ECE 417 MP5 Walkthrough - Real-Time Speech-Driven Facial Animation

Dan Soberal, Po-Sen Huang (ECE 417 TA's)

Spring 2014

Goal

1. Up to this point, we've worked on recognition and classification. In this MP, we'll switch gears and focus on synethesis (generation).
2. Specifically, we want to generate a "talking head" model.
3. The goal of this machine problem is to use speech data to try to deform a mouth image such that we can give the impression that it is actually speaking the audio data we give it.

The Basic Idea

Given a set of audio and image features, train a mapping between the two. Then, use said mapping to generate a set of test images given a set of audio features.

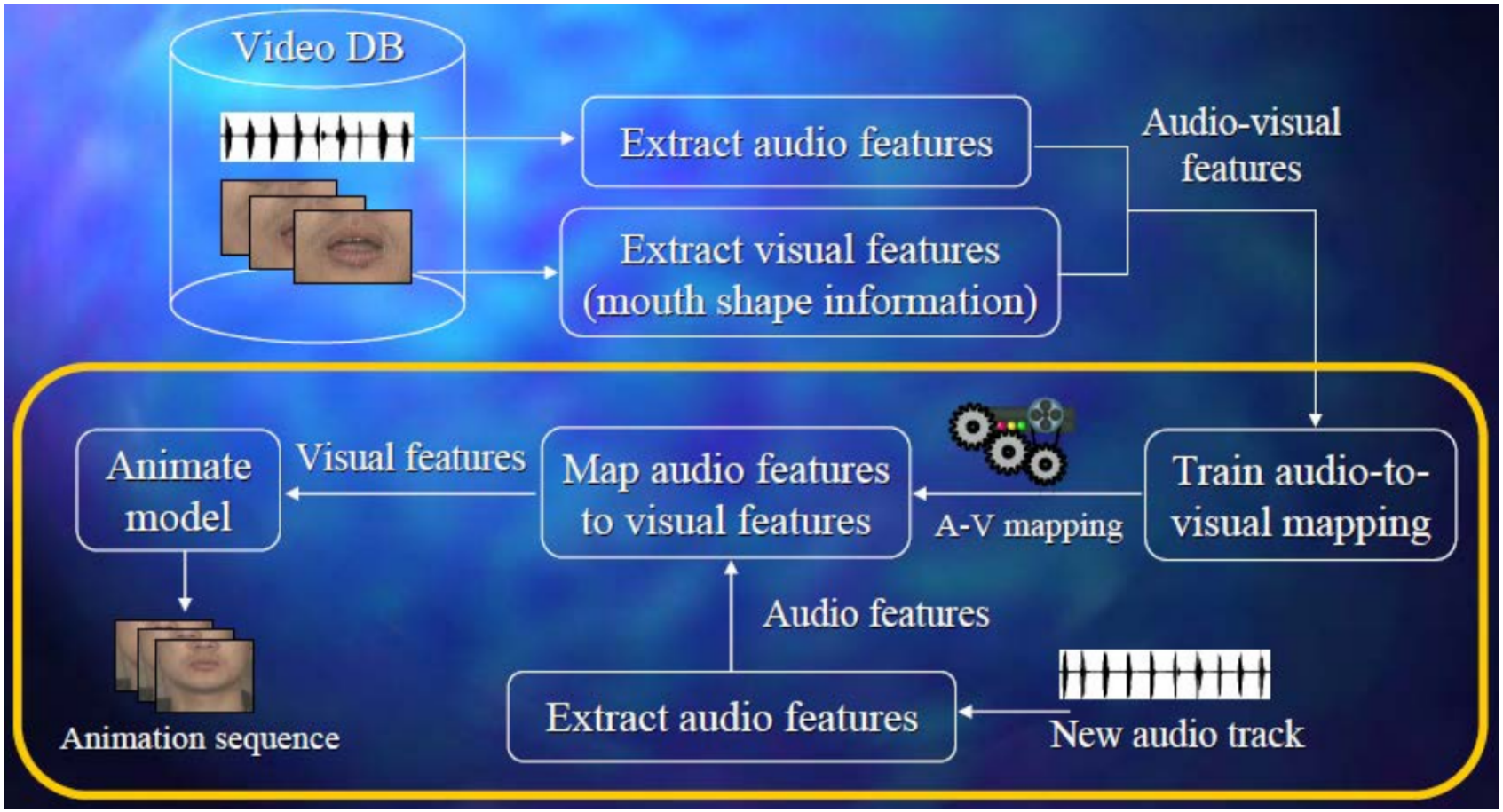


Figure : General Block Diagram *(DB Stands for database)

The Basic Idea (Specific to MP5)

We can generalize this so that it is specific to the data we will use in MP5. We will use image deformation to warp a base model of a pair of lips based on an audio-image mapping system.

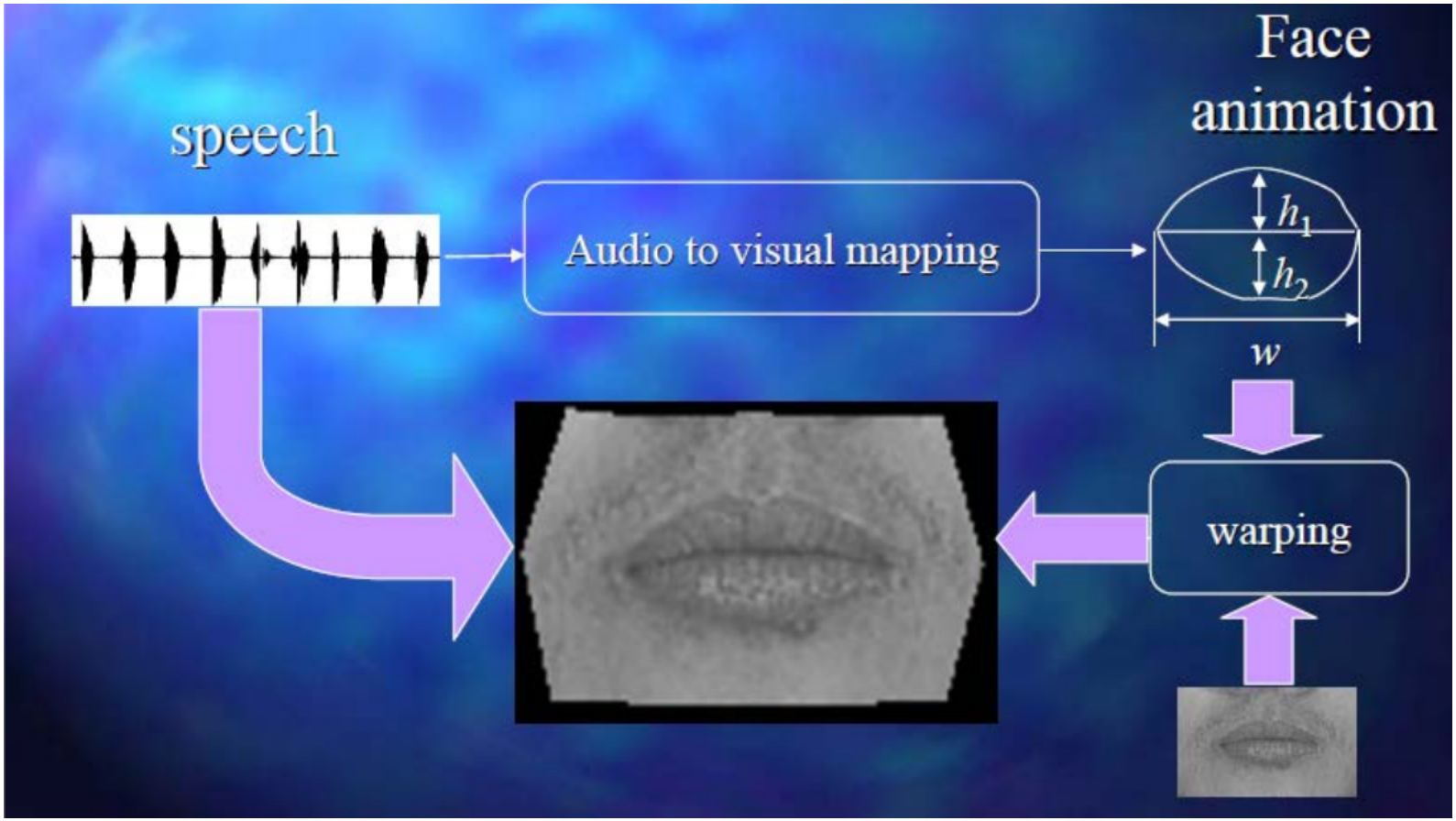


Figure : MP5 Specific Block Diagram

Things You Are Given

1. Code You Are Given
   - 1.1 ECE417_MP5_clean_silence.m - finds unvoiced sections in the speech and eliminates them
   - 1.2 ECE417_MP5_smooth.m - frame-by-frame smoothing after silence deletion
   - 1.3 ECE417_MP5_test.m - estimates visual features for test audio data
   - 1.4 ECE417_MP5_train.m - Trains a 3 layer nueral network for audio-image map (requires Neural Network Toolbox, which EWS computers should have)
   - 1.5 interpVert.m - generate output vertices (x, y) based on (w, h_1, h_2).
   - 1.6 DxBMP.exe - executable file that generates movie based on indivdual image frames
2. Data You Are Given
   - 2.1 mouth.jpg - a base (neutral) image of the mouth that you will deform.
   - 2.2 mesh.txt - L vertices with coordinates (x, y), M mesh triangles with vertices (v_1, v_2, v_3).
   - 2.3 ECE417_MP5_AV_Data.mat - contains the audio features (N cepstral coefficients) for T frames.
   - 2.4 test.wav - audio file used to generate the audio features. Use it to overlay audio in your final movie.

High-Level Approach

1. Use 3 neural networks to create 3 image coordinates (w, h_1, h_2) for a frame based on audio cepstral coefficients.
2. Generate deformed mesh vertices using interpVert.m.
3. Generate new image coordinates using image warping based on deformed mesh vertices for every frame.
4. Generate movie from warped frames and overlay given audio track using given executable file or using Matlab.

Workflow

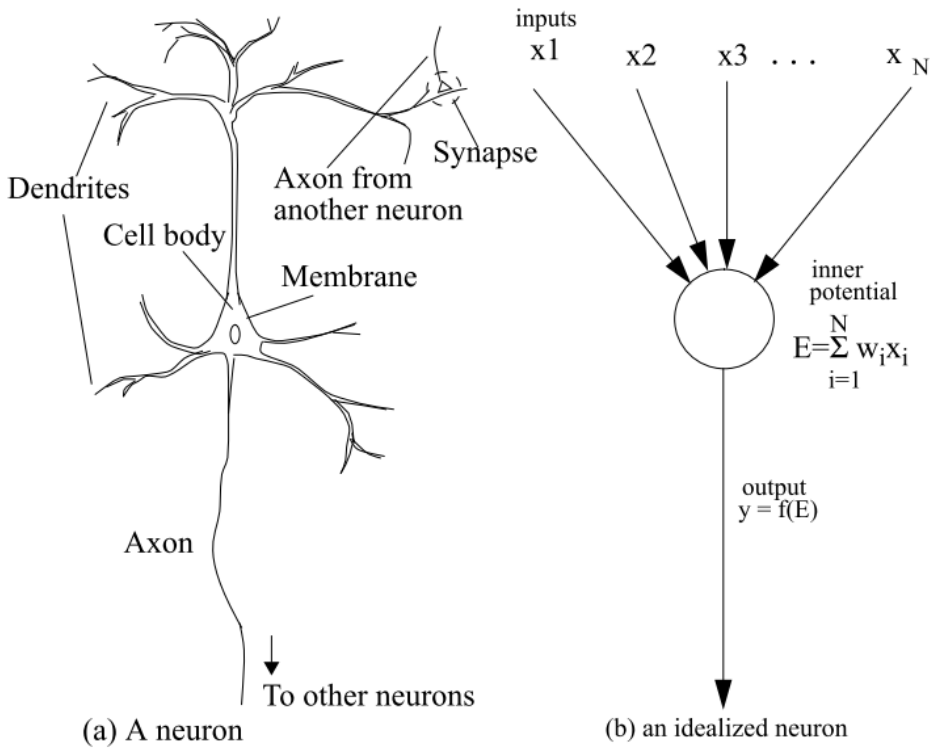
1. main.m - wrapper function for overall code
   - 1.1 Audio-Visual Training
     - 1.1.1 read neutral image
     - 1.1.2 load training data
     - 1.1.3 train Neural Network
   - 1.2 Audio-Visual Test Implementation
     - 1.2.1 use Neural Network to find visual features for given 456 frames of test data
   - 1.3 Image Deformation
     - 1.3.1 Iterate through each frame, and for each one apply warping function warping.m.
2. warping.m - function to deform neutral image according to mapping
   - 2.1 interpolate visual features into mesh
   - 2.2 for each triangle:
     - 2.2.1 if triangle has changed, then warp changed triangle using either Affine transformation or Barycentric coordinates
     - 2.2.2 interpolate new image using either nearest neighbor or bilinear interpolation
3. generate_movie.m - function to concatenate frames and create movie

   - 3.1 either use a system call to run given executeable or use Matlab to create a movie from individual frames and overlay audio

Perceptron (Artificial Neuron)

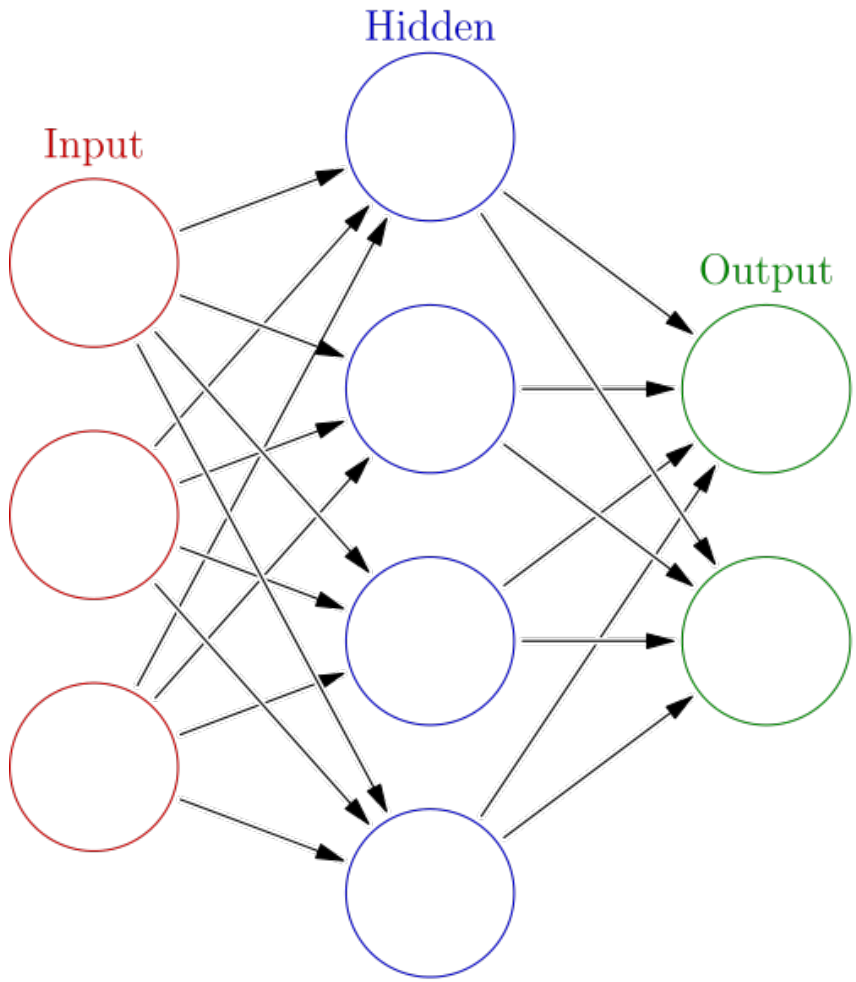
1. A perceptron is a binary classifier which maps a real-valued input $\mathbf{x}$ to an output value $f(\mathbf{x})$.
2. Output $+1$ if the result is greater than some threshold and -1 (or 0) otherwise.

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w}^{\mathbf{T}}\mathbf{x} > \mathbf{0} \\ 0 & \text{otherwise} \end{cases}$$



Feedforward Neural Network

1. The representation power of perceptron is limited (e.g. XOR function cannot be represented by a perceptron)
2. Use a network of perceptrons - Artifical Neural Network, or Multi-layer Perceptron



Neural Network I/O and Method

1. In MP5, use 3-hidden layer neural networks (numN=3) for audio-to-image mapping
2. Training function:
   [mapping] = ECE417_MP5_train (avTrainingData, avValidateData, silenceModel, numN, resultFile)
   - Feedforward neural networks are trained to map input audio features to the corresponding visual coordinate.
   - Train three neural networks for visual dimensions (w, h_1, h_2), respectively.
3. Testing function:
   [results] = ECE417_MP5_test (testAudio, silenceModel, mapping)
   - Given the trained neural networks and audio test features, the neural networks predicts visual coordinates.

Image Deformation/Warping for MP5

1. For each triangle find an affine transformation system deform the triangle.
2. Apply deformation to points inside given triangle.
3. Assign image values to triangle points

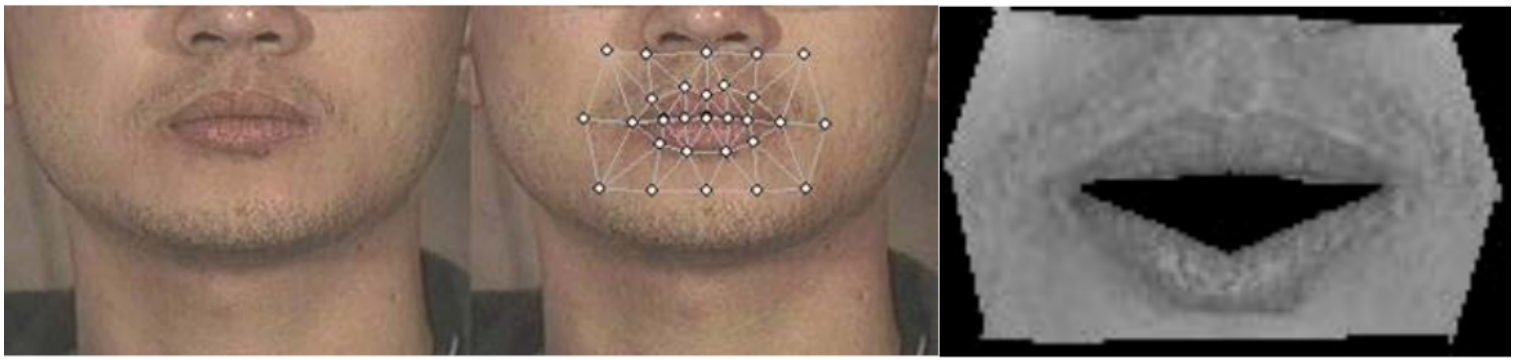


Figure : Neutral Mouth, Mouth Mesh, and Deformed Mouth

Warping for Trianglular Mesh

For each point in the image, you need to figure out which triangle it belongs to and then deform it using the transform that corresponds to that particular triangle in the mesh.

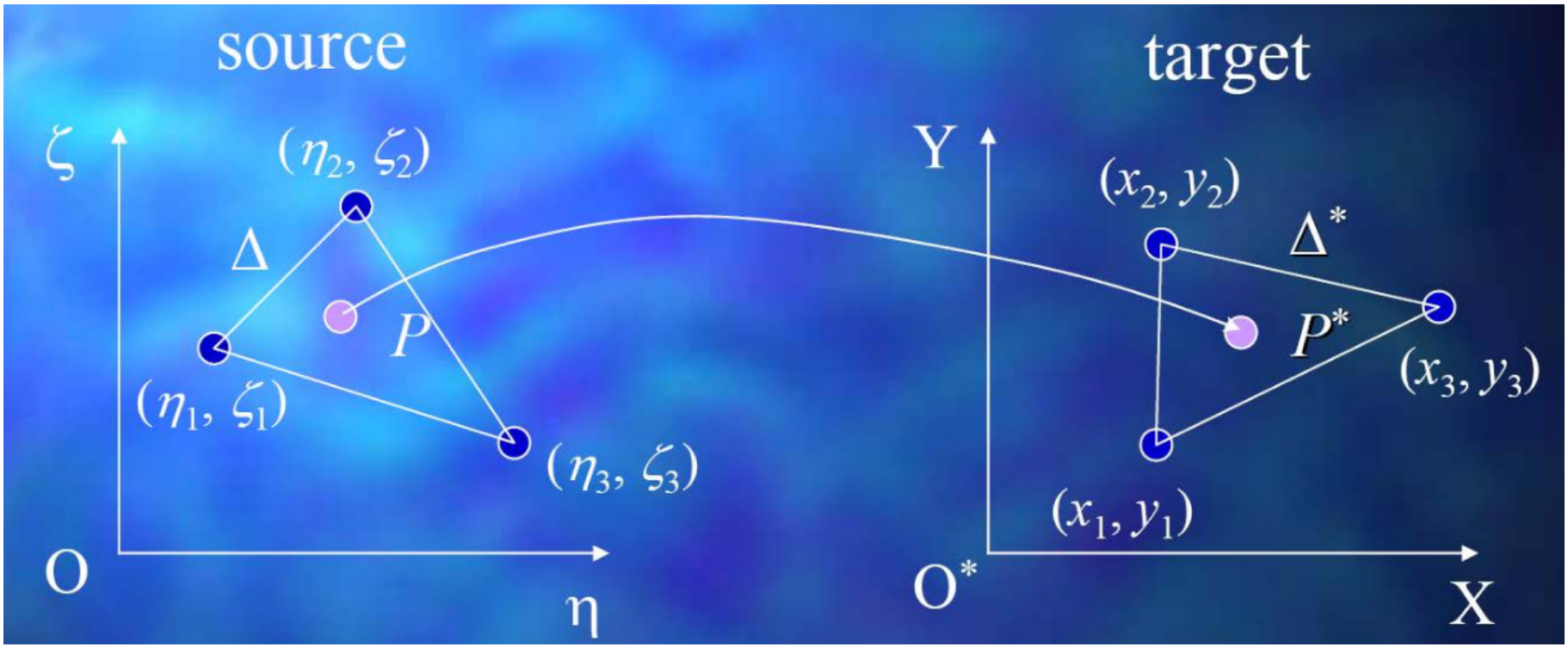


Figure : Triangle Affine Warping

Affine Transformations

The Affine transformation equation is (where the pair (η, ζ) is the source and (x, y) is the warped destination):

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

Can solve two separate equations for **a** and **b** using Least-Squares.

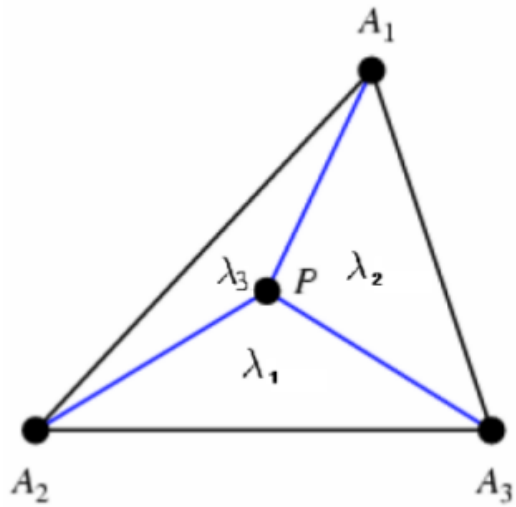
$$\begin{bmatrix} 1 & \eta_1 & \zeta_1 \\ \vdots & \vdots & \vdots \\ 1 & \eta_N & \zeta_N \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} \qquad \begin{bmatrix} 1 & \eta_1 & \zeta_1 \\ \vdots & \vdots & \vdots \\ 1 & \eta_N & \zeta_N \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \tag{2}$$

Can also just solve for both at once.

$$\begin{bmatrix} \eta_1 & \zeta_1 & 0 & 0 & 1 & 0 \\ 0 & 0 & \eta_1 & \zeta_1 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \eta_N & \zeta_N & 0 & 0 & 1 & 0 \\ 0 & 0 & \eta_N & \zeta_N & 0 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \\ a_0 \\ b_0 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ \vdots \\ x_N \\ y_N \end{bmatrix} \tag{3}$$

Barycentric Coordinates

For any (η, ζ) in a given mesh triangle P, we can write the coordinates as a function of the vertices of said triangle. In addition, homogeneous barycentric coordinates are invariant to affine transformations.

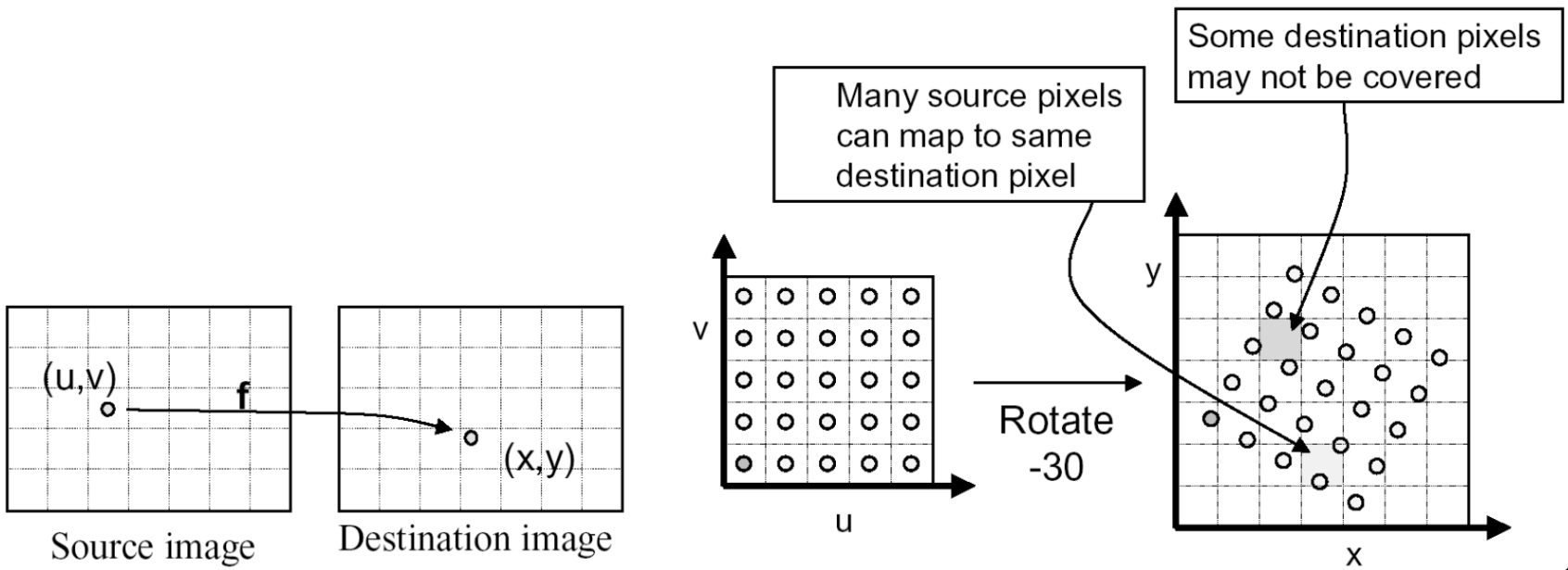


$$\begin{bmatrix} \zeta \\ \eta \\ 1 \end{bmatrix} = \begin{bmatrix} \zeta_1 & \zeta_2 & \zeta_3 \\ \eta_1 & \eta_2 & \eta_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} \qquad \begin{bmatrix} \zeta \\ \eta \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} \tag{4}$$

If the parameters $0 < \lambda_i < 1 \quad \forall i$ then when the system is solved for a given set of points (ζ, η) then said points must be in the triangle. In addition, if $0 \leq \lambda_i \leq 1$, then said points could be on *or* inside the triangle. Otherwise, the points cannot be inside the triangle (nor on it).

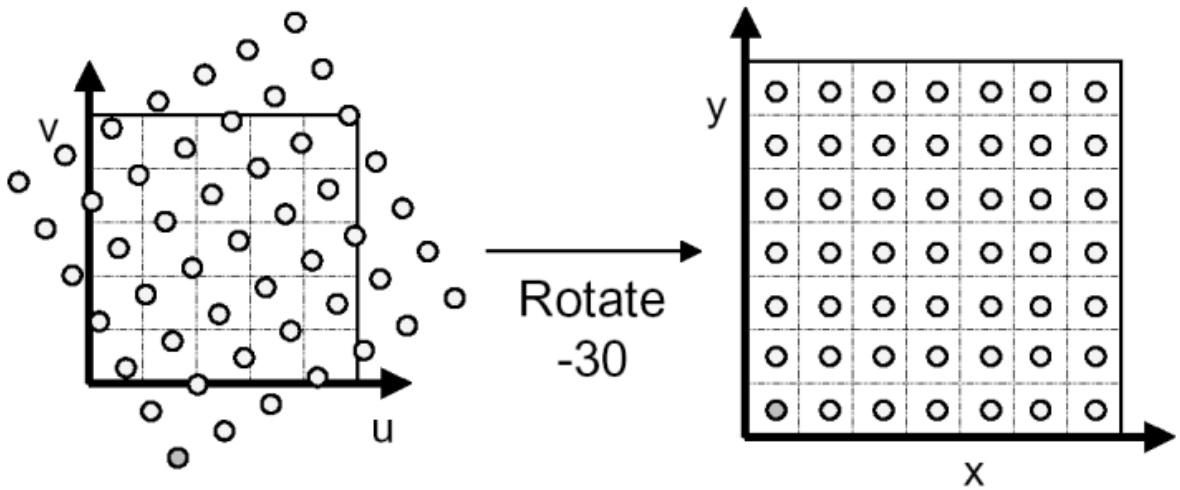
Forward Mapping

Forward: iterate over source image and map pixels in the source image to the destination image.



Backward Mapping

1. Backward: iterate over destination image and map pixels in the destination image back to original image.
2. may oversample, and will have to resample (interpolate) source (because non-integer index returns)

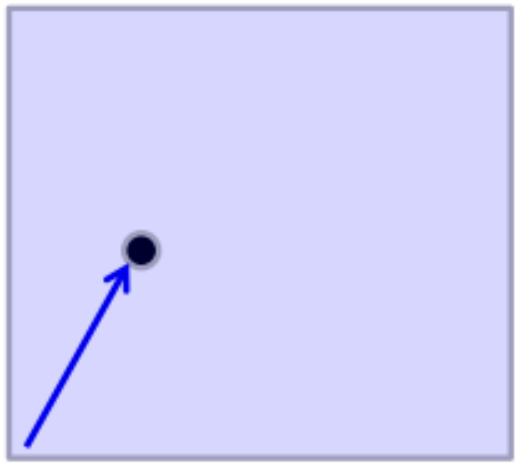


Resampling

1. Evaluate source image at arbitrary (u, v)
   - (u, v) does not usually have integer coordinates
2. Some kinds of resampling
   - Nearest neighbor
   - Bilinear interpolation
   - Gaussian filter

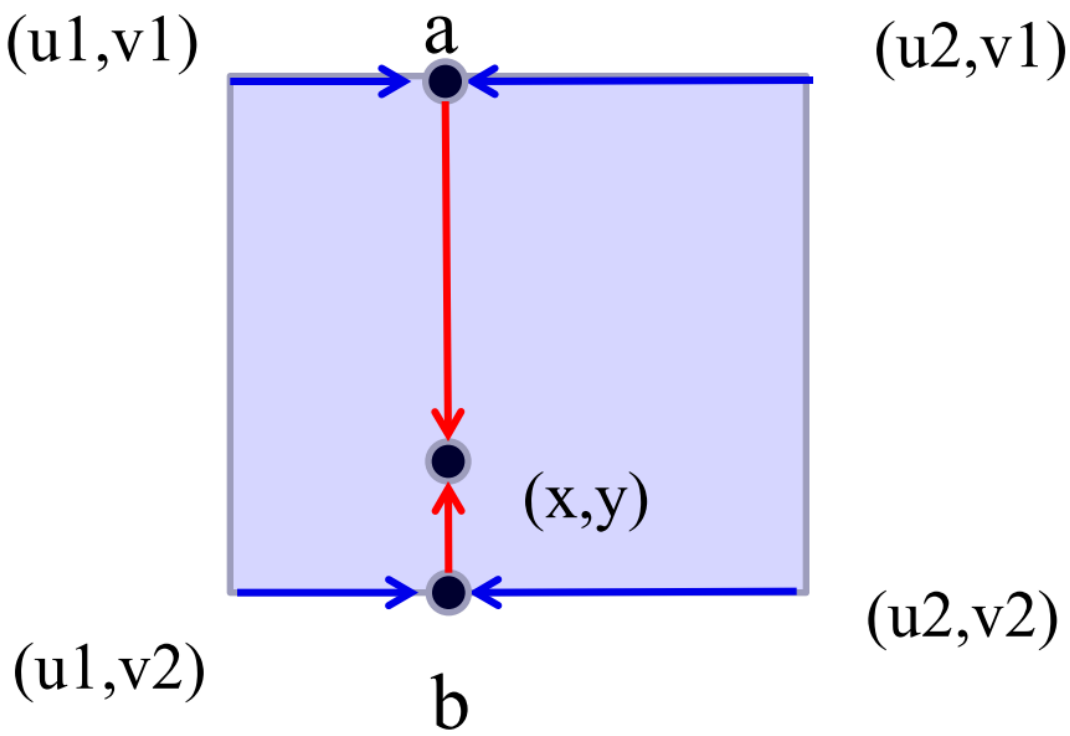
Nearest Neighbor Interpolation

1. Take value at closest pixel
   - int iu = trunc(u + 0.5);
   - int iv = trunc(v + 0.5);
   - dst(x, y) = src(iu, iv);
2. Simple, but causes aliasing



Bilinear Interpolation

1. Bilinearly interpolate four surrounding pixels
   - a = linear interpolation of src(u_1 , v_1) and src(u_2 , v_1)
   - b = linear interpolation of src(u_1 , v_2) and src(u_2 , v_2)
   - dst(x, y) = linear interpolation of 'a' and 'b'



Gaussian Kernel Interpolation

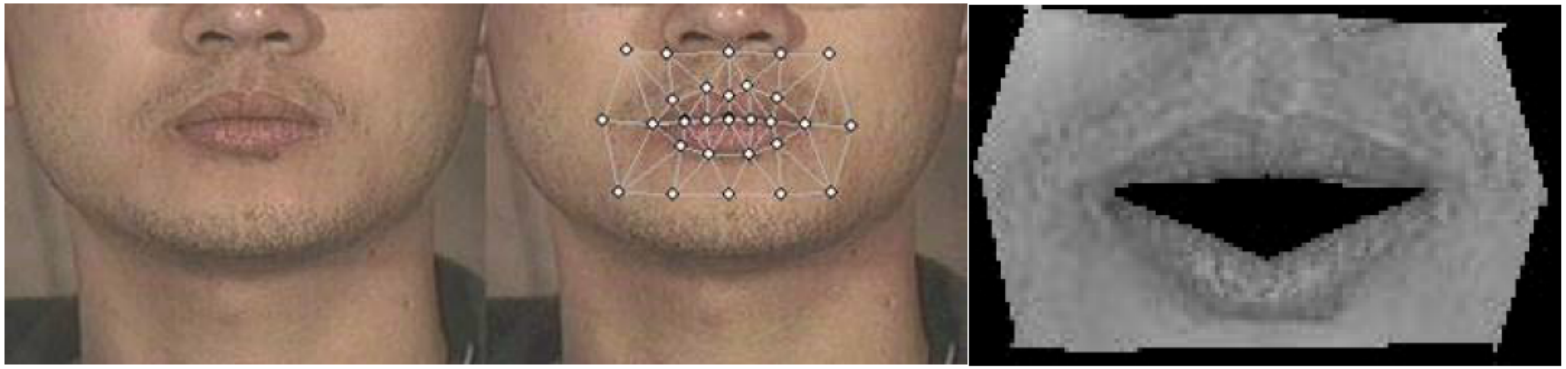
Convolve with Gaussian filter

Input

Output

Width of Gaussian kernel affects bluriness

Warping in MP5



1. Warping
   - For each triangle find the affine transform
   - Apply it to the points inside
   - Assign image value to the triangle points
2. Suggestions
   - Barycentric coordinate
   - Reverse mapping

Easy Mistakes to Watch Out For

1. Make sure you know how x and y correspond to rows and columns. If you are getting strange results or dimension mismatches, this is the first thing I would look at.

Deliverables

1. Code - A self-contained file that does everything listed in the requirements. You can write as many functions (or as few) as you wish, but there must be one overall function that will call everything automatically run without any hassle.
2. Report - Tell us about your general approach to this problem, and describes the algorithms you use. In addition, tell us the approach you used to generate the movie file, as this will probably vary across submisions.
3. Readme - a readme file that tells us how to run your code, and any options that need to be set.
4. Movie - a movie file (avi seems the best choice) that represents the total output of your algorithm

Matlab Functions You Are Allowed to Use / Might be Helpful

Remember, Matlab functions that are not yours, that you get online, or that trivialize the problem at hand are not allowed. In general, if you think a function is suspicious, then it probably is. Check with us if you are ever unsure about a function.

1. dlmread, fscanf
2. vertcat, horzcat, find, any, unique
3. bsxfun, cellfun, cell2mat, mat2cell, cell, struct, zeros, ones
4. inv, transpose, lu, qr, pinv, svd
5. system, movie, im2frame, movie2avi
6. Neural Network training function inherently called by functions in mp5.zip