MP7: Video Animation

Group: Akhil Alapaty & Ojus Deshmukh

I. Introduction.

In the case of this MP, we are directly given the audio and visual features. The audio features were found using the Cepstrum of each of the raw audio vectors. This is done by converting the raw pixel feature vector signal into a vector of frames using the sig2frames function. Next, you take the Inverse Fourier Transform of the logarithm of the frames vector's magnitude spectrum. The Cepstrum feature vector is more effective than the Raw Pixel feature vector because it catches features such as pitches and frequencies in a speech signal, since they are part of the logarithmic representation. The cepstrum is generally characterized as containing information that depicts the rate of change in a signal's separate spectrum bands.

As for the visual data, it is vectors containing data referring to the lip width, the distance between the upper/lower lips, and the line connecting the mouth corners. Clearly, these visual features are more effective than the Raw Pixel feature vector since each of these measurements can now be used along with the mesh to animate the mouth image. This allowed us to create a mesh of triangles that made up the face, which we could then warp using Affine transformations to animate the mouth.

The learning algorithm used for this MP is the Artificial Neural Network. ANNs function by modeling the biological neural network through building blocks known as perceptrons. As defined in the powerpoint on the website, perceptrons output either +1 or -1 after a linear combination of a vector of real-valued inputs. These perceptrons can be used to implement common Boolean expressions, such as AND, OR, NOT etc. In this case, we train a three-layered ANN, each layer for each visual dimension: w, h1, and h2. These dimensions respectively refer to the width of the mouth, the vertical distance of the top lip, and the vertical distance of the bottom lip. With the trained neural networks and the test audio features, the neural network then predicts the visual dimensions w, h1, and h2 for each frame of the face.

II. Methods

Our affine transformations were done entirely in imagewarp.m. It first get the deformed mesh matrix, then find out which new deformed mesh triangle each pixel of the original image belongs to. We can calculate the warped coordinate (x,y) from the original coordinate(η, ζ), using the equation:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_0 \\ b_1 & b_2 & b_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \eta \\ \zeta \\ 1 \end{bmatrix}*$$

This calculation is done entirely in interpVert.m. We also used Barycentric coordinates while creating the animation frames, as homogeneous Barycentric coordinates remain the same regardless of affine transformations.

$$\begin{bmatrix} \zeta \\ \eta \\ 1 \end{bmatrix} = \begin{bmatrix} \zeta_1 & \zeta_2 & \zeta_3 \\ \eta_1 & \eta_2 & \eta_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}$$

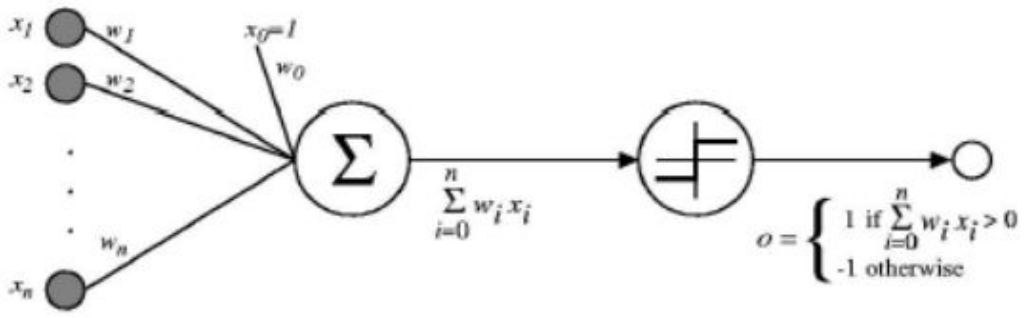
```
function out = imagewarp(vert,tri_pts,lip_pts,input)

length = size(lip_pts,2);
[h, w] = size(input);
out = zeros(h, w, length);
xdim = vert(:,1);
ydim = vert(:,2);

for fr = 1:length % iterate through each of the image's frames
    [xwarp, ywarp] = interpVert(xdim, ydim, 0, 0, 0, lip_pts(1,fr), lip_pts(2,fr), lip_pts(3,fr), 1);
    for tri = 1:42 % per mesh triangle
        vert1init = [xdim(tri_pts(tri,1)) ydim(tri_pts(tri,1))];
        vert2init = [xdim(tri_pts(tri,2)) ydim(tri_pts(tri,2))];
        vert3init = [xdim(tri_pts(tri,3)) ydim(tri_pts(tri,3))];
        vert1warp = [xwarp(tri_pts(tri,1)) ywarp(tri_pts(tri,1))];
        vert2warp = [xwarp(tri_pts(tri,2)) ywarp(tri_pts(tri,2))];
        vert3warp = [xwarp(tri_pts(tri,3)) ywarp(tri_pts(tri,3))];
        imgwarp = [vert1warp, 1; vert2warp, 1; vert3warp, 1]';
        for i = 1:h
            for hor = 1:w
                l = inv(imgwarp) * [hor i 1]';
                if max(l) <= 1 && min(l) >= 0
                    imginit = [vert1init, 1; vert2init, 1; vert3init, 1]' * l;
                    out(i, hor, fr) = (imginit(2) - floor(imginit(2))) * ((imginit(1) - floor(imginit(1))) *
                    input(ceil(imginit(2)),ceil(imginit(1))) + (1 - (imginit(1) - floor(imginit(1)))) *
                    input(ceil(imginit(2)),floor(imginit(1)))) +
                    (1 - (imginit(2) - floor(imginit(2)))) * ((imginit(1) - floor(imginit(1))) *
                    input(floor(imginit(2)),ceil(imginit(1))) + (1 - (imginit(1) - floor(imginit(1)))) *
                    input(floor(imginit(2)),floor(imginit(1))));
                end
            end
        end
    end
```

Pictured above is our imagewarp.m file, in which we take our visual dimensions as inputs and output each frame as desired.

Our second critical transformation was training the ANN. This was done entirely in ECE417_MP5_train.m. There were three neural networks, one for each visual dimension w, h1, and h2. The formula we have below depicts the perceptron function, as its output relative to a threshold value determines how the neural network will work.



$$o(x_1, \ldots, x_n) = \begin{cases} 1 \text{ if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 \text{ otherwise.} \end{cases}$$

**

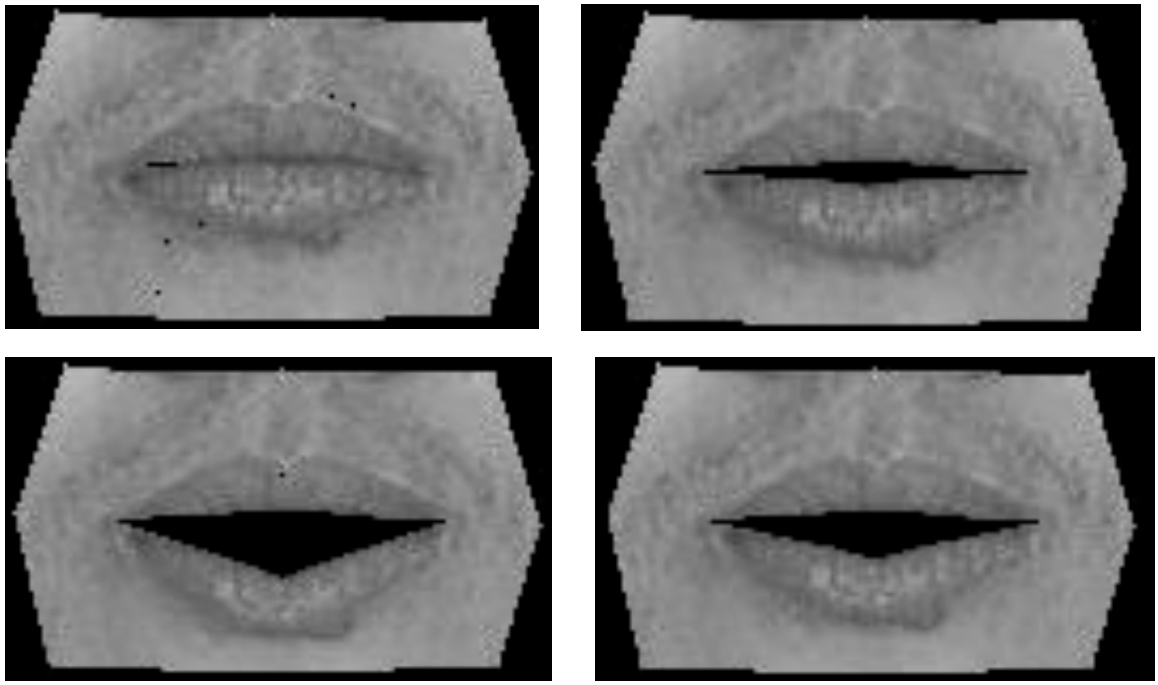
Gradient Descent was then used on each network to find new parameters by updating the weights of inputs and biases of individual neurons. The weight vector w was updated to the direction of steepest descent along the error surface.

$$\nabla E(w) \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \cdots, \frac{\partial E}{\partial w_n} \right]$$

$$w \leftarrow w + (-\eta \nabla E(w))$$

III. Results

Our final video animation can be seen in our test.avi file provided with our code. However, we have included screenshots of the facial animation when saying the digit ‘5’, as an example.



(Progression of pronunciation is clockwise beginning from top-left image)

IV. Discussion

As seen in the images above, the mouth opens a lot more when uttering the digit 5, when compared to the movements during other digits. We noticed a similar pattern with the digit 9. After doing some thinking, we realized that the “aye” sound which comes from the ‘i’ present in both 5 and 9 requires the mouth to open most, while the other numbers don’t have that sound.

*Diagrams taken from Vuong Le, “Image Warping”, University of Illinois Spring 2013

**Diagrams taken from Vuong Le/Hao Tang’s slides, “Introduction to Artificual Neural Network”