CS485/685 Computer Vision

Spring 2011 – Dr. George Bebis

Programming Assignment 3

Due Date: 3/24/2011

In this assignment, you will implement an algorithm for normalizing face image using SVD. Face normalization is a required preprocessing step for many face recognition algorithms to account for variations with respect to face location, orientation (in-plane), and scale. To address these issues, you will implement an iterative algorithm based on affine transformations.

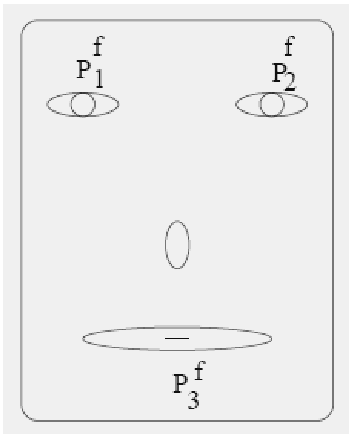

Figure 1. An example showing facial features of interest for normalization.

The main idea is using an affine transformation to map certain facial features to predetermined locations in a fixed window (see Figure 1). Figure 2 shows examples of un-normalized and normalized faces; normalization was performed by mapping the left eye center, right eye center, and mouth center to predetermined locations in a fixed 20 x 20 window.

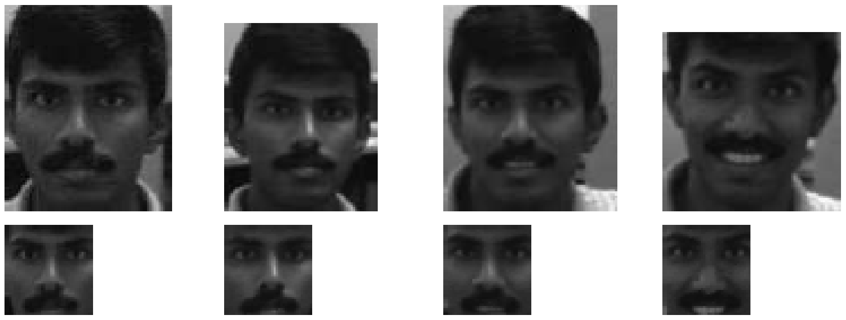

Figure 2. Examples of face images before (top) and after (bottom) normalization.

The face images to be used in your experiments are available from the course's webpage. You will use the following five facial features for normalization: **(1)** left eye center, **(2)** right eye center, **(3)** tip of nose, **(4)** mouth center, and **(5)** tip of chin. The coordinates of these features have been extracted manually and are available from the course's webpage. In practice, however, these features should be extracted automatically.

Fixed window

You will be using a fixed window of size 48 x 40. Note that the original images have size 112 x 92, so this choice maintains the aspect ratio of the face images. You can download the coordinates of the facial features within the fixed window from the course's webpage.

Algorithm

You need to implement the algorithm described below to compute the parameters of an affine transformation that maps the facial features of each image to the predetermined facial feature locations in the 48 x 40 window. This can be done in different ways. We describe below an iterative algorithm (see page 3 from reference [1]) which has shown to work well:

Step1: Use a vector $\overline{F}$ to store the average locations of each facial feature over all face images (i.e., (i.e., $\overline{F}$ **=(** $\overline{P}_1$, $\overline{P}_2$, $\overline{P}_3$, $\overline{P}_4$, $\overline{P}_5$) where $\overline{P}_1$ and $\overline{P}_2$ correspond to the average positions of the eye centers, $\overline{P}_3$ corresponds to the average position of the nose's tip, $\overline{P}_4$ corresponds to the average position of the mouth center, and $\overline{P}_5$ corresponds to the average position of the chin's tip); Initialize $\overline{F}$ with the feature locations in the first face image F_1 (i.e., $\overline{F}$ **=** F_1).

Step 2: Using SVD, compute the affine transformation that aligns $\overline{F}$ with the predetermined positions F^f=(P_1^f **,** P_2^f **,** P_3^f **,** P_4^f **,** P_5^f **)** in the 48 x 40 window.

An affine transformation can be used in this step. Since each pair of corresponding features must satisfy the affine transformation equations, we have the following equations:

$$P_1^f = A\overline{P}_1 + b$$
$$P_2^f = A\overline{P}_2 + b$$
$$P_3^f = A\overline{P}_3 + b$$
$$P_4^f = A\overline{P}_4 + b$$
$$P_5^f = A\overline{P}_5 + b$$

where

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \text{ and } b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Since there are 10 equations (i.e., two for each feature) in 6 unknowns, the system is over-determined and can be solved using SVD. By separating the x-coordinates from the y-coordinates, the above equations can be rewritten as follows:

$$Pc_1 = p_x$$
$$Pc_2 = p_y$$

where

$$P = \begin{bmatrix} \overline{X}_1 & \overline{Y}_1 & 1 \\ \overline{X}_2 & \overline{Y}_2 & 1 \\ \overline{X}_3 & \overline{Y}_3 & 1 \\ \overline{X}_4 & \overline{Y}_4 & 1 \\ \overline{X}_5 & \overline{Y}_5 & 1 \end{bmatrix} \quad p_x = \begin{bmatrix} X_1^f \\ X_2^f \\ X_3^f \\ X_4^f \\ X_5^f \end{bmatrix} \quad p_y = \begin{bmatrix} Y_1^f \\ Y_2^f \\ Y_3^f \\ Y_4^f \\ Y_5^f \end{bmatrix}$$

$$c_1 = \begin{bmatrix} a_{11} \\ a_{12} \\ b_1 \end{bmatrix} \quad c_2 = \begin{bmatrix} a_{21} \\ a_{22} \\ b_2 \end{bmatrix}$$

Once the transformation has been computed, apply it on $\overline{F}$ **;** let's call the aligned features $\overline{F}'$**.** Update $\overline{F}$ by setting $\overline{F} = \overline{F}'$.

Step 3: For every face image F_i, use SVD to compute the affine transformation that aligns the facial features of F_i with the average facial features $\overline{F}$ **;** call the aligned features F_i' **.**

Step 4: Update $\overline{F}$ by averaging the aligned feature locations F_i' for each face image i.

Step 5: If $||\overline{F}$ (t) - $\overline{F}$ (t-1)$||$ is less than a threshold, then stop; otherwise, go to Step 2.

The above algorithm typically converges within 5-7 iterations depending on the threshold used. For each face image, it yields an affine transformation that maps the face image to the 48 x 40 window. **You should verify that the computed affine transformation aligns the facial features with the fixed facial features.**

Computing the affine transformation using OpenCV's SVD function

In Step 2 (and 3), you will need to use SVD to find c_1 and c_2. OpenCV's function **cvSVD()** performs SVD. Also, OpenCV's function **cvSVBkSb()** uses the results of SVD to solve a system of equations Ax=b where A is over-determined. Instead of calling these functions separately, you can use OpenCV's function **cvSolve()** which calls both of these functions to solve systems of linear equations (see pages 73-76 from the OpenCV book). I would recommend that you first try **cvSolve()** on a problem (overdetermined system) whose solution you know to make sure that you get the correct results.

Computing the normalized images

To avoid gaps when computing the normalized images, each point in the normalized images should be determined by applying the **inverse** affine transformation equations as shown in Figure 3. The inverse affine equations are given below. So, you should iterate through every point in the output image, find the corresponding point in the input image, and copy it over in the output image. **Make sure that you understand how to implement correctly the equation shown below** (i.e., multiply [X' Y' 1] with every column of the inverse matrix and divide the results by the quantity shown).

$$[X, Y, 1] = [X', Y', 1]\frac{1}{a_{11}a_{22} - a_{21}a_{12}}\begin{bmatrix} a_{22} & -a_{21} & 0 \\ -a_{12} & a_{11} & 0 \\ a_{21}b_2 - a_{22}b_1 & a_{12}b_1 - a_{11}b_2 & a_{11}a_{22} - a_{21}a_{12} \end{bmatrix}$$

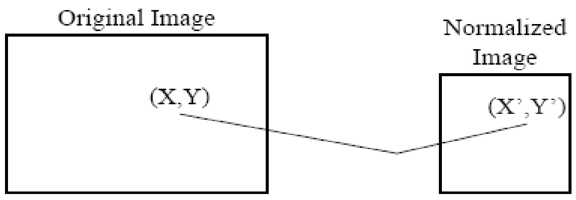

Figure 3. Inverse affine mapping.

Graduate Students Only

Once a face image has been normalized with respect to position, scale, and orientation, some light correction can be applied to account for non-uniform illumination (see pages 9-10, from reference [2]). First, a linear (or higher order) model is fit to the intensity of the image; for example, the following model can be used:

$$f(x,y)=ax+by+cxy+d$$

Each pixel (x,y) in the input image must satisfy the above equation where f(x,y) is the intensity value of the input image at location (x,y). Using SVD, we can find the "best" coefficients a, b, c, and d (i.e., in a "least-squares" sense). For an N x M image, this yields NM equations with four unknowns (i.e., over-determined system).

Figure 4 shows an example; the first row shows some input images while the second row shows the linear model fit to each of these images. To correct for lighting, simply **subtract** the linear model from the original image. The last row of Figure 4 shows some results. Your task in this part of the assignment will be to apply light normalization on the normalized face images.

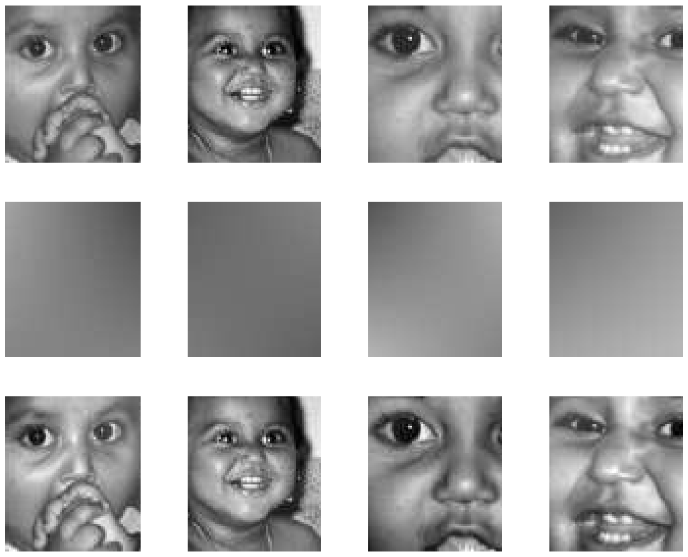

Figure 4. Original images (1st row), model fit (2nd row) light-corrected images (3rd row).

What to turn in

You are to turn in a report including a print-out of your source code. Your report should include the following: a description of the experiments, results (i.e., include graphic output of your results), discussion of results and comparisons, and a brief summary of what you have learned. **The report is very important in determining your grade for the programming assignment.** Be well organized, type your reports, and include figure captions with a brief description for all the figures included in your report. Motivation and initiative are greatly encouraged and will earn extra points.

References

[1] H. Rowley, S. Baluja, and T. Kanade, “Neural Network-Based Face Detection”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 1, January, 1998, pp. 23-38.

[2] G. Bebis, S. Uthiram, and M. Georgiopoulos, "Face Detection and Verification Using Genetic Search", *International Journal on Artificial Intelligence Tools*, vol 9, no 2, pp. 225-246, 2000.