CS 485/685 – Computer Vision

Programming Assignment 1

(Due February 25)

1. (40 pts) Implement image smoothing using convolution with Gaussian masks. You should use 2 input images of your choice. First, implement 2D Gaussian convolution using 1D Gaussian masks as discussed in class. Code for generating the Gaussian mask is available on the class webpage. For comparison purposes, also 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.

   Submit the following:
   - (hardcopy) a report describing your results; it must include the masks used, the original and the smoothed images, a discussion of results and comparisons; for all figures provide captions with a brief description
   - (email) one ZIP file containing:
     - the source code files
     - a README file with instructions on how to compile and run the program

2. (60 pts) Most face recognition algorithms require that human faces in an image are normalized prior to recognition. In general, normalization is performed with respect to location, size, orientation, and lighting. Here, you would need to implement a simple algorithm based on affine transformations.

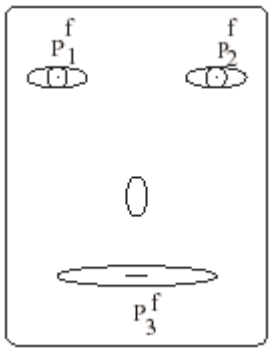

Figure 1. A typical face image showing three features of interest.

Specifically, the algorithm uses an affine transformation to map certain facial features to predetermined locations in a fixed window (e.g., 20x20). Figure 1 shows an example of such

facial features. Figure 2 shows examples of faces normalized with respect to location, size, and orientation. You do not have to worry about lighting normalization in this assignment.

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

A set of images to be used in your experiments is available from the class webpage. In this assignment, you will be using the following 4 facial features: left eye center, right eye center, nose tip and mouth center. First, you will need to extract manually the actual coordinates of these facial features from each face image provided. Use any image viewer that shows the image coordinates for the location currently under the mouse cursor. Then, you have to choose the predetermined locations of these features in a fixed size window, say 40x48 (note that the original images have size 112x92, so the aspect ratio is maintained) and compute the parameters of the affine transformation.

Every feature point needs to be mapped to its predetermined location in the fixed window, thus it needs to satisfy the affine transformation equations. The example below considers only 3 features, although you will use 4. Denote the actual eye and mouth locations with (P_1, P_2, P_3) and their predetermined locations with (P_1^f, P_2^f, P_3^f). 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 (one for the *x* coordinates and one for the *y* coordinates), by putting together all the equations for that image we obtain an over-determined set of linear equations that can be solved using Singular Value Decomposition (SVD). The code for solving over-determined systems of equations is available on the class webpage. We will cover SVD in detail later in the course.
Once you have solved for the parameters of the affine transformation for each image, you have to normalize each image using its computed affine transformation.

Submit the following:

- (hardcopy) a report describing your results; it must include the recovered parameters for the affine transformations and the normalized images, a discussion of results and comparisons; for all figures provide captions with a brief description
- (email) one ZIP file containing:
    - the source code files
    - a README file with instructions on how to compile and run the program