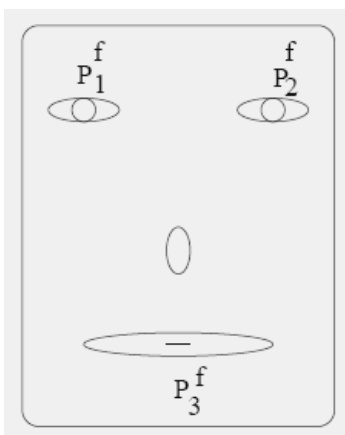


**CS485/685 Computer Vision**  
**Spring 2009 – Dr. George Bebis**  
**Programming Assignment 1**  
**Due Date: 2/26/2009**

The goal of this assignment is to help you familiarize yourselves with Singular Value Decomposition (SVD) and OpenCV. In this context, you will implement an algorithm for face image normalization. Specifically, most face recognition algorithms require that once a human face has been detected in an image, normalization is applied prior to recognition to account for scale, orientation, and location differences. To address this issue, you will implement an iterative scheme based on affine transformations.



**Figure 1.** An example face showing the features of interest.

The main idea is using an affine transformation to map certain facial features from a face image to predetermined locations in a fixed window (e.g., 40 x 20) such as the one shown in Figure 1. Figure 2 shows some normalized face examples; 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 (i.e., using left eye center, right eye center, and mouth center)

The images to be used in your experiments are available from the course's webpage. You will need to use the following five facial features in your experiments: (1) left eye center, (2) right eye center, (3) tip of nose, (4) mouth center, and (5) tip of chin. In principal, the location of these features should be extracted automatically; here, you will extract them manually using *IrfanView* or any other image viewer of your preference. To find the image coordinates of a given facial feature, move the mouse cursor on top of that feature and left-click; the coordinates of the cursor's position will appear on the upper-left corner of *IrfanView*'s window (*warning*: the column number appears first and the row number appears second). Simply record these coordinates in a file a repeat the same process for every facial feature in all the images.

### Fixed window size

Define the predetermined locations of the five facial features in a fixed size window, for example, 48 x 40. Note that the original images have size 112 x 92, so this choice maintains the aspect ratio of the face images.

### Algorithm

Once you have extracted the locations of the facial features from all images, you will need to compute the parameters of an affine transformation that maps them to the predetermined locations in the 48 x 40 window. This can be done in different ways. We describe below an iterative algorithm which has shown to work well in practice:

**Step1:** Use a vector  $\bar{F}$  to store the average locations of each facial feature over all face images; initialize  $\bar{F}$  with the feature locations in the first face image  $F_1$ .

**Step 2:** Compute the best transformation that aligns  $\bar{F}$  (i.e., average positions eye centers ( $\bar{P}_1$  and  $\bar{P}_2$ ), average position of nose's tip ( $\bar{P}_3$ ), average position of mouth center ( $\bar{P}_4$ ) and average position of chin's tip ( $\bar{P}_5$ )) with the predetermined positions ( $P_1^f$ ,  $P_2^f$ ,  $P_3^f$ ,  $P_4^f$ ,  $P_5^f$ ) 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:

$$\begin{aligned} P_1^f &= A \bar{P}_1 + b \\ P_2^f &= A \bar{P}_2 + b \\ P_3^f &= A \bar{P}_3 + b \\ P_4^f &= A \bar{P}_4 + b \\ P_5^f &= A \bar{P}_5 + b \end{aligned}$$

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 we have 10 equations (two for each feature) and 6 unknowns, we have an over-determined system which 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

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

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

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

**Step 3:** For every face image  $F_i$  compute the best transformation that aligns the facial features of  $F_i$  with the average facial features  $\bar{F}$ ; call the aligned features  $F_i'$ . The best alignment transformation can be computed using an affine transformation again as before. You will need to solve an over-determined system, so SVD will be useful again.

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

**Step 5:** If the error between  $\bar{F}(t)$  and  $\bar{F}(t-1)$  is less than a threshold, then stop; otherwise, go to Step 2.

The above algorithm typically converges within seven iterations. For each face image, it yields an affine transformation that maps the face image to the 48 x 40 window.

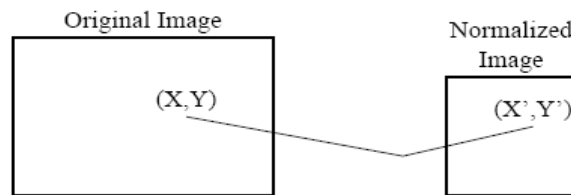
## Computing the affine transformation

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. For more information, see pages 73-76 from "**Learning OpenCV**" by Bradski and Kaehler.

## Applying the affine transformation

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.

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



**Figure 3.** Illustration of the inverse mapping.

## Graduate Students Only

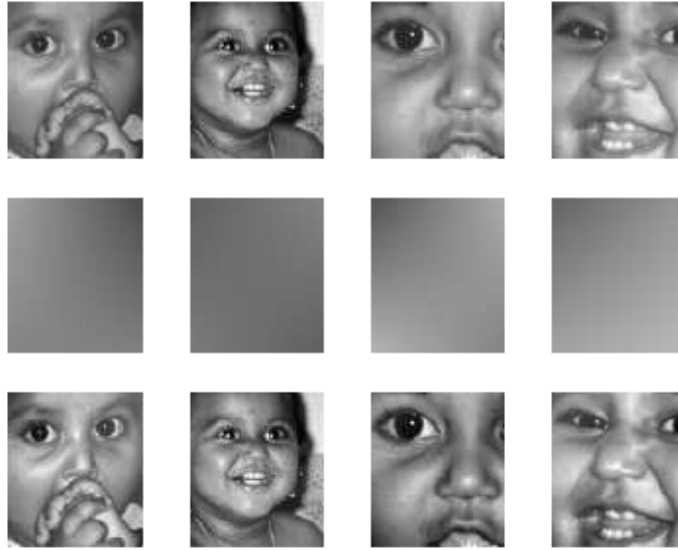
Once a face image has been normalized with respect to position, scale, and orientation, some light correction can be applied to account for non-uniform illumination. First, a linear (or higher order) model is fit to the intensity of the image; for example, the following model can be used:

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

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  $N \times M$  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 face images



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

### **What to turn in**

For each programming assignment, you are to turn in a brief report including a print-out of your source code. Your report should include the following: **(1)** methodology (i.e., explanation of the methods used), **(2)** description of the experiments, **(3)** results (i.e., include graphic output of your results), **(4)** explanation of results (i.e., justify your results), and **(5)** a brief summary of what you have learned. Organize your report nicely, label all sections and figures. Neat reports and extra work will win extra credit.