Facial Feature Extraction

Thiago Sonego Goulart
Universidade Federal de Minas Gerais
thiagosg@dcc.ufmg.br

Abstract

In this work, we propose an algorithm based on the template matching technique to find the location of facial features. We assume the boundaries of the face are given within the image, the pictures are frontal and there is no occlusion. It works by finding the best match of a segmented template over the original image. To look for the best match, the template is both scaled and translated over a predefined local neighborhood, since the features we are looking for are not expected to be far from the original template position. Experimental results indicate that the segmented template is able to locate at least 50% of the given features, and even better results are achieved depending on the template used for each image.

1. Introduction

Facial feature detection plays an important role in human-computer interaction and psychological research, and it can be used as input for many applications, such as face identification and emotion recognition. This can be done from both video sequences and static images.

Face recognition can be used in many important scenarios, such as personal identification and access control. Security is perhaps the area with the most applicability for face recognition.

Facial expressions is an important issue in the communication process. With the automated feature extraction, it might be possible for intelligent computer systems to decide the actions to me made taking into account the mood of the user.

Various methods have been proposed for feature extraction, most of them relying on classification algorithms or neural networks based approaches. These techniques can be very efficient on this task, but they all need a massive preprocessing over a training set of images, from which the classifiers can build their knowledge base.

Very few researchers have worked with static templates for facial feature extraction, and the explanation is quite simple: if people have different faces, how can we create a perfect template to match every single faces we try? Well, we can't. But it is possible to work with many different templates to increase the probability that at least one of the templates is similar to every face.

In this work, we proposed a segmented template to handle facial discrepancies among the population.

2. Related Work

There are many related works to the face detection problem. One of the most relevant is [10], where they used the AdaBoost learning algorithm to reduce the search space of the features and combined classifiers which allows the background of the image to be quickly removed.

A different approach is discussed in [5]. They present a methodology for face tracking and recognition from video sequences.

In a recent work [9], they recognize facial expressions by analyzing the histogram of different regions of the face, and apply a Linear Programming technique to classify the expressions.

In [6], it is presented a method that relies on many techniques, including the template matching for the eyebrows and Hough transform for cheeks and chin detection. Also, mathematical models were used to find the relative position between eyes, nose and mouth.

In [11], they proposed and accurate method for finding the eyeballs, mouth corners and nose midpoint. They used the SUSAN [3] operator to extract the edge and corner points of the features.

3. Feature Extraction

The template matching technique consists in finding the best $I(i,j)$ coordinate which maximizes a desired function, where I is the input image. Some preprocessing is needed in order to minimize erroneous results of the process.

3.1. Template Processing

Our template is a previously generated grayscale image representing the borders of the features we are interested, so a pixel in the template must be, ideally, black or white. Due to image compression, this might not be true, so the pixel values are rounded to the nearest of these values (0 or 255).

After the binarization is done, the black pixels must be changed to white and vice versa. We want the border values to have a meaning, so they must have a value different than zero. This change will later affect the matching function that will be used.

Two basic templates[8][4] were first used and then simplified and merged to create other potentially better variants.

3.2. Image Processing

As this algorithm for feature extraction does not rely on any color pigment information, the color on our input images are not needed. The first step is to convert the original image into a 8-bits grayscale image.

Since the template is a binary image of borders, we are also interested on the borders of the original image. This can be done by applying an edge detector algorithm that gives as result a binary image with only the borders that are important for our features.

Obtaining the best borders is not a simple task. Every image taken suffers from noise, both from the environment from which the image is taken as from the camera itself. Also, even if the pictures were all noise free, a threshold must be carefully chosen to decide whether a border found by the algorithm is really relevant for our goals.

The second step consists in filtering the image to potentially decrease the number of false positives returned by the edge detector. This is done by applying two filters: the median filter, which removes the salt and pepper noise, and the gaussian filter, responsible for smoothening the fake borders, respectively. This order was empirically found to give better results after the processing.

After the image is filtered, it is ready to be used as input to the edge detector algorithm. The algorithm we used at this step is the Canny edge detector[2], a very robust and widely applicable technique. It returns an image containing the strongest borders of the face (and, of course, the entire image). At this point, we are ready to proceed to the next step, the matching of the two border images.

3.3. Template Matching

Now that we have the borders image *I* of the input and the template *T* ready, it is time to perform the most important operation: the template matching. This will give us the results we want, i.e. the best coordinate *(i,j)* of *I* where *T* can be overlayed.

To perform the matching, the dimensions of *T* must be less than or equal to the respective dimensions of *I*. The result of the matching is a matrix *R* whose size is the difference between the sizes of *I* and *T*. *R* is calculated by sliding *T* through *I* and storing the result of the overlapped patches by the Equation 1. Since the possible values on the images are 0 and 255, this equation sums the number of times that a white pixel of *I* matches with a white pixel of *T*.

$$R(x,y) = \sum_{x',y'} T(x',y') * I(x+x', y+y') \qquad (1)$$

After the matrix *R* is calculated, it is easy to find out which is the best position for *T*. The answer is the pixel *(i',j')* that maximizes the value among all the possible *R(i,j)*.

4. Results

We are interested in basically six features: two eyes, two eyebrows, nose and mouth. The informations we want to extract from the image is the exact location and scale of these features. Our algorithm was implemented in C and used the OpenCV[1] library.

The templates were segmented to allow individual feature processing. Each feature was manually enclosed by the smallest possible rectangle that contains the feature on the original template, and its bounds were hardcoded on the algorithm so we can extract them from the template. The template with its individual features can be seen in Figure 1.

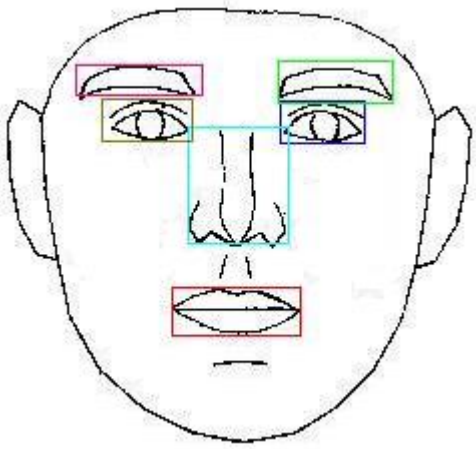

Figure 1. Original template with highlighted features.

The face boundaries used as input for our algorithm were found using the same templates that contains the features we are trying to find, but also taking into account the face and ears contours. Other methods could have been used, like the Haar-like feature recognition[7] (that is already implemented in OpenCV), but neither of the approaches is perfect.

After the face was found, the algorithm tries to find the best position for each of the features considering scale and translation. The features are matched with scale varying from 80% to 120% (with step 2%) of its original size. For each of these 20 values of scale, the rectangle containing the feature is translated on both vertical and horizontal directions within a distance of 20% of its length value, measured in pixels. Thus, it covers an entire local neighborhood with variant scale, providing a fine adjustment for the feature.

The values of R for smaller scales surely have a disadvantage over the bigger scales, since the matching is basically a counting over the pixels that matches. The solution to this problem was to scale the values of R according to the total area of the feature. This way, the values of different scales can be compared more fairly.

Once all done, the resulting features are then drawn over the borders of the original image to show the difference between the locations estimated a priori by the mask and the face boundaries. Some resulting images can be seen in Figures 2, 3, 4 and 5.

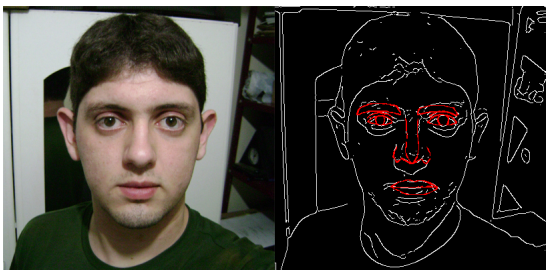

Figure 2. Great eyes, nose and left eyebrow estimation, mouth a few pixels away from ideal position and bad right eyebrow location.

These results were not compared to other published methods mainly due to the lack of a good metric, but the subset of features that each method tries to extract is also a limiting factor. This algorithm might not be as successful as some of the other previous techniques, but the results show there is a lot of room for improvements on almost every steps, from the template development to the matching equation.

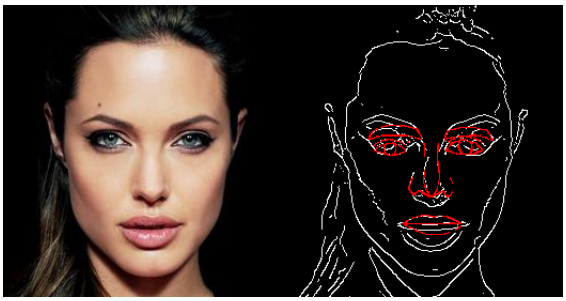

Figure 3. Great nose estimation and good mouth location, though not so good scaling. Poor eyes and eyebrows estimation.

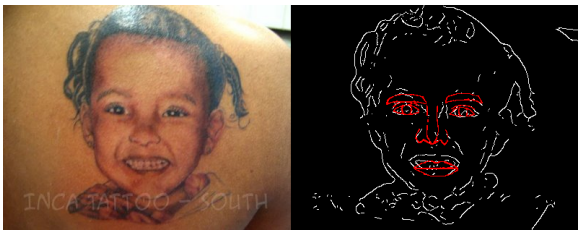

Figure 4. Picture of a tattoo. Good eyes, nose and right eyebrow estimation, but bad left eyebrow and mouth placement.

5. Conclusions

From the analysis of the resulting images, we can see that no set of features were perfectly extracted from the tested input. On the other hand, when the face boundaries were found within a reasonable accuracy, at least half of the features were correctly located. The nose is the feature with best overall accuracy, and the eyebrows, the worst.

Experiments show that the template is the most relevant factor for a good feature extraction. Unfortunately, it is a hard task to create templates that can be used to find features of almost every faces, since each face has its own singularities. One possible improvement over this could be to use many different templates for each feature. A single mouth template can't find both closed and opened eyes, for example, so this might be a promising future work.

An important point is that, depending on the size of the input image from which we want to extract the features, this approach can be used to real-time applications. With a template of 242x230 pixels, images of size up to 400x300 pixels can be processed in less than 100ms on a Core 2 Duo 2.16 GHz with 2 GB RAM.

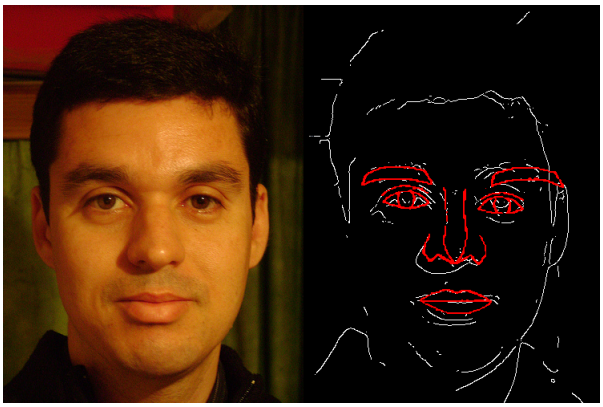

Figure 5. Great left eye and eyebrow, nose and mouth estimation, wrong right eye position and bad right eyebrow location.

References

[1] D. G. R. Bradski and A. Kaehler. *Learning opencv, 1st edition*. O'Reilly Media, Inc., 2008.

[2] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986.

[3] M. Hess and G. Martinez. Facial feature extraction based on the smallest univalue segment assimilating nucleus (susan) algorithm. In *Proceedings of the Picture Coding Symposium (PCS '04)*, San Francisco, California, USA, 2004.

[4] E. Holland. Marquardts phi mask: Pitfalls of relying on fashion models and the golden ratio to describe a beautiful face. *Aesthetic Plastic Surgery*, 32(02), January 2008.

[5] S. Z. Li and A. K. Jain. *Handbook of Face Recognition, 1st ed.* Springer, 2005.

[6] A. Nikolaidis, C. Kotropoulos, and I. Pitas. Facial feature extraction using adaptative hough transform, template matching and active contour models. In *13th International Conference on Digital Signal Processing Proceedings*, volume 2, pages 865–868, July 1997.

[7] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. *Sixth International Conference on Computer Vision*, pages 555–562, 1998.

[8] D. I. Perrett, K. A. May, and S. Yoshikawa. Facial shape and judgements of female attractiveness. *Nature*, 368(6468):239–242, March 1994.

[9] C. Shan, S. Gong, and P. W. McOwan. Facial expression recognition based on local binary patterns: A comprehensive study. *Image Vision Comput.*, 27(6):803–816, 2009.

[10] P. Viola and M. J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 57(2):137–154, 2004.

[11] C. T. Yuen, M. Rizon, W. S. San, and M. Sugisaka. Automatic detection of face and facial features. In *ISPRA'08: Proceedings of the 7th WSEAS International Conference on Signal Processing, Robotics and Automation*, pages 230–234, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS).