International Journal of Signal Processing, Image Processing and Pattern Recognition
Vol.9, No.2 (2016), pp.13-24
http://dx.doi.org/10.14257/ijsip.2016.9.2.02

Face Detection and Recognition in Color Images under Matlab

Deise Maia[1] and Roque Trindade[2]

[1]*Depatment of Computer Science, Federal University of Minas Gerais, Belo Horizonte, Brazil*
[2]*Department of Mathematical Sciences, State University of Southwest Bahia, Vitória da Conquista, Brazil*
deisemaia@best.com.br, roquetrindade@uesb.edu.br

Abstract

In this paper we describe our implementation of algorithms for face detection and recognition in color images under Matlab. For face detection, we trained a feedforward neural network to perform skin segmentation, followed by the eyes detection, face alignment, lips detection and face delimitation. The eyes were detected by analyzing the chrominance and the angle between neighboring pixels and, then, the results were used to perform face alignment. The lips were detected based on the analysis of the Red color component intensity in the lower face region. Finally, the faces were delimited using the eyes and lips positions. The face recognition involved a classifier that used the standard deviation of the difference between color matrices of the faces to identify the input face. The algorithms were run on Faces 1999 dataset. The proposed method achieved 96.9%, 89% and 94% correct detection rate of face, eyes and lips, respectively. The correctness rate of the face recognition algorithm was 70.7%.

Keywords: *Image processing, Skin segmentation, Eyes detection, Lips detection, Face recognition*

1. Introduction

A face detection and recognition system aims to reproduce one of the innate human abilities: recognizing characteristics of faces in different environments and associate them with the knowledge stored in their memory.

Many areas take advantage of face recognition systems, including entertainment (human-computer interaction and video games); information security (personal devices logon, internet security and database security); and surveillance (post-event analysis and investigation) [1].

Face recognition is an easy and straightforward task for humans, but it seems to be a challenge for computing systems due to the several environmental variables: face position and its distance from the camera(s), lighting conditions and shadows on the face, and some individual aspects, such as beards and hairstyles. However, some individual face features hardly change regardless of environment, and the face recognition is performed by analyzing the pattern found in different images of the same individual [1, 2].

As any other computing system, face recognition systems are developed to speed up the execution of tasks that could be performed by humans, with the benefit of a higher performance of the machines when running repetitive tasks that require attention and patience. Facial recognition in humans is more complex and precise than in any automatic system, because it involves the context of the social environment and the acquired knowledge throughout their lives. Nevertheless, during a video analysis, for example, the individual concentration may decrease along the time due to fatigue and distracting elements. This is not the case for automatic systems, which has a higher

ISSN: 2005-4254 IJSIP

capacity than humans regarding storing and retrieving facial data from its memory, even though the stored facial features are limited.

In this paper, we present techniques for face detection and recognition in color images under the software Matlab. The face detection algorithm was divided into the following steps: face localization in the input image; eyes and lips detection; face alignment and delimitation. They were implemented by using a neural network and analyzing the chrominance and standard deviation of the color of neighboring pixels on each input face. Then, the face recognition algorithm used a pattern classifier to compare the color distribution among different faces. The neural network used for skin segmentation was trained using images from Helen dataset [3] and the algorithms were run on Faces 1999 dataset [4], which contains 447 color images of 28 individuals in frontal position.

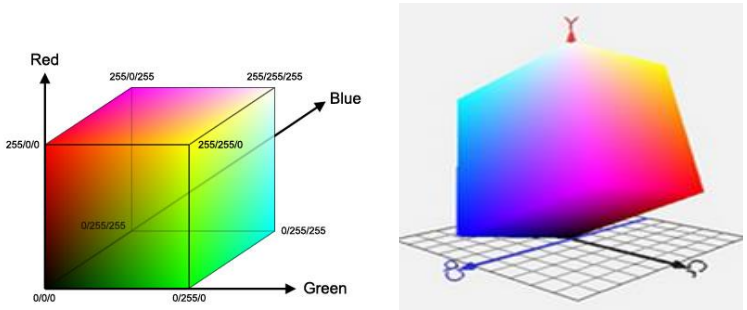
In the next sections, the color spaces and algorithms used for face detection and recognition, and the experimental results will be detailed.

2. Color Spaces

Several color spaces have been created to represent digital images, including RGB and YCbCr, which were used in this work.

In RGB space, pixels are represented by the intensity of their components Red, Green and Blue [5]. Thus, pixel values in RGB can be illustrated in Figure 1 (a), in which each dimension measures its component intensity. White and black are represented by the maximum and minimum values of the three components, respectively.

The YCbCr color space is divided into the luminance and chrominance components, and it is useful in image compression applications, and transmission of video signals. The luminance (Y component) measures the density of light intensity at a certain point in the image, and the chrominance (Cb and Cr components) measures the color values [5]. The YCbCr space is illustrated in Figure 1 (b), in which the x and y axes correspond to the chrominance, and the z axis corresponds to the luminance.



(a) (b)

Figure 1. (a) RGB Color Space [7]; (b) YCbCr Color Space [8]

The following equation converts RGB values into YCrCb [6]:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & 0.081 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (1)$$

3. Image Processing and Neural Networks in Matlab

Image files can be manipulated in Matlab through the *Image Processing Toolbox*, which includes the functions *imread* and *imwrite*, used to open and save image files, respectively.

Imread allows different syntaxes, and the most used in this work was *A = imread (filename, fmt)*, in which *A* receives the color matrix of the image *filename.fmt*. The color matrix size matches the image dimensions, and each position stores the color of a pixel. For binary images, the color matrix values are limited to 0 and 1, which represent black and white, respectively. For color images, the matrix in RGB is tridimensional, with integer values in the range 0-255 [9].

The function *imwrite* also allows different syntaxes, including *imwrite (A, filename.fmt)*, in which the color matrix A is stored in the image file *filename.fmt*.

The color matrices can be manipulated using basic matrix operations such as addition, subtraction and multiplication, or specific image processing functions, such as erosion and dilation [9].

Neural Networks can be created and managed in Matlab through the *Neural Network Toolbox*, which includes several training functions such as *feedforwardnet* and *cascadeforwardnet*, used to create Feed-forward and Cascade-forward neural networks, respectively.

The networks created in Matlab can be set on the number of hidden layers, training function, number of epochs, learning error, training rate and output format. The supervised network training can be performed through the function *net = train (net, P, T)*, where *net* is the neural network to be trained from the input matrix *P* and desired output in *T* [9].

4. Face Detection

Face detection is a pre-requisite for face recognition performing, since, at that stage, the faces will be delimited and the essential face features will be extracted.

In this study, face detection was implemented aimed at locating, aligning and delimiting faces in color images containing a single face in frontal position.

4.1. Face Localization

Techniques for face localization using skin segmentation in RGB and YCbCr have already been described in the literature [5, 10, 11].

In this work, face localization has been implemented by using a skin segmentation technique to find the image region containing the highest percentage of pixels classified as skin pixels. The pixels were classified according to their values in RGB space through a Feedforward neural network with two hidden layers, where the first layer has five neurons and the second one has three neurons.

The input layer of the neural network has three neurons, corresponding to the values for Red, Green and Blue of each input pixel. The output is a single value in the range from -1 to 1, where the outputs closest to -1 or 1 are most likely to be skin pixels. The Figure 2 depicts the neural network layers, which was returned from Matlab after training.

The training input was composed of 546.310 pixels of 40 images in RGB and the desired output was a binary vector with 546.310 elements, in which the value 1 was assigned to skin pixels and the value 0 to the rest of them.

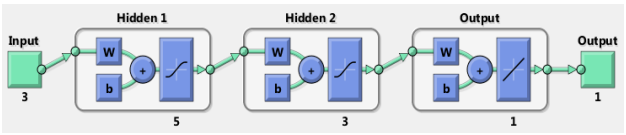


Figure 2. Diagram of the Neural Network used for Skin Segmentation

The images used to train the neural network contain people of different ethnic groups, chosen from the *Helen* dataset. In each image, skin pixels were painted blue and, from these painted images and the original ones, the input matrices and desired output vector have been created, in which pixels in blue received the output value 1. Some examples are shown in Figure 3.

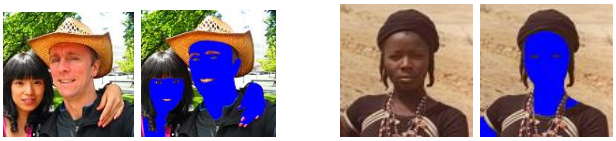


Figure 3. Manual Segmentation of Two Images used for Neural Network Training

For setting up the network, we used the Matlab function *feedforwardnet* and the training was performed in 100 epochs. The creation and training of the network were performed through the following code, in which *inputMatrix* contains the input set for training and *outputMatrix* contains the desired output:

```
function [] = setUpAndTrainNeuralNetwork()
inputMatrix = dlmread('inputMatrix.txt');
outputMatrix = dlmread('outputMatrix.txt');

net =feedforwardnet([5 3], 'trainbr');
net.divideParam.trainRatio = 1;
net.divideParam.valRatio = 0;
net.divideParam.testRatio = 0;
net.trainParam.epochs = 100;
net = train(net, inputMatrix', outputMatrix');
save('net.mat', 'net');
end
```

After training, the neural network was run on pixels of the test dataset *Faces 1999*. The result of running the network for each image was an array with values in the range from -1 to 1, in which each value is the output for one pixel. The resulting arrays were converted into positive matrices, in which the values closer to 1 represent the skin pixels, and, then, the matrices were scaled according to the original dimensions of each image. The neural network was run through the following function, which has a color image as input and returns a grayscale image corresponding to the network output:

```
function [grayscaleMatrix] = runNeuralNetwork(imageName)

load net;
image = imread(strcat(imageName, '.jpg));
rows = size(image, 1);
columns = size(image, 2);
matrix1 = reshape(image(:,:,1),[size(image,1)*size(image, 2), 1]);
matrix2 = reshape(image(:,:,2),[size(image,1)*size(image, 2), 1]);
matrix3 = reshape(image(:,:,3),[size(image,1)*size(image, 2), 1]);
inputMatrix = [matrix1 matrix2 matrix3];

result = sim (net , double(inputMatrix)');
grayscaleMatrix = abs(result);
grayscaleMatrix = reshape(grayscaleMatrix, rows, columns);
imwrite(grayscaleMatrix, strcat(imageName, 'greyScale.jpg'));
end
```


Some results are shown in Figure 4, wherein the darker areas represent the values closer to zero of the resulting matrices.

Figure 4. Some Inputs and their Outputs after Running the Neural Network for Skin Segmentation

To delimit the faces, the resulting images were analyzed in order to find the area (200x120 pixels) in the image with the highest percentage of 'skin', *i.e.*, with the highest density of bright pixels. The appropriate dimensions of the rectangle were estimated based on the average area of faces in the database. Some results are shown in Figure 5.

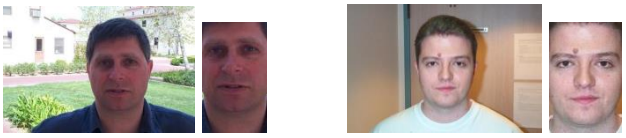


Figure 5. Results of the Face Delimitation

4.2. Eyes Detection

Techniques for eyes detection based on Hough Transform and skin detection have been described in [10, 12].

In this study, eyes detection has been implemented by combining two techniques. The first technique was based on the angles between neighboring pixels in RGB and the second one analyzes the chrominance in sets of neighboring pixels in YCbCr.

Given an image in which the eyes are open, it is expected that the rows intersecting that region present significant color changes of their pixels in at least eight points, which correspond to the corners of the eyes and iris. So, it can be inferred that the rows of this region present higher values for the angles between neighboring pixels, since the angle between two pixels in RBG space represents the chromaticity difference between them [13]. Given two pixels *p1 = (r1, g1, b1)* and *p2 = (r2, g2, b2)*, the angle θ between them can be calculated through the scalar product of the vectors *v1 = (r1, g1, b1)* and *v2 = (r2, g2, b2)*, as follows:

$$v_1 \cdot v_2 = \|v_1\| \cdot \|v_2\| \cdot \cos(\theta_{v1v2}) \quad (2)$$

$$\cos(\theta_{v1v2}) = \frac{v1 \cdot v2}{|v1| \cdot |v2|} \quad (3)$$

$$\theta_{v1v2} = \arccos\left(\frac{v1 \cdot v2}{|v1| \cdot |v2|}\right) \quad (4)$$

The sum of the angles between neighboring pixels in a given row L of the image I with n columns is obtained by the following equation:

$$\sum_{c=1}^{n-1} \arccos\left(\frac{I(l,c) \cdot I(l,c+1)}{|I(l,c)| \cdot |I(l,c+1)|}\right) \quad (5)$$

For example, the image in Figure 6 shows the sum of angles in five rows of an image containing a face. It can be noticed that the two rows in the eye region present the greatest sums.

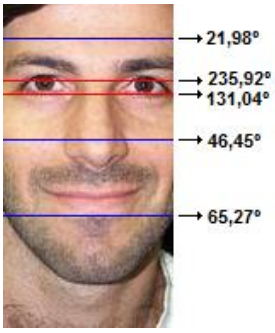


Figure 6. Sum of the Angles between Neighboring Pixels of Five Rows of an Image Containing a Face

Also, if we equate the luminance value for all pixels of an image in YCbCr containing a face, it can be seen that the color of the eye regions are closer to blue/green, whereas skin and lips regions are closer to orange/red. Thus, it is expected that the pixels of the eyes region have higher values for green and blue than skin pixels and the lowest values for red in RGB. Therefore, this region will present the lowest values for Cr and the highest values for Cb in YCbCr. Figure 7 illustrates this property, in which the images were converted from RGB to YCbCr and the value 0.7 was assigned to their luminance arrays.

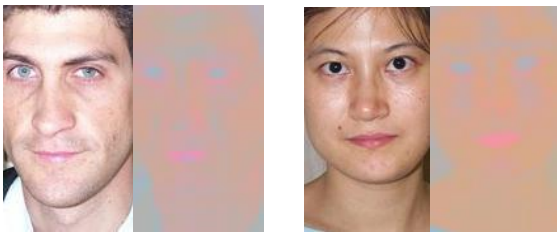


Figure 7. Faces Before and After Brightness-Leveling. The Color of the Eyes Regions is Closer to Green and Blue

Thus, to find the eye region using both techniques described above, the same algorithm was run for each vertical half of the input image, which searches the rectangular region (30x5 pixels) that presents the highest values for Cr and the greatest sum of angles between neighboring pixels. Assuming that the input images contain a properly delimited face, the search space was limited to an upper region of the face by the rows *startRow* and *finalRow*, defined as:

```
startRow = 1 + int32((numberRowsImage/2)*0.3);
finalRow = int32(numberRowsImage/2) -
int32((numberRowsImage/2*0.1);
```

The following code has been performed to search the left eye position in the input face, in which *leftEyeX* and *leftEyeY* store the initial row and column of the left eye region, *Cb* is the chrominance matrix and *anglesMatrix* stores the angles between the neighboring pixels of all rows in the input face:

```
leftEyeX = 1;
leftEyeY = 1;
greatestSumCb = sum(sum(Cb(1:1+eyeRows,1:1+eyeColumns)));
greatestAnglesSum = 0;
for i=startRow:finalRow-eyeRows
    for j=1:int32(eyeColumns/2)- eyeColumns
```



```
        cutFaceCb = Cb(i:i+eyeRows,j:j+eyeColumns);
        sumCutCb = sum(sum(cutFaceU));

        anglesSum = sum(sum(anglesMatrix(i:i+(eyeRows-2),
                                         j:j+(eyeColumns-2))));

      if ((sumCutU>greatestSumU && anglesSum >
0.9*greatestAnglesSum)
             || (anglesSum > greatestAnglesSum))
            leftEyeX = i;
            leftEyeY = j;
            greatestSumCb = sumCutCb;
            greatestAnglesSum = anglesSum;
        end
    end
end
```

This code was run on the resulting images from the face localization algorithm. Figure 8 shows some outputs, in which the red lines define the search space limits and the blue rectangles define the eye region edges.

Figure 8. Outputs of Eyes Detection Algorithm

4.3. Face Alignment

Face alignment was implemented based on the output of the eyes detection algorithm. This algorithm calculates the slope of the line passing by the center points of the eyes regions and then uses the Matlab function *imrotate(image, α)*, which rotates *image* in *α* degrees counter-clockwise around its center. The images were rotated clockwise or counter-clockwise depending on the angle sign. Since the most of the faces were already aligned, only the images where the angle *α* ranged from 5º to15º were changed by the algorithm in order to prevent errors. Figure 9 shows one output of this algorithm.

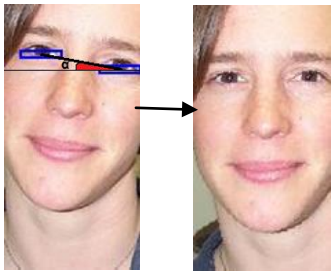


Figure 9. A face Before and After its Alignment, which was Rotated Counterclockwise in α=10.7389º

4.4. Lips Detection

Lips detection has been performed similarly to the eyes detection, except that, unlike the eyes regions, it is expected that the lips region presents the highest values for the Red color component, what can be seen in Figure 7. Thus, the area (20x50 pixels) in the image with the largest sum of Cr and Cb was defined as the lips region. Given:

$$Cb+Cr = (-0.14713*R- .28886*G + 0.436*B) + (0.615*R - 0.51498*G - 0.10001*B) \quad (6)$$
$$= 0.46787*R – 0.80386*G + 0.33599*B$$

We can conclude that Cb + Cr sum will be greater for larger values of R, since the first coefficient is positive.

The algorithm search space was limited to the lower region of the input image, because, if the input contains a properly delimited face, it is expected that the lips will be in the lower region.

The following code was run to perform the lips detection, in which *lipsX* and *lipsY* will store the initial row and column of the lips region after running the code, and Cb and Cr are the chrominance matrices:

```
lipsX = 1;
lipsY = 1;
greatestSum = 0;
for i=int32(1.2*imageRows/3):imageRows-lipsRows
    for j=1:imageColumns-lipsColumns
        cutFaceCb = Cb(i:i+lipsRows, j:j+lipsColumns);
        cutFaceCr = Cr(i:i+lipsRows, j:j+lipsColumns);
        sumCut = cutFaceCb+cutFaceCr;
        if (sum(sum(sumCut)) > greatestSum)
            lipsX = i; lipsY = j;
            greatestSum = sum(sum(sumCut));
        end
    end
end
```

Figure 10 shows some outputs of the lips detection algorithm, in which the rectangles in blue represent the lips region edges, and the red line is the first row of the search space.

Figure 10. Lips Detection Outputs

4.5. Faces Delimitation

The faces were delimited in two phases, after the eyes and lips detection, in order to eliminate the forehead, neck and chin regions. The objective was to crop out part of the faces keeping their principal components: eyes, nose and lips. Some results are shown in Figure 11.

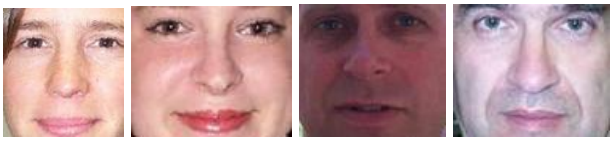


Figure 11. Delimited Faces

4.6. Resizing and Brightness-Leveling

This step aimed to resize and balance the illumination level of resulting images from the previous algorithm. First, it was necessary to establish a size standard for the faces: all images were resized to 100x100 pixels. To balance the images' luminance, some faces were chosen to define the illumination standard for the remaining faces. Eight images with an intermediate light level were selected. Then, the average matrix (M) of the eight luminance arrays was calculated and used as a parameter to change the luminance of all images in the dataset.

For each image, the resulting luminance component was defined as follows, where Y_i^{old} is the old luminance matrix of the image *i*, λ_i is the average value of Y_i, and λ_β is the average value of M.:

$$Y_i^{new} = Y_i^{old} * (\lambda_i / \lambda_\beta) \quad (7)$$

The result was the lighting enhancement for poorly illuminated images and lighting reduction for bright images, as shown in Figure 12.

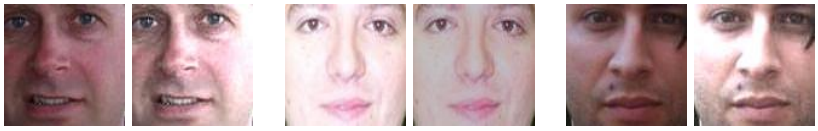


Figure 12. Faces Before and After the Brightness-Leveling

5. Faces Recognition/Classification

Faces detection and standardization on the previous steps have affected the results of the face recognition algorithm. It was expected that the higher the accuracy of face detection results, the greater the recognition rate.

Face recognition has been performed based on the color distribution pattern of each face, which is a global classification technique, because the face components are not separately analyzed. For each individual, we have chosen two images with their faces correctly detected, except for two individuals of which there was only one picture in the dataset. The average color matrices of the selected faces were the basis for the classification of the remaining faces and they will be referred here as *average face α*, where *α* is the individual identifier. None of the images chosen for the *average face*s were used in the test.

In the algorithm, each face was compared to the 28 individuals' *average face*. The algorithm uses the YCbCr space and calculates the standard deviation of the matrices Cb and Cr of the difference between the *average faces* and test faces, and then returns the individual identifier corresponding to the lowest standard deviation. Figure 13 illustrates some results for one test image of Individual 1, wherein the matrices Cb and Cr are represented as grayscale images.

It can be noticed in Figure 13 that gray shades variation in the matrices Cb and Cr is higher when the *average face* does not match the input face. It occurs due to particular properties of each face, as the eye, lips and nose dimensions, and skin tone. Thus, the smaller the variation in values of Cb and Cr, the smaller the standard deviations sum of Cb and Cr and the greater the probability of the input face belongs to that individual. The face in Figure 13 (a) was correctly recognized as Individual 1, and the standard deviations sum of the matrices in Figure 13 (e), Figure 13 (f) and Figure 13 (g) are equal to 0.0307, 0.0481 and 0.0419, respectively.

However, this technique works only if the face components have been properly located and aligned in all pictures, so that eye, nose and lips regions can be subtracted from their corresponding regions in the *average faces*.

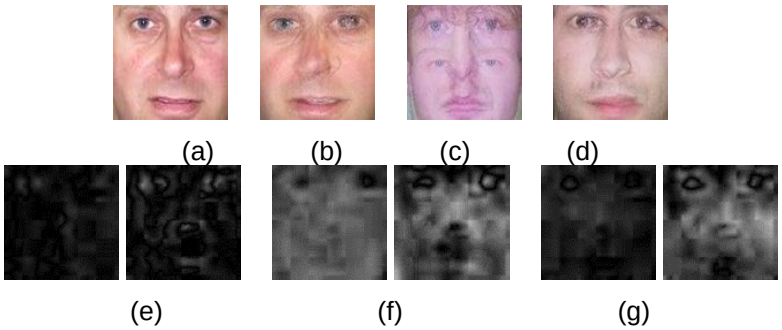


Figure 13. (a) Face of the Indivudual 1; (b) *Average face* 1; (c) *Average face* 16; (d) *Average face* 19; (e) Matrices Cb and Cr of the Difference between (a) and (b); (f) Matrices Cb and Cr of the Difference between (a) and (c); (g) Matrices Cb and Cr of the Difference between (a) and (d)

6. Experimental Results and Discussion

The correctness rate of face, eyes and lips detection, and face recognition/classification algorithms are presented in Table 1.

Table 1. Correctness Rate of Algorithms for Face Detection and Recognition

Algorithm	Correction rate
Face localization	96,9%
Eyes detection	89%
Lips detection	94%
Face recognition	70,7%

The correct results from the face localization algorithm include the faces that were correctly located without cutting out its main components (eyes, nose and lips). The results containing a large background area were considered incorrect. This algorithm failed when the pictures were poorly lit, containing a face with dimensions much smaller than 200x100 pixels, or when the background contained colors close to the human skin colors, as brown or beige. Some incorrect results are shown in Figure 14.

Figure 14. Incorrect Results of Face Localization Algorithm

The correct results of the eyes detection algorithm include images in which both eyes were correctly located. Among the faces in which the eyes were incorrectly located, poorly lit images and faces with fringes stood out, besides incorrect outputs of the previous algorithm. Some examples are shown in Figure 15.

Figure 15. Incorrect Results of Eyes Detection Algorithm

Regarding the lips detection algorithm, among 27 incorrect results, 17 errors occurred due to incorrect outputs of the face localization, and the other images were poorly lit. Some incorrect results are shown in Figure 16.

Figure 16. Incorrect Results of Lips Detection Algorithm

The correctness rate of the face recognition algorithm was 70.74% for all 393 test images and 77.81% for the 356 correctly detected faces.

Among the correctly detected faces, identification errors occurred mainly due to inaccurate face delimitations and asymmetric illumination in some images, wherein shadows were projected on a part of the face. However, the correctness rate was also affected by images chosen for the *average faces*, because the probability of correctly identifying a face increases according to the similarity of the input face with its correspondent *average face*.

The correctness rate of the face recognition algorithm could increase if a more accurate eyes detection algorithm were used, because, with the precise pupil location, the faces could be better laterally delimited. Furthermore, if the *average faces* were calculated using more images, it would be possible to cover a larger number of illumination's level and facial expressions, which would improve the algorithm result.

7. Conclusion

This paper presented techniques to detect and locate faces in color images. Faces have been detected through skin segmentation in RGB performed by a neural network. The regions of the eyes and lips were found by analyzing the chrominance and color variation in different regions of the face. Successively, face recognition was implemented through a global classification technique, which classifies the faces from the standard deviation of the difference between input faces and *average faces*.

By applying the techniques above, we have implemented in Matlab algorithms for face detection, which reached satisfactory results in comparison to the existing techniques, with correctness rates between 89.04 and 96.87%. The result of the face recognition algorithm was reasonable (70.74% correctness rate).

8. Future Works

In order to improve the results, the algorithms for face detection could be upgraded to detect multiple faces in the same image. Moreover, the neural network used for skin segmentation could be trained with more images, making it less prone to incorrect detection of face regions.

The face recognition algorithm could be better evaluated if it was run on very well delimited faces of a larger number of individuals.

Finally, it is necessary to implement a graphical user interface, so that the system could be used by non-techniqueal users.

References

[1] W. Zhao, R. Chellappa, P. J. Phillips and A. Rosenfeld, "Face recognition: A literature survey", Journal ACM Computing Surveys (CSUR), vol. 35, **(2003)**, pp. 399-458.
[2] R. Gottumukkal and V. Asari, "An improved face recognition technique based on modular PCA approach", Pattern Recognition Letters, Elsevier, vol. 25, **(2003)**, pp. 429–436.
[3] Helen dataset. (n. d.). Retrieved from http://www.ifp.illinois.edu/~vuongle2/helen/
[4] Faces 1999. (2005). Retrieved from http://www.vision.caltech.edu/html-files/archive.html
[5] S. K. Singh, D. S. Chauhan, M. Vatsa and R. Singh, "A robust skin color based face detection algorithm", Tamkang Journal of Science and Engineering, vol. 6, **(2003)**, pp. 227-234.
[6] M. H. Asmare, V. S. Asirvadam and L. Iznita, "Color Space Selection for Color Image Enhancement Application", Signal Acquisition and Processing, **(2009)**, pp. 208 – 212.
[7] Modelo RGB. **(2013)**. Retrieved from http://cindy2906.blogspot.com.br/2013/02/modelo-rgb.html
[8] The dimensions of color. **(2012)**. Retrieved from http://www.huevaluechroma.com/012.php
[9] MATLAB and Statistics Toolbox Release 2014 a, The MathWorks, Inc., Natick, Massachusetts, United States.
[10] D. Sidib, P. Montesinos and S. Janaqui, "A simple and efficient eye detection method in color images", International Conference Image and Vision Computing, New Zealand, **(2006)**.
[11] H. Lakshmi and S. Patilkulakarni, "Segmentation algorithm for multiple face detection in color images with skin tone regions using color spaces and edge detection techniques", International Journal of Computer Theory and Engineering. **(2010)**, pp. 162-166.
[12] Y. Ito, W. Ohyama, T. Wakabayashi and Fumitaka Kimura, "Detection of eyes by circular Hough transform and histogram of gradient", 21st International Conference on Pattern Recognition (ICPR 2012), **(2012)**, pp. 1795 – 1798.
[13] R. D. Dony and Wesolkowski, "Edge detection on color images using RGB vector angles", Proc. IEEE CCECE'99, Edmonton, Canada, vol. 2, **(1999)**, pp. 687-692.

Authors

Deise Santana Maia received her BS degree in Computer Science from State University of Southwest Bahia in 2014. She is currently a master student of Computer Science at Federal University of Minas Gerais.

Roque Mendes Prado Trindade received his degree in Mathematics from State University of Southwest Bahia (1994), MS degree in Computer Science from Federal University of Pernambuco (2002), and PhD degree in Electrical and Computer Engineering from Federal University of Rio Grande do Norte (2009). He is currently adjunct professor at State University of Southwest Bahia. His research interests include symbolic mathematics, interval mathematics, signal processing, intelligent systems, automation and control systems.

