CS 4/791E Computer Vision

Spring 2004 - Dr. George Bebis

Programming Assignment 1

Due date: 2/26/04

This assignment has two parts. The goal is to familiarize yourselves with some fundamental image processing algorithms and the OpenCV library.

1. **(25 pts)** Implement image smoothing using convolution with Gaussian masks. First, implement 2D Gaussian convolution using 1D Gaussian masks as discussed in class. I have put on the web the code for generating the Gaussian mask. For comparison purposes, implement 2D Gaussian convolution using 2D Gaussian masks. For this part, use OpenCV's *cvSmooth* function and the option *CV_GAUSSIAN*. In both cases, show your results using mask sizes 3x3, 5x5, and 7x7.

2. **(75 pts)** The majority of face recognition algorithms require that human faces in an image have been detected an normalized prior to recognition. By normalization we imply normalize faces with respect to location, size, orientation, and lighting. Here, you would need to implement a simple algorithm based on affine transformations.

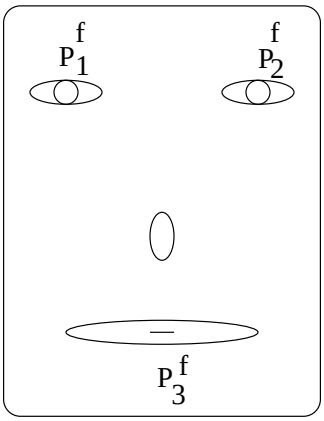

Figure 1. A typical face image showing the features of interest.

Specifically, the algorithm uses an affine transformation to map certain facial features (e.g., eyes) to predetermined locations in a fixed window (e.g., 20x20). Figure 1 shows an example. Light correction and histogram equalization can be applied on the result to normalize faces with respect to illumination but you would not have to worry about it in this assignment. Figure 2

shows examples of faces normalized with respect to location, size, and orientation while Figure 3 shows the results of illumination normalization.

Figure 2. Examples showing face images before and after normalization.

Figure 3. Original images (top), light-corrected images (3rd row), and histogram equalized images (last row).

A set of images to be used in your experiments are available from the course's webpage. In this assignment, you will be using the following facial features: left eye center, right eye center, tip and mouth center. First, you would need to extract manually the coordinates of these facial features from each face image provided. Use *xv* for this step or any other image viewer. For example, when placing the cursor on the image locations of interest and clicking the middle mouse button, *xv* displays the image coordinates of the cursor on the screen (*warning*: the column number is shown first and the row number second). Then, you would need to choose the predetermined locations of these features in a fixed size window, let's say, 40x48 (note that the original images have size 112/92, so the aspect ratio is maintained) and compute the parameters of the affine transformation.

Every feature point needs to be mapped to its corresponding location in the fixed window, thus, it needs to satisfy the affine transformation equations. For example, let's say that the average eye and mouth locations are denoted as (P_1, P_2, P_3), and the predetermined locations as (P_1^f, P_2^f,

P_3^f), then the affine transformation equations are given below:

$$P_1^f = AP_1 + b$$
$$P_2^f = AP_2 + b$$
$$P_3^f = AP_3 + b$$

where

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

The above equations can be rewritten as

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

where

$$P = \begin{bmatrix} X_1 & Y_1 & 1 \\ X_2 & Y_2 & 1 \\ X_3 & Y_3 & 1 \end{bmatrix}$$

$$p_x = \begin{bmatrix} X_1^f \\ X_2^f \\ X_3^f \end{bmatrix} \quad p_y = \begin{bmatrix} Y_1^f \\ Y_2^f \\ Y_3^f \end{bmatrix}$$

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

Since each facial feature from each image contributes a pair of equations (i.e., one of the x-coordinates and one for the y-coordinates), by putting together all the equations you get an overdetermined set of linear equations that can be solved using Singular Value Decomposition (SVD). I have put on the web the code for solving overdetermined systems of equations. We will talk about SVD in detail later in this course. Once you have solved for the parameters of the affine transformation, you would need to normalize each face using the computed affine transformation.

You would need to turn in a report, including the results of your experiments, and a discussion. Show the inverse equations for the affine transformation too. Email me your code with instructions of how to compile it and run it.