EXTRACTION AND TRACKING OF THE EYELIDS

Montse Pardas

Bell Labs, Lucent Technologies, Murray Hill, NJ, USA
Universitat Politecnica de Catalunya, Barcelona, Spain

ABSTRACT

Facial analysis is used in a number of applications like face recognition systems, human-computer interaction through head movements or facial expressions, or model based coding. In all these applications a very precise extraction of the feature points corresponding to the eyes is necessary. In this paper we present a method for automatic extraction and tracking of the eyelids. First the images are filtered in order to enhance the interesting features. Then a dynamic programming algorithm is applied to extract the precise shape of the eyelids. Although restrictions are applied on the possible shapes, they are not modeled as ellipse arcs. This allows extracting the corner of the eyes with higher precision. Once the eyelids are extracted for the first image, a new robust tracking algorithm based on snakes is applied to track them along the sequence, which allows large motion of the eyes and rotation and zooming of the head.

1. INTRODUCTION

Many systems have been presented in the literature that require a precise location of the eyelids, whether for still images or for sequences. Among them we can cite face recognition based on facial features [3], facial analysis for emotion or gesture estimation [5], or facial analysis for model adaptation in a model based video coding system [15]. For instance, many systems which aim at computing the 3D motion of the head use as feature points for this computation the relative position of the corner of the eyes [8]. For this kind of applications it is necessary to have a very precise detection and the tracking has to be robust to large local motion (eye blinking is the most frequent motion) and to rotation and translation of the head. However, most of the systems proposed for eye detection and tracking are based on simple models or templates of the eyes [18][4]. These models usually assume that the eye can be described by two semi-ellipses, which is true for frontal faces with open eyes, but leads to errors when dealing with a broader class of images (see Figure 2 for examples).

We distinguish in this paper two procedures: one for the detection of the eyes in the first image and another one for tracking. For the detection step the eyes are modeled with a more general template which simply assumes that the eyelids are two curves, the upper with only one maximum and the lower with only one minimum. To find these curves a minimal path is searched on a previously filtered image, using a dynamic programming algorithm. For tracking we propose a method based on active contours or snakes which introduces motion estimation in the energy minimization procedure.

The resulting method is able to detect and track eyes with various shapes and supports blinking, rotation and translation of the head.

This paper is organized as follows. In Section 2 we will present the eye detection procedure. Section 3 describes the tracking and Section 4 presents the results. Finally we summarize our major findings.

2. EYE DETECTION

2.1 Rough location of the eyes

Before the precise detection of the eyelids is carried out a rough location of the eyes position is necessary. Many systems can be used for this aim. We have used the one presented in [12]. First, the face is extracted from the scene using [16]. Then, a morphological filtering is applied to extract contrasted components of a given size. Among these components we can find the eyes, mouth and usually the eyebrows. They will be identified using a priori knowledge about geometry of the searched features. That is, we look for a set of components following a specific spatial configuration. This procedure allows identifying the location of the eyes, eyebrows and mouth.

2.2 Definition of the search area

The minimal paths to define the eyelids are searched in the area where the eyes are presumably located. However, it is necessary to restrict this area as much as possible in order to minimize the necessary computations. For this aim, the dark contrasted components (which usually include the pupils and eyelids) are combined with the white contrasted components (the white of the eyes). These two sets of components define the area where the search will take place.

The dark contrasted component of the eye which has been extracted in the previous step to locate the different features is used. Then, the white contrasted component is computed with a top hat [14]. For the kind of images that we have used, CIF images (350x286 pixels) in which the face occupies most part of it, a 5x5 structuring element has been used. A threshold is applied on this top hat image, and those pixels that are larger than this threshold are selected as belonging to the white area of the eyes. This result is going to be used for two aims. First, to define with higher precision the limits of the search area for the minimal path. And second, to avoid these areas in the minimal path computation. For this aim the original image is modified so that all the paths which cross this white contrasted component have a maximal cost.

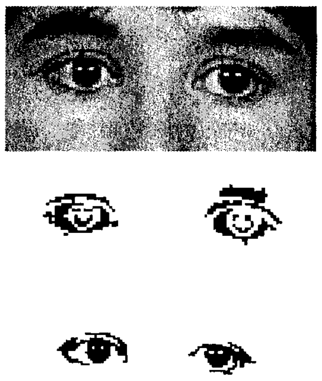

Figure 1. Original, dark contrasted components and clear contrasted components

2.3 Minimal path algorithm

The minimal path algorithm used to find the eyelids was already used in [12] to find the outline of the face. To apply this algorithm the two extreme positions (that is, the two corners of the eyes in this case) are necessary. As we only have approximate knowledge of the position of these corners we will make an exhaustive search for them. That is, the minimal path algorithm will be applied for every couple of candidate corner points. The couple of corner points leading to the minimal path will be selected as the correct one.

A first approximation to the eyes corner points is extracted using deformable line templates [2]. Specific patterns corresponding to the eye shape are searched for within the selected eye zone. The patterns used allow certain deformation, so that the detected eyes do not need to have a size or opening previously defined. The pattern consists of 3 ordered segments. Each segment is defined by an ordered set of elements. Segments cannot deform, but the best configuration of the 3 segments is found inside the detected eye component, so that it better matches the upper eyelid line in the original image. We will consider as candidates all the pixels in a small area around the corners extracted using this approximation. In the following we will describe the minimal path algorithm.

Let us call the right and left corners A and B. Two minimal path between them are computed, using dynamic programming algorithms [17], one for the upper eyelid and the other for the lower. The knowledge about the position and shape of the eyelids is used in order to restrict the possible paths. This knowledge consists of the facts that the path must be a descending path up to a horizontal position around the mid point between A and B, and ascending after this point for the lower eyelid and viceversa for the upper eyelid.

Let us call the image where the algorithm is going to be applied I. The minimal path P is defined as a path between A and B which minimizes the cost $C_I(P)$, where $C_I(P)$ is the sum of all the edge values of the path, and which fulfills the previous condition. The transition value between two neighboring pixels p and q is $T_I(p,q)=I(p)+I(q)$. The distance d_I between two pixels p and q is defined as $d_I(p,q)= min\{C_I(P), P$ path between p and $q\}$. The steps that are followed to compute the optimum path are the following:

- Initialize distance $d_I(A,p)=0$ if $p=A$ and *Inf* otherwise.
- Initialize distance $d_I(B,p)=0$ if $p=B$ and *Inf* otherwise.
- Scan the image in raster order, and assign $d_I(A,p)= min\{d_I(A,q)+T_I(p,q),\ q \in N^+(p)\}$, where $N^+(p)$ denotes the neighbors of pixel p scanned before p in a raster scan of image I.
- Scan the image from right to left and top to bottom, and assign $d_I(B,p)= min\{d_I(B,q)+T_I(p,q),\ q \in N^-(p)\}$, where $N^-(p)$ denotes the neighbors of pixel p scanned before p in the defined scanning order.
- A pixel C is searched which minimizes the distance $d_I(A,C)+d_I(B,C)$. This search is only made in a specific (central zone between A and B). Once this central position has been found, the whole path is defined backtracking the minimal path from C to A and B.

This algorithm provides the minimal lower path from A to B. In a similar fashion the minimal upper path from A to B is computed. Its costs are added, and the couple (A,B) which produces the lower cost is selected, and the minimal paths between them are taken as the eyelids.

The advantage of this algorithm is that it constraints only slightly the shape of the eyes, thus allowing a more precise location. Errors in the localization of the eyes occur when the size of the eyes is outside the margins allowed in the initial templates.

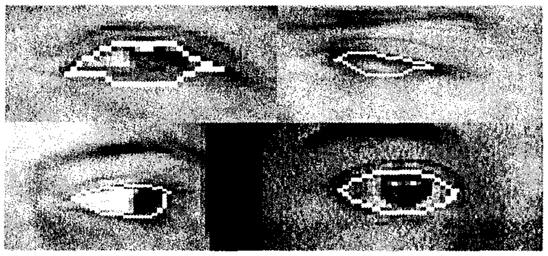

Figure 2. Examples of eye detection

3. EYE TRACKING

The eyes are tracked using an adaptation of the active contour algorithm (snakes) for tracking. Conventional snakes approaches for tracking ([9], [10], [6]) initialize the current frame snake with the snake obtained in the previous frame and then optimize this result considering only the current frame information. In our approach [13] motion estimation is embedded in the energy minimization process of the snake. This is possible using a dynamic programming approach for this minimization. This allows for large displacements of the contour and for a more robust tracking.

3.1 Snakes

In the discrete formulation of active contour models the contour to be tracked is represented as a set of snaxels $v_i=(x_i,y_i)$ for i=0,...,N-1, where x_i and y_i are the x and y coordinates of the snaxel *i*, and its energy, which has to be minimized, is defined by:

$$E_{snake}(v) = \sum_{i=0}^{N-1} \left(E_{\text{int}}(v_i) + E_{ext}(v_i) \right) \qquad (1)$$

We use a discrete approximation of the second derivative to compute E_{int}.

$$E_{\text{int}} = \left| v_{ss}(s) \right| \approx \lambda \left| v_{i-1} - 2v_i + v_{i+1} \right| \qquad (2)$$

This is an approximation to the curvature of the contour at snaxel i, if the snaxels are equidistant. We will force the snaxels to be equidistant when the snaxel is initialized in the first image. λ is a constant introduced to select the relative importance between the internal and external forces.

Actually, this is only valid for snaxels which are not corners of the contour. In the case of corners the energy has to be low when this second derivative is high. We use:

$$E_{\text{int}}(v_i) = \lambda\left[\beta_i \left|v_{i-1} - 2v_i + v_{i+1}\right| + (1-\beta_i)\left(\mathbf{B} - \left|v_{i-1} - 2v_i + v_{i+1}\right|\right)\right] \qquad (3)$$

where β_i is set to 1 if v_i is not a corner point and to 0 if it is. *B* represents the maximum value that the approximation to the second derivative can take.

The purpose of the term E_{ext} is to attract the snake to desired feature points or contours in the image. In this work we have used the mean value of the image (*I(x,y)*), along the contour from v_i to v_{i+1}, as the eyelids can generally be modeled by a dark line. Besides, we have approximated the contour between v_i and v_{i+1} by a straight line.

Thus, the E_{ext} at snaxel v_i will depend only on the position of the snaxels v_i and v_{i+1}. That is,

$$E_{ext}(v_i) = E_{cont\, v_i - v_{i+1}} = f(I, v_i, v_{i+1}) \qquad (4)$$

Taking into account the local dependencies of the Energy of the snake, we can express it as:

$$E_{snake}(v) = \sum_{i=0}^{N-1} E_{\text{int}}(v_{i-1}, v_i, v_{i+1}) + E_{ext}(v_i, v_{i+1}) = \sum_{i=0}^{N-1} E_i(v_{i-1}, v_i, v_{i+1}) \qquad (5)$$

This energy, for open snakes, can be computed recursively using a Dynamic Programming algorithm [2]. We have followed the approach proposed in [7] that performs the minimization for closed snakes in two open contour optimization steps [13].

3.2 Introducing motion estimation

To perform the optimization in the DP approach we need to have for every snaxel v_i a finite (and hopefully small) number of candidates. The computational complexity of each optimization step is $O(nm^3)$, where *n* is the number of snaxels and *m* the number of candidates for every snaxel. Thus, it is very important to maintain m low.

Different solutions to select these candidates that are not based in motion estimation are proposed in [2], [6] and [7]. The solution we propose uses motion estimation in order to select the search space for every snaxel.

A small region around every snaxel is selected as basis for the motion estimation. The shape of this region is rectangular and its size is application dependent. This region should be small enough so that its motion can be approximated by a translational motion. The compensation error for all the possible displacements *(dx,dy)* of the block in a given range is computed as:

$$MCE_{vi0}(dx, dy) = \sum_{j=-Ry1}^{j=Ry2} \sum_{i=-Rx1}^{i=Rx2} \left| I(x_0 - i, y_0 - j) - I(x_0 - i - dx, y_0 - j - dy) \right|^2 \qquad (6)$$

Being (x_0, y_0) the *x* and *y* coordinates of the snaxel v_i in the previous frame, which we have called v_{i0}. The region under consideration for the motion estimation is centered at the snaxel and with size *(Rx1+Rx2)* in the horizontal dimension and *(Ry1+Ry2)* in the vertical dimension.

The range for *(dx,dy)* determines the maximum displacement that a snaxel can suffer. The matrix $MCE_{vi0}(dx,dy)$ is stored for every snaxel, and the *m* best results are selected as possible new locations for snaxel v_i.

The DP algorithm is now applied considering as candidates for every snaxel those locations that have been selected by the motion estimation algorithm.

Besides, we have introduced a new term in the external Energy. This new term introduced in the E_{ext} improves the tracking capabilities of the algorithm. It is a memory term which corresponds to the compensation error obtained in the motion estimation. In this way preference is given to those positions with the smaller compensation error. That is, the energy will be lower in those positions which texture is most similar to the texture around the position of the corresponding snaxel in the previous frame. Thus, the external energy will be composed of two terms, the one which makes the snake be attracted by contours of the image, and this new one. Its expression will thus be:

$$E_{ext}(v_i) = \gamma E_{cont\, v_i - v_{i+1}} + (1-\gamma) MCE_{v_{i0}}(v_i) \qquad (7)$$

The constant γ can be set depending on the strength of the contour that is being tracked. If it is a strong contour we will make γ approach 1. Otherwise, we will give more importance to the Motion Compensation Error term.

3.3 Tracking of the eyelids

A closed snake with two corner points will be defined to track the eyelids. This snake is built by selecting a small percentage of the pixels that describe the contour, as obtained in Section 2. It is important to take as snaxels those position which are corners of the contour and mark them as corners. Besides, the number of snaxels selected should be large enough so that if there are any errors in the motion estimation, they can be neutralized by neighbor snaxels.

Motion compensation errors $MCE_{vi0}(dx,dy)$ are computed for every snaxel v_{i0} within a given range of allowed displacements for *(dx,dy)*. The size of the block that we have used is 4x4 for the images shown in the examples, which are 350x286 pixels large. The shape of the blocks has been adapted because there are big changes in the texture inside the eyes due to their closure or opening. Thus, we have placed the block above the snaxels in the upper eyelid and below the snaxels in the lower eyelid.

Those pixels *(x_0+dx, y_0+dy)* in the successive frame corresponding to the displacements *(dx,dy)* which produced a smaller compensation error are selected as possible candidates for snaxel v_i in this frame

Once the candidates for every snaxel have been chosen, the two steps DP algorithm is run. The constant λ has been set heuristically to 2 and $\gamma$to 0.5.

Examples of the results obtained for different sequences of the M2VTS face database [11] are presented in Figure 2. We show 2 images of each sequence, among which there is rotation of the head and blinking of the eyes. Larger rotations, where the eyes are partially occluded are not supported by this method.

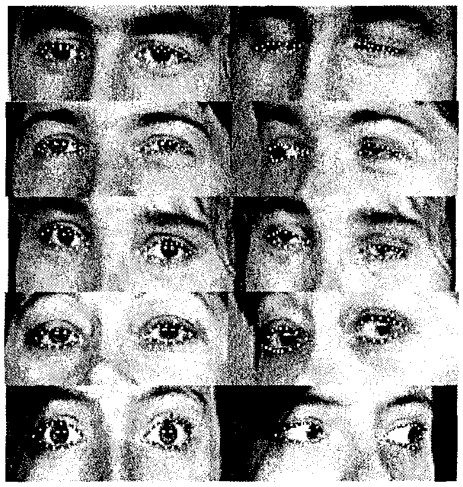
Figure3. Examples of eye detection and tracking. The dots represent the snaxel position.

4. CONCLUSIONS

We have presented a method for automatic extraction and tracking of the eyelids. The extraction is based in morphological filtering and a minimal path algorithm that only restricts partially the shape of the eyes, thus allowing different eye shapes which cannot be modeled by two semi-elipses. The tracking of the detected contour has been performed with a new active contour algorithm that uses motion estimation for a more robust tracking. This approach to the tracking reduces the computational cost with respect to other DP implementations, as only a small number of pixels have to be considered as candidates for every snaxel. It allows to track contours which are not the global minimum, as the snake is actually tracking the texture around the snaxel and it is much more robust to deformations and large motion of the contours than the classical approaches.

5. REFERENCES

[1] J. Ahlberg, "Extraction and Coding of Face Model Parameters, Linkoping Studies in Science and Technology, Thesis No. 747, Dept. of Electrical Engineering, Linkoping University, Sweden, 1999.

[2] A. Amini, T. Weymouth and R. Jain, "Using Dynamic Programming for Solving Variational Problems in Vision", IEEE PAMI, Vol. 12, No. 9, September 1990.

[3] Brunelly R., Poggio, T. Face recognition: *Features versus templates*, IEEE PAMI 15 (10), 1042-1052.

[4] C. Colombo, A. Del Bimbo, Real Time head Tracking form the deformation of eye contours using a piecewise affine camera, Pattern Recognition Letters 20, pp. 721-730, 1999.

[5] Essa I., Coding, Analysis, Interpretation and Recognition of Facial Expressions, IEEE PAMI 19 (7) July 1997.

[6] D. Geiger, A. Gupta, L. Costa and J. Vlontzos, "Dynamic Programming for Detection, Tracking, and Matching Deformable Contours", IEEE PAMI, Vol. 17, No. 3, March 1995.

[7] S. Gunn and M. Nixon, "Global and Local Active Contours for Head Boundary Extraction", IJCV 30(1), pp. 43-54, 1998.

[8] T. Horprasert, Y. Yacoob, L. Davis, Computing 3-D Head Orientation from a Monocular Image Sequence, Proc. of SPIE, Vol. 2962, No.31, pp. 244-252, 1997.

[9] M. Kass, A. Witkin and D. Terzopoulos, "Snakes: Active Contour Models", IJCV, Vol. 1, No. 4, pp. 321-331, 1988.

[10] F. Leymarie and M. D. Levine, "Tracking Deformable Objects in the Plane Using an Active Contour Model", IEEE PAMI, Vol. 15, No. 6, June 1993.

[11] M2VTS http://www.tele.ucl.ac.be/M2VTS

[12] Pardas, M., Automatic Face Analysis for Model Calibration, Proceedings of the IWSNHC 1999, pp.12-15, Sept. 1999.

[13] Pardas, M., Sayrol, S., A new approch to active contours for tracking, to be submitted to Int. Conf on Image Processing, Vancouver, 2000.

[14] J. Serra," Image Analysis and Mathematical Morphology", Academic Press, New York 1982.

[15] A. Smolic, B. Makai, T. Sikora, Real Time Estimation of Long-Term 3D Motion Parameters for SNHC Face Animation and Model-Based Coding Applications, IEEE Trans. CSVT, Vol. 9, No2, pp. 255-263, March 1999.

[16] Vilaplana V., Marqués F. , Salembier P., Garrido L., "Region Based Segmentation and Tracking of Human Faces", Proc. of EUSIPCO-98, pp. 311-314, 1998.

[17] L. Vincent, "Minimal path algorithm for the robust detection of linear features in gray images", in Mathematical Morphology and Applications to Image and Signal Proc., Kluwer Academic Publishers, pp. 331-338, 1998.

[18] Yuille A., Cohen D., Halliman P. Feature Extraction from faces using deformable templates, IJCV, 9:104-109, 1999.