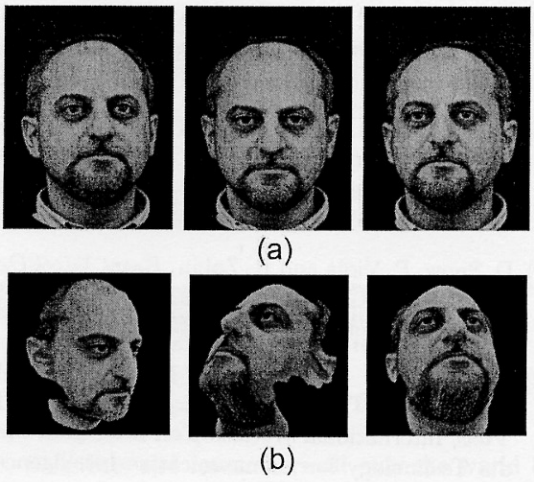

Figure 15: *A depth-map from a face: a) the acquired images. b) three different views from 'virtual observers' of the depth-map*

parameters can be varied to gain some insight about the algorithm: for big values of λ the smoothness dominates the correlation, resulting in a map with many flat blocks of pixels, whereas little values of λ yields to an irregular depth map with many wrong discontinuities. In our experiment, we chose the values $K = 30$ and $\lambda = 5$.

5 3D eigenfaces method

Depth maps enable representation of faces as *cloud of N points* together with associated color information. Typically such spatial set contains more than 10^5 points in 3D space ($N > 10^5$) and it is too complex as a 3D model for face recognition [22].

It appears that the collection of *head clouds,* obtained for a large group of L persons ($L > 50$), and considered as a set of points in a high N dimensional space can be approximated by a hyperplane (subspace) of relatively low dimension M (typically $M \approx 50$). One of possible linear algebraic bases spanning this hyperplane is obtained by Principal Component Analysis (PCA – cf. [23]) as eigenvectors of covariance matrix for the given training set of points.

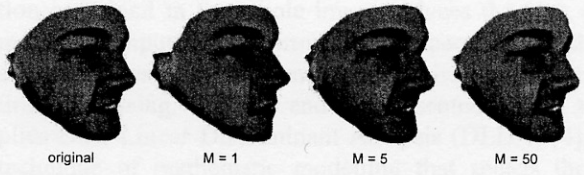

Figure 16: *2D views from reconstructed 3D model.*

In case of 2D facial images PCA eigenvectors are called eigenfaces [24] and by the analogy for 3D facial cloud of points, PCA eigenvectors are recognized as 3D eigenfaces [25]. Having M 3D eigenfaces $F_1, \ldots, F_M$ and average face F_0, any 3D face F can be approximated by a linear combination of 3D eigenfaces where coefficients α_i are appropriate dot products (cf. fig. 16):

$$F \approx F_0 + \sum_{i=1}^{M} \alpha_i F_i, \quad \alpha_i \triangleq (F - F_0)^t F_i$$

3D eigenfaces are built by Singular Value Approximation [26] (SVA) what allows to avoid building prohibitively large covariance matrices. In the current experiments each 3D eigenface consists of the depth and the color maps which are next used to compute *color cloud of points.*

6 CONCLUSIONS AND FUTURE WORK

In the paper a concept of advanced 3D face recognition framework was split into a few basic steps, namely face localization, facial feature extraction, 3D face model reconstruction from a few 2D images and 3D eigenface method. All steps were completed and the results on real facial examples were presented.

The main advantage of the proposed framework is its fast processing time. The most time consuming step that is the 3D face model building takes less than 3 seconds at Pentium 4 3GHz 1GB RAM for image size about one milion pixels.

The future research basic need is gathering large database of 3D face models to be able to check the 3D eigenfaces method performance in recognition as well as develop other recognition methods to fully utilize the 3D facial information.

Acknowledgment: The work presented was developed within VISNET, a European Network of Excellence (http://www.visnet-noe.org), funded under the European Commission IST FP6 programme.

References

[1] R. Chellappa, C. L. Wilson, S. Sirohey, *Human and Machine Recognition of Faces: A Survey,* Proceedings of the IEEE, 83(5):705-740, May 1995.

[2] M.-H. Yang, D. Kriegman, and N. Ahuja, *Detecting Faces in Images: A Survey,* IEEE Trans. on PAMI, vol. 24, no. 1, pp. 34-58, 2002.

[3] P. Viola P, M. Jones, *Rapid Object Detection using a Boosted Cascade of Simple Features,* Computer Vision and Pattern Recognition, 2001.

[4] Y. Freund, R.E. Schapire, *A decision theoretic generalization of on-line learning and an application to boosting,* Journal of Computer and Systems Sciences, 55(1):119-139, August 1997.

[5] W. Skarbek, K. Kucharski, *Tutorial on face and eye detection by AdaBoost method*, Special VISNET Session at Polish National Conference on Radiocommunications and Broadcasting (KKRRiT 2004) Warsaw, Poland, June 2004.

[6] W. Skarbek, K. Kucharski, *Image Object Localization by AdaBoost Classifier*, International Conference on Image Analysis and Recognition (ICIAR 2004), Porto, Portugal, Sept. 29 - Oct. 1, 2004.

[7] K. Kucharski, *Optimization of Face Detection with AdaBoost Cascade Algorithm*, Workshop on Image Analysis for Multimedia Interactive Services WIAMIS'05, Montreux, 2005.

[8] L. Zhang, *Automatic adaptation of a face model using action units for semantic coding of videophone sequences*, SPIE Conf. on Digital Compression Technologies and Systems for Video Communications, vol. 2952, Oct. 1996, pp.21-28.

[9] B. Cao, S. Shan, W. Gao and D. Zhao, *Localizing the iris center by region growing search*, Int. Conf. on Multimedia and Expo, vol. 2, 2002, pp. 129-132.

[10] G. Chow and X. Li, *Towards a system for automa-tic facial feature detection*, Pattern Recognition, vol. 26, no. 12, 1993, pp. 1739-1755.

[11] M. Kass, A. Witkin and D. Terpozopoulus, *Snake: active contour models*, Int. Jounal of Computer Vi-sion, 1988, pp. 321-331.

[12] A. Yuille, P. Hallinan and D. Cohen, *Feature extraction from faces using deformable templates*, Int. Jounal of Computer Vision, 8:2, 1992, pp. 99-111.

[13] Y. Sheng, A.H. Sadka and A. Kondoz, *Fast and automatic facial feature extraction for 3D model-based video coding*, Proc. of the 4th WIAMIS, London, April 2003, pp.387-390.

[14] R. C. Gonzalez and R. E. Woods, *Digital image processing*, Prentice-Hall, Inc, 2002.

[15] M. Kampmann, *Estimation of the chin and cheek contours for precise face model adaptation*, Proc. of Int. Conf. on Image Processing, Santa Barbara, CA, pp.300-303, 1997.

[16] V. Kolmogorov and R. Zabih, *Multi-camera Scene Reconstruction via Graph Cuts*, In European Conference on Computer Vision, 2002.

[17] V. Kolmogorov and R. Zabih, *Computing Visual Corrispondence with Occlusion via Graph Cuts*, In International Conference on Computer Vision, 2001.

[18] V. Kolmogorov and R. Zabih, *What energy functions can be minimized via graph cuts?*, In European Conference on Computer Vision, 2002.

[19] Y. Boykov, O. Veksler and R. Zabih, *Fast Approximate Energy Minimization via Graph Cuts*, IEEE Trans. on PAMI, 2001.

[20] Y. Boykov, O. Veksler and R. Zabih, *Markov Random Fields with efficient approximations*, IEEE Conference on Computer Vision and Pattern Recognition, 1998.

[21] D. Snow, P. Viola and R. Zabih, *Exact Voxel Occupancy with Graph Cuts*, In Proc. Computer Vision and Pattern Recognition Conf., 2000.

[22] W. Skarbek, K. Ignasiak, M. Morgos, M. Tomaszewski, *Towards 3D Face Model from 2D View*, International Workshop on Intelligent Media Technology for Communicative Intelligence, Warsaw, 2004.

[23] I. T. Jolliffe I.T, *Principal Component Analysis, Second Edition,* Springer, 2002.

[24] M. Turk, A. Pentland, *Eigenfaces for Recognition*, Journal of Cognitive Neuroscience, 3(1): 71-86, 1991.

[25] V. Blanz, S. Romdhani, T. Vetter, *Face Identification across Different Poses and Illumination with a 3D Morphable Model,* Proc. FG'02, pp.202-207, 2002.

[26] G. Golub, C. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, 1996.