Image-based Face Position and Angle Estimation: "State of the Art"

C. Emiliano Solórzano Espíndola, carlosemiliano04@gmail.com

ETSEIB- Universitat Politècnica de Catalunya

***Abstract*— The present works present some of the current approaches in techniques for face detection and the posterior estimation of the angle. For each of this steps various algorithms can be combined in order to solve this problems. The first one can be treated as a computer vison problem and is desired to detect on a given image the region of interest that represents a face. Many systems are adapted to process the images with frontal type of faces only. Once the region that represents the face is isolated, the deviation angle can be computed with geometric algorithms using some biometric features.**

Keywords: face, algorithms, angle, computer vision

I. Introduction

Face detection is one of the most recurring subjects in computer vision, that's why there are many works published every year since the last few decades treating this subject, developing new algorithms depending on the goal of each one. This is because of the many applications that can be given to this, like face recognition, face tracking or create a 3D-model of a subject.

Also, face detection is still an area of active research since a completely successful approach or model has yet not be proposed to solve this problem. A human face is a dynamic object having high degree of variability in its appearance, which makes face detection a difficult problem in computer vision. In this field, accuracy and speed of identification is a main issue.

This problem however is a very easy task for a human, who can detect a face despite changes due to viewing conditions, emotional expressions, ageing, added artefacts, or even circumstances that permit seeing only a fraction of the face. Furthermore, humans are able to recognize thousands of individuals during their lifetime, meanwhile this algorithms are developed for a specific dataset.

One of this problems has to do with the relative angle of inclination of the face to the camera, some algorithms for recognition of a subject don't have rotational invariance, and this can be solved by either using a different algorithm for the matter or using some biometric features like the relative position and distance of the eyes, nose and mouth to re-align and rotate the image for further analysis.

II. Face Detection

Face detection is the process that determines the locations and sizes of human faces in arbitrary (digital) images. It detects facial features and ignores anything else. This is essentially a segmentation problem and in practical systems, most of the effort goes into solving this task.

Real-time face detection involves detection of a face from a series of frames obtained from a video-capturing device. While the hardware requirements for such a system are far more stringent, from a computer vision stand point, real-time face detection is actually a far simpler process than detecting a face in a static image. This is because unlike most of our surrounding environment, people are continually moving.

Methods for face detection can be classified as follows:

- Color
- Motion
- Edge-based
- Weak classifiers cascades
- A mixture of the above

Each of this classes has many algorithms developed to try to solve this problem. Some of this are:

A. PCA (color)

The color of human skin is distinctive from the color of many other natural objects, hence color is a very important feature that can be used for face detection. Analyzing the skin-tone color statistics, one observes that skin colors are distributed over a small area in the chrominance plane and the major difference between skin tones is intensity. Thus, the image is first converted into a color space capable of separating into a luminance channel and two chrominance components like the YCbCr or HSV color space.

With the remaining two components, a conditional Gaussian probability function is created where $p(w_{ij}|S)$ denotes the probability of belonging to the skin class *S*.

$$p(w_{ij}|S) = \frac{\exp\left[-\frac{1}{2}\left(w_{ij}-\mu_S\right)^T \Sigma_S\left(w_{ij}-\mu_S\right)\right]}{2\pi|\Sigma_S \quad|^{\frac{1}{2}}}$$

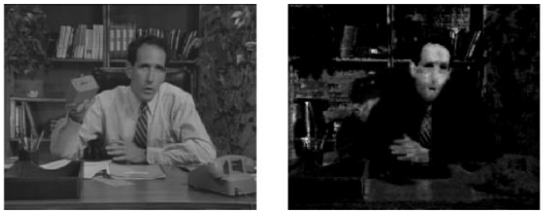

Figure 1: Original image and probability image

Now we can create a new image with the probability of each pixel belonging to the skin. The major benefit of using the skin probability image is that a facial region is enhanced compared to the background and the influence of different lightning conditions is reduced since the skin probability doesn't depends on it.

The central idea of principal components analysis is to find a low dimensional subspace (the feature space) which captures most of the variation within the data set and therefore allows the best least square approximation.

Given a set of training vectors $\{x\}$ (face samples) with sample covariance matrix Σ, the KLT basis can be computed by solving the eigenvalue problem

$$A = P^T \Sigma P$$

Where P is the eigenvector matrix of Σ and A is the diagonal matrix of eigenvalues. The orthogonal projection matrix P_M into the M-dimensional principal subspace ($M \ll N$) is given by the M eigenvectors corresponding to the largest eigenvalues. These eigenvectors ("eigenfaces") form the columns of the projection matrix P_M. The principal components vector ***y*** is obtained by projecting the image ***x*** into the face space:

$$y = P_M^T(x - \bar{x})$$

where $\bar{x}$ denotes the mean face image.

The approximation of a face using M "eigenfaces" is given by $x = P_M\, y + \bar{x}$. The residual reconstruction error $\in^2$

$$\in^2 = \|(x - \hat{x})\|^2$$

$$\in^2 = \|(x - \bar{x}) - P_M y\|^2$$

$$\in^2 = \|(x - \bar{x})\|^2 - \sum_{i=1}^{M} y_i^2$$

indicates how well the test pattern can be approximated in the face space. Thus, the "distance from face space" (DFFS) defined by equation can be used to determine if the image pattern represents a face.

B. Fleck and Forsyth algorithm (color)

In this case the process can be divided in two phases, the first one is skin detection and labeling and the second is find holes in the image that can represent the eyes.

In the first step we take the RGB image and from there extract the *log-opponent* chromacity representation, these can be represented as follows:

- $I = L(G)$
- $Rg = L(R) - L(G)$
- $By = L(B) - (L(G) + L(R))/2$

Where the L operand is defined as: $L(x) = 105 * (x + 1)$, this is in order to make *Rg* and *By* chromacity values, as well as *I* texture amplitude values independent of illumination.

After that we use the texture amplitude values to find regions of low texture information, since the skin has a very smooth texture. In those regions, we further select skin based on measures of hue and saturation, so as their color matches that of the skin

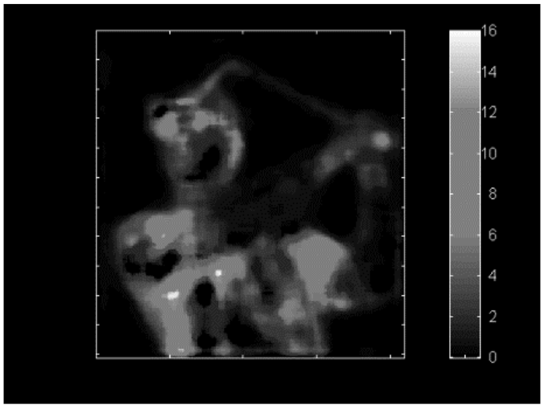

Figure 2. Texture map using the illumination values

Once that we have computed the texture map we can use a threshold to discriminate the parts of the image than we are interested in. The range that can describe the skin will depend also on the hue and saturation of the given image. If a pixel falls into the range it is marked as being skin in a binary map.

Once we have the binary map we can label each of the resulting groups that represent portions of skin, but first a dilatation operator is applied in order to close some of the holes in the resulting image. After that we subtract the original image without dilatation and find where the holes are and create a new image with the absolute value, this holes can represent the eyes or other facial features, Once we have done this we can find which of the labeled groups is the nearest to this holes, and thus where the face is.

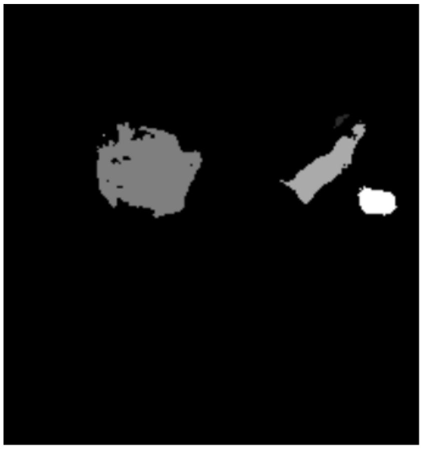

Figure 3. Labeled groups of skin. It can be seen that in the face group there are the holes of the nostrils that will give us where the face is.

C. Haar like features (edge based & weak classifier cascade)

The simple features are reminiscent of Haar basis functions. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. There are three kinds of features used:

- A two rectangle feature: the difference between the sums of the pixels within both rectangles. They have the same size and orientation.
- A three rectangle feature: calculates the sum within the two external rectangles and subtracts the sum of the central one.
- A four rectangle feature: calculates the difference between the sums of the diagonals.

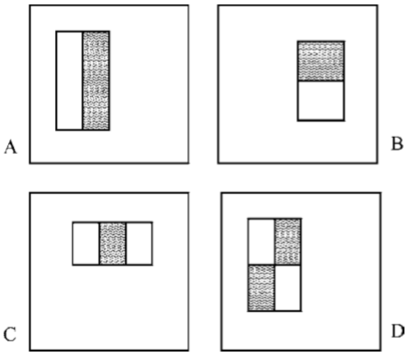

Figure 4. Rectangle features: A) Two rectangle vertical, B)Two rectangle horizontall, C)Three rectangle, D)Four rectangle

Now all possible sizes and locations of each kernel is used to calculate plenty of features. Given that the base resolution of the detector is 24 × 24, we end up with 160,000 features.

To speed up the process of calculating the features we can generate the integral image. This image at a given *x,y* location contains the sum of pixels above and to the left of *x,y*. It's defined as:

$$ii(x,y) = \sum_{x' \leq x, y\prime \leq y} i(x', y')$$

where *ii* is the integral image and *i* is the original image. Using the integral image any rectangular sum can be computed in four array references.

Among all the calculated features most of them are irrelevant. Even though each feature can be computed very efficiently, computing the complete set is prohibitively expensive. In order to find the most important features a variant AdaBoost is used for both, to select the features and to train the classifier. It does this by combining a collection of weak classification functions to form a stronger classifier. In the language of boosting the simple learning algorithm is called a weak learner.

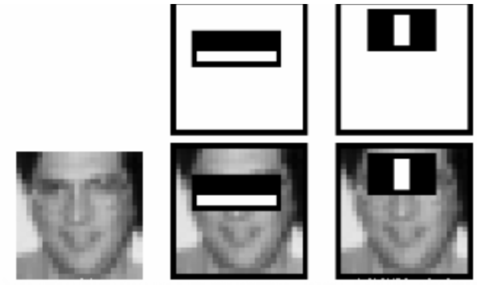

Figure 5. Some representative Haar-like features in a face

For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. But obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that best classifies the face and non-face images. The authors were able to reduce to around 6000 features to achieve 95% accuracy.

Now the classifier still has a window of only 24x24 that has to be applied to the whole image to find the regions where there is a face. But instead of checking every possible window, some first set of features can be applied and check the expected output, if they match another set of features is applied and so on until getting to the 6000 features, in case that it doesn't in one step, it is discarded and the algorithm moves to the next window. This concept is known as Cascade of Classifiers.

D. Bayesian Classifier and moments (color)

For the detection of the region that represent the face, the normalized RGB space (*rgb*) is first obtained, this allows to minimize the variation due to the illumination since the face is a highly curved surface. The skin was found that can be tracked with high efficiency using only the r and g components [7].

$$r = \frac{R}{R+G+B}, \quad g = \frac{G}{R+G+B}$$

Then a conditional probability densities for a pixel being part of the skin using the r and g channels histogram can be estimated using the *Bayes rule* as in [8]. Then we can estimate the probability of being skin $p(Pix \in Skin|r,g)$ given the values in r and g with some examples. A Gaussian distribution can be used for the estimation of the pixel's probability.

$$p(Pix \in Skin|r,g) = \frac{p(r,g|Pix \in Skin)p(Pix \in Skin)}{p(r,g)}$$

The face position tracking con be achieved using moments. The first moment represents the center of gravity and provides a robust estimation of the face position, the second moment can give an estimation of height, width and slant of the face. The first and second moment are used to normalize the position and orientation. The moments can be estimated given the formulas:

$$\mu_x = \frac{1}{S}\sum p(x,y) * x * G(x,y,\vec{\mu},C)$$

$$\mu_y = \frac{1}{S}\sum p(x,y) * y * G(x,y,\vec{\mu},C)$$

$${\sigma_x}^2 = \frac{1}{S}\sum p(x,y) * (x-\mu_x)^2 * G(x,y,\vec{\mu},C)$$

$${\sigma_y}^2 = \frac{1}{S}\sum p(x,y) * (y-\mu_y)^2 * G(x,y,\vec{\mu},C)$$

$${\sigma_{xy}}^2 = \frac{1}{S}\sum p(x,y) * (x-\mu_x)(y-\mu_y) * G(x,y,\vec{\mu},C)$$

Where G represents the Gaussian prediction of the given pixel and S the total sum of pixels.

After that we can estimate the important are to examine in the image and since we already we have the relative position of this specific frame we can compare in the case of a video input to measure some changes.

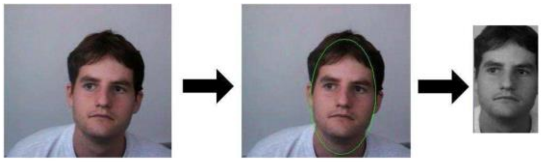

Figure 6. Extraction of the representative are of the image using first and second moments

III. Conclusions

Some of these techniques has been proved to work for the specific task that they were given. But some of them like the *Flecks and Forsyth* and *PCA* are used for still images since the process for computing them is accurate but slow and demands big processing power by the system.

In the case of *Haar-like features* is great for slow systems since the systems requirements are low and that's why it is a commonly used method today for cameras and mobile devices. The problem with this one is that it is trained using a database with low variety of angles to compare against.

The *Bayesian classifier* although is not as accurate as the other color methods, it can be processed in a faster way and have some tolerance to the light and the angle. Also the combination of this method with the moments can help to determine the relative position of the face and the important area to explore for the case of a live video input were we can compare this area with a new frame.

IV. Resources

[1] J. C. Terrillon, M. David, S. Akamatsu. (1998). *Automatic Detection of Human Faces in Natural Scene Images by Use of a Skin Color Model and of Invariant Moments*. In FG '98 Proceedings of the 3rd. International Conference on Face & Gesture Recognition (112). Washington, DC, USA : IEEE Computer Society .

[2] B. Menser and F. Mueller. (1999). *Face detection in color images using principal components analysis*. In Image Processing and Its Applications, 1999. Seventh International Conference on (Conf. Publ. No. 465), pp 620 - 624. Manchester: IET.

[3] Paul Viola, Michael J Jones. (2004). *Robust Real-Time Face Detection*. In International Journal of Computer Vision 57, pp. 137-154, Netherlands.

[4] Storring, M. (2004). *Computer Vision and Human Skin Colour* (Ph.D.). Faculty of Engineering and Science, Aalborg University.

[5] Stathopoulou, I., & Tsihrintzis, G. (2010). *Visual affect recognition*. Amsterdam: IOS Press.

[6] Ballester, F. (2016). Computer Vision and Face Recognition (Bachelor). Kymenlaakson Ammattikorkeakoulu.

[7] K. Schwerdt, J.L. Crowley. *Robust face Tracking using Color.* Automatic face and Gesture Recognition, pp 90-95, 2000.

[8] Gourier, N., Hall, D., & Crowley, J. (2004). Estimating Face Orientation from Robust Detection of Salient Facial Structures. *FG NET WORKSHOP ON VISUAL OBSERVATION OF DEICTIC GESTURES*.