CS 480/680 Computer Graphics

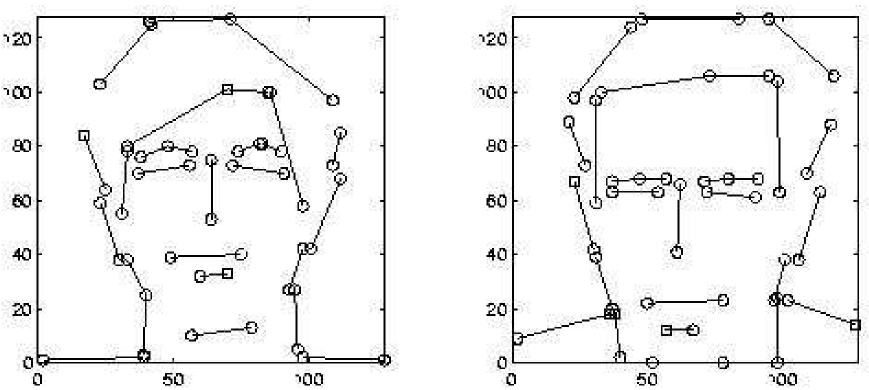
Spring 1999 - Dr. George Bebis

Programming Assignment 4

(for students who take the class at the 700 level)

Due date: 5/11/99

For this project, you will work on image morphing. Morphing is a technique used for the metamorphosis from one image to another. The idea is to get a sequence of intermediate images which, when put together with the original images, would represent the change from one image to the other. The simplest method of transforming one image into another is to *cross-dissolve* between them. According to this method, the color of each pixel is interpolated over time from the first image value to the corresponding second image value. This is not so effective in suggesting the actual metamorphosis. For morphs between faces, the metamorphosis does not look good if the two faces do not have the same shape approximately. In this project, you will implement a morphing scheme which would combine cross-dissolve with warping methods to give good morphs. This is based on the paper *Feature-Based Image Metamorphosis* by Thaddeus Beier and Shawn Neely. The morph process consists of a warping stage before cross-dissolving so that the two images have the same shape. The warp is specified, in this case, by a mapping between lines in the first and second images. The following figure shows an example of corresponding lines between two face images.



Feature-Based Image Metamorphosis

You need to read the paper provided to you but here are the basic ideas:

Transformation with One Pair of Lines

In this case, one line in the source image is mapped to a corresponding line in the destination image. The other parts of the image are moved appropriately to maintain their relative position from the specified line. Each of these lines is directed. This could be used to rotate and scale images.

Transformation with Multiple Pairs of Lines

Transformations performed using more than one pair of lines involve a weighted combination of the transformations performed by each line. Since reverse mapping is used, for each pixel in the destination image we find a pixel in the source image which would be used. The displacement corresponding to each line is calculated for each pixel. A weigted average of these displacements is used to determine how much each pixel needs to be moved. The weight used depends on the distance of the point under consideration from the line. The weight can also depend on the length of the line. This transformation, based on multiple lines, can be used effectively for morphing.

Pictures

Use your own pictures for morphing. You need to go to 315 LME and ask somebody to help you take a picture of your face (you can also ask me to help you with this). If you want to edit your original picture before morphing, simple operations (e.g., clipping a certain part of the picture, changing colors, resizing) may be done by using *xv*.

Defining features

To find point coordinates in a picture, you can open the picture using *xv* and move the cursor on the picture and click the mouse at a desired point. The x-y coordinates will be displayed somewhere in the window. Feature points or edges in the two pictures (source and destination) can be saved manually and used later in your morphing program.

Morphing

The following routine should be used to generate a morph sequence between pictures A and B. Suppose that two corresponding features have already been selected and recorded in seperate files F_a (for picture A) and F_b (for picture B). These files define a collection of line features.

Step 1. First, picture A will be morphed based on corresponding features in F_a and F_b. Let $A_{morphed}$ be the final destination image created by using picture A and features recorded in F_a and F_b. We want to create a number of intermediate pictures between A and $A_{morphed}$. Beier's paper suggests two methods for defining edge features for intermediate images: (i) endpoints are linearly interpolated between F_a and F_b (simple method), and (ii) centerpoint and angle of a line feature are interpolated seperately (a bit better method). Either way, intermediate features can be generated and by using these features, we can generate a morph sequence $A_0 = A$, A_1 , A_2 , ..., $A_n = A_{morphed}$. The number n determines the length of animation (10 at least).

Step 2. Do the same thing for picture B. We can create a morph sequence $B_0 = B_{morphed}$, B_1, B_2, ..., $B_n = B$. Note that the suffix ordering of this sequence is different from that of A.

Step 3. Now, by crossresolving A_i and B_i, we can generate C_i as follows. Let $t = \frac{i}{n}$. Let P_a , P_b, and P_c be corresponding pixels in A, B, and C, respectively. Then, $P_c = tP_b + (1 - t)P_a$.

Making mpeg

Each picture in the morph sequence can be viewed independently by using *xv* or can be viewed as an animated sequence by using *xanim*. To view it as an animation, you need to convert the sequence of images into an Mpeg file by using *makempeg* (see http://pinon.cs.unr.edu/~bebis/CS480/make_mpegs.html).

Submission

You need to submit an Mpeg file together with your program. You can store your original pictures in your own directory and make them accessible by me. This way, we can save some disk space. In case you run out of disk space to store intermediate pictures, let me know.