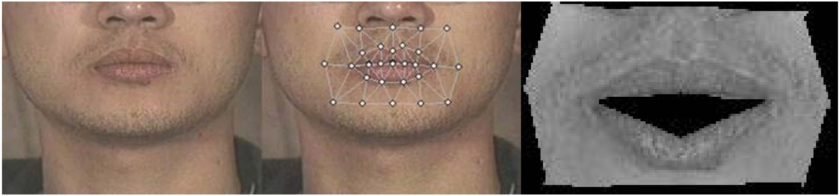
Lecture 21: Barycentric Coordinates and Deep Voxel Flow

Mark Hasegawa-Johnson

University of Illinois

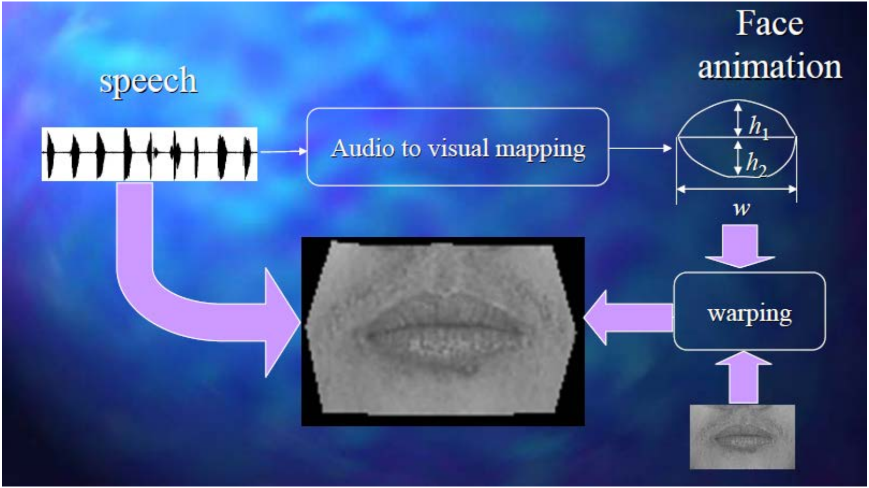
ECE 417: Multimedia Signal Processing, Fall 2020

Outline



Goal of MP4: Generate video frames (right) by warping a static image (left).

Talking head, full outline



How it is done

$$\begin{aligned}
\text{lip_height,width} &= \text{NeuralNet}\,(\text{audio features}) \\
\text{out_triangs} &= \text{LinearlyInterpolate}\,(\text{inp_triangs,lip_height,width}) \\
\text{inp_coord} &= \text{BaryCentric}\,(\text{out_coord,inp_triangs,out_triangs}) \\
\text{out_image} &= \text{BilinearInterpolate}\,(\text{inp_coord,inp_image})
\end{aligned}$$

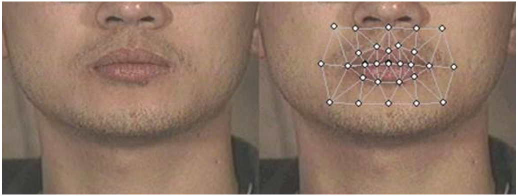
Outline


Affine Transformations
Affine Transformations
* Combines linear transformations, and Translations
TECH
TECH
x'
w
a b c
d e f
0 0 1
x
y
w
the ones we looked at, that were
the you know the rotation scaling and
0:26 / 1:19
CC
HD

Piece-wise affine transform

- OK, so somebody's given us a lot of points, arranged like this in little triangles.
- We know that we want a DIFFERENT AFFINE TRANSFORM for EACH TRIANGLE. For the k^{th} triangle, we want to have

$$A_k = \begin{bmatrix} a_k & b_k & c_k \\ d_k & e_k & f_k \\ 0 & 0 & 1 \end{bmatrix}$$

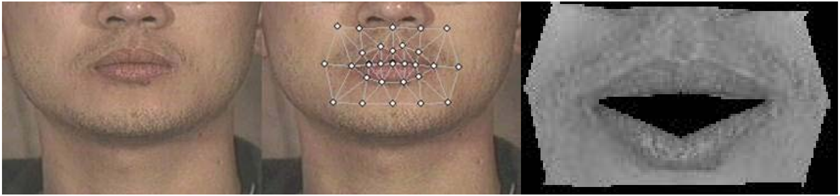


Piece-wise affine transform

$$\text{output point: } \vec{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \text{input point: } \vec{u} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Definition: if $\vec{x}$ is in the k^{th} triangle in the **output image**, then we want to use the k^{th} affine transform:

$$\vec{x} = A_k \vec{u}, \quad \vec{u} = A_k^{-1} \vec{x}$$



If **it is known that** $\vec{u} = A_k^{-1}\vec{x}$ for some unknown affine transform matrix A_k,

then

the method of barycentric coordinates finds $\vec{u}$

without ever finding A_k.

Barycentric Coordinates

Barycentric coordinates turns the problem on its head. Suppose $\vec{x}$ is in a triangle with corners at $\vec{x}_1$, $\vec{x}_2$, and $\vec{x}_3$. That means that

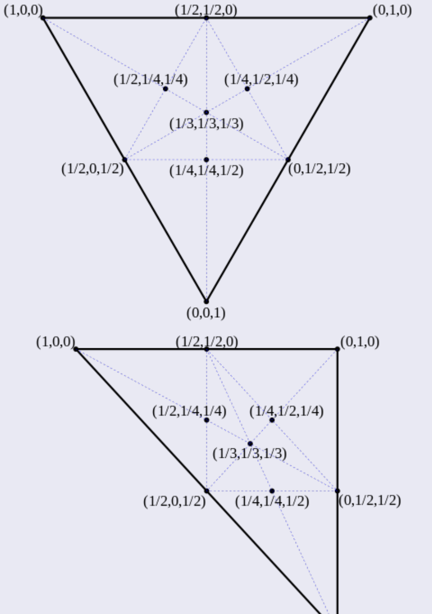
$$\vec{x} = \lambda_1 \vec{x}_1 + \lambda_2 \vec{x}_2 + \lambda_3 \vec{x}_3$$

where

$$0 \leq \lambda_1, \lambda_2, \lambda_3 \leq 1$$

and

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$



Barycentric Coordinates

Suppose that all three of the corners are transformed by some affine transform A, thus

$$\vec{u}_1 = A\vec{x}_1, \quad \vec{u}_2 = A\vec{x}_2, \quad \vec{u}_3 = A\vec{x}_3$$

Then if

$$\text{If: } \vec{x} = \lambda_1\vec{x}_1 + \lambda_2\vec{x}_2 + \lambda_3\vec{x}_3$$

Then:

$$\begin{aligned}\vec{u} &= A\vec{x}\\ &= \lambda_1 A\vec{x}_1 + \lambda_2 A\vec{x}_2 + \lambda_3 A\vec{x}_3\\ &= \lambda_1\vec{u}_1 + \lambda_2\vec{u}_2 + \lambda_3\vec{u}_3\end{aligned}$$

In other words, once we know the λ's, we no longer need to find A. We only need to know where the corners of the triangle have moved.

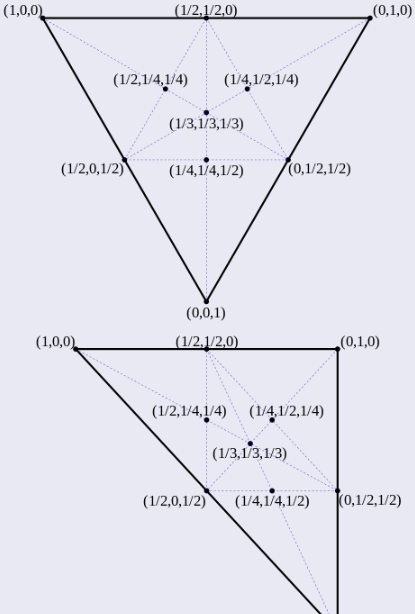
Barycentric Coordinates

If

$$\vec{x} = \lambda_1 \vec{x}_1 + \lambda_2 \vec{x}_2 + \lambda_3 \vec{x}_3$$

Then

$$\vec{u} = \lambda_1 \vec{u}_1 + \lambda_2 \vec{u}_2 + \lambda_3 \vec{u}_3$$



How to find Barycentric Coordinates

But how do you find λ_1, λ_2, and λ_3?

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \lambda_1\vec{x}_1 + \lambda_2\vec{x}_2 + \lambda_3\vec{x}_3 = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}$$

Write this as:

$$\vec{x} = X\vec{\lambda}$$

Therefore

$$\vec{\lambda} = X^{-1}\vec{x}$$

This **always works:** the matrix X is always invertible, unless all three of the points $\vec{x}_1$, $\vec{x}_2$, and $\vec{x}_3$ are on a straight line.

How do you find out which triangle the point is in?

- Suppose we have K different triangles, each of which is characterized by a 3×3 matrix of its corners

$$X_k = [\vec{x}_{1,k}, \vec{x}_{2,k}, \vec{x}_{3,k}]$$

  where $\vec{x}_{m,k}$ is the m^{th} corner of the k^{th} triangle.
- Notice that, for any point $\vec{x}$, for ANY triangle X_k, we can find

$$\lambda = X_k^{-1}\vec{x}$$

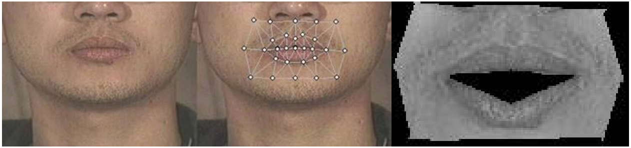
- However, the coefficients λ_1, λ_2, and λ_3 will all be between 0 and 1 **if and only if** the point $\vec{x}$ is inside the triangle X_k. Otherwise, some of the λ's must be negative.

The Method of Barycentric Coordinates

To construct the animated output image frame $J[y, x]$, we do the following things:

- First, for each of the reference triangles U_k in the input image $I(u, v)$, decide where that triangle should move to. Call the new triangle location X_k.
- Second, for each output pixel (x, y):
  - For each of the triangles, find $\vec{\lambda} = X_k^{-1}\vec{x}$.
  - Choose the triangle for which all of the λ coefficients are $0 \leq \lambda \leq 1$.
  - Find $\vec{u} = U_k\vec{\lambda}$.
  - Estimate $I(u, v)$ using bilinear interpolation.
  - Set $J[y, x] = I(v, u)$.

How to Make a Talking Head



$$\text{lip\_height,width} = \text{NeuralNet}\,(\text{audio features})$$
$$\text{out\_triangs} = \text{LinearlyInterpolate}\,(\text{inp\_triangs,lip\_height,width})$$
$$\text{inp\_coord} = \text{BaryCentric}\,(\text{out\_coord,inp\_triangs,out\_triangs})$$
$$\text{out\_image} = \text{BilinearInterpolate}\,(\text{inp\_coord,inp\_image})$$

Outline

Video Frame Synthesis Using Deep Voxel Flow

Liu et al., ICCV 2017

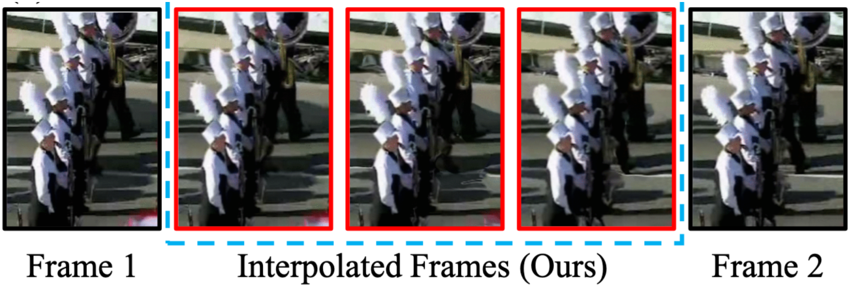


Image (c) ICCV and the authors

Video Frame Synthesis Using Deep Voxel Flow

Liu et al., ICCV 2017

- Objective: Given video frames at times 0 and 1, generate missing frame at time $t \in (0, 1)$.
- Voxel Flow: Generated frame is made by copying pixels from frames 0 and 1, with some shift in position, $(\Delta x, \Delta y)$.
- The coordinate shift $(\Delta x, \Delta y)$ is (almost) a **piece-wise affine** function of (x, y), so it is (almost) equivalent to a mapping based on Barycentric coordinates—but without ever explicitly choosing the triangle locations.
- When $(x - \Delta x, y - \Delta y)$ are non-integer, the input pixels are constructed using **bilinear interpolation**.

Voxel Flow

The generated frame, $\hat{\mathbf{Y}}(y, x, t)$, is generated as a linear convex interpolation between selected pixels of the two reference images, $\mathbf{X}(y, x, 0)$ and $\mathbf{X}(y, x, 1)$:

$$\hat{\mathbf{Y}}(y, x, t) = (1 - \Delta t)\,\mathbf{X}\,(y - \Delta y, x - \Delta x, 0) + \Delta t\mathbf{X}\,(y + \Delta y, x + \Delta x, 1)$$

where $\Delta t \in (0, 1)$.

Piece-Wise (Nearly) Affine

The voxel flow field is generated as

$$\mathbf{F} = (\Delta x, \Delta y, \Delta t) = \mathcal{H}(\mathbf{X}; \theta)$$

where $\mathcal{H}(\mathbf{X}; \theta)$ uses:

- A series of CNN layers with ReLU nonlinearity, to compute a piece-wise affine function of $\mathbf{X}$, then
- A final layer with a tanh nonlinearity, squashing the output to the range $\Delta x \in (-1, 1)$, $\Delta y \in (-1, 1)$.

Piece-Wise (Nearly) Affine

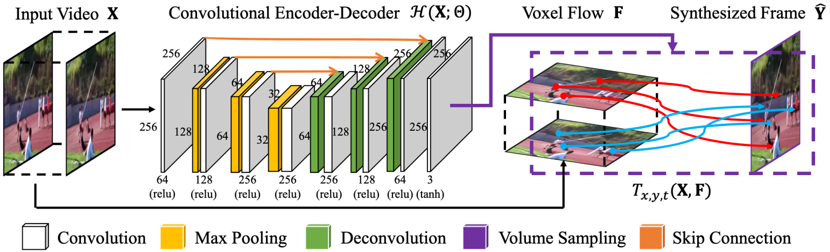


Image (c) ICCV and the authors

Bilinear Interpolation

The reference pixels, $(y - \Delta y, x - \Delta x)$ and $(y + \Delta y, x + \Delta x)$, are usually not integers, so they are constructed using bilinear interpolation:

$$\hat{\mathbf{Y}}(y, x, t) = \sum_{i,j,k \in \{0,1\}} \mathbf{W}^{ijk} \mathbf{X}(\mathbf{V}^{ijk}),$$

where:

$$\begin{aligned}
\mathbf{V}^{000} &= (\lfloor x - \Delta x \rfloor, \lfloor y - \Delta y \rfloor, 0) \\
\mathbf{V}^{100} &= (\lceil x - \Delta x \rceil, \lfloor y - \Delta y \rfloor, 0) \\
&\vdots \\
\mathbf{V}^{111} &= (\lceil x + \Delta x \rceil, \lceil y + \Delta y \rceil, 1)
\end{aligned}$$

and the weights $\mathbf{W}^{ijk}$ are constructed according to bilinear interpolation.

Differentiable

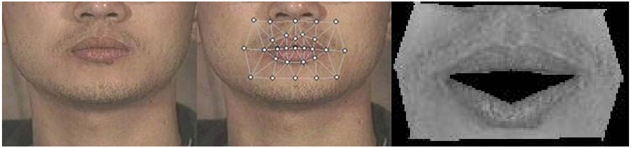
Because bilinear interpolation is a piece-wise linear function of Δx and Δy, the error can be differentiated w.r.t. those parameters. From the original paper:

$$\frac{\partial \hat{\mathbf{Y}}(x,y)}{\partial(\Delta x)} = \sum_{i,j,k \in [0,1]} \mathbf{E}^{ijk}\mathbf{X}(\mathbf{V}^{ijk}),$$

$$\begin{aligned}
\mathbf{E}^{000} &= (1 - (\mathbf{L}_y^0 - \lfloor \mathbf{L}_y^0 \rfloor))(1 - \Delta t) \\
\mathbf{E}^{100} &= -(1 - (\mathbf{L}_y^0 - \lfloor \mathbf{L}_y^0 \rfloor))(1 - \Delta t) \\
&\vdots \\
\mathbf{E}^{011} &= -(\mathbf{L}_y^1 - \lfloor \mathbf{L}_y^1 \rfloor)\Delta t \\
\mathbf{E}^{111} &= (\mathbf{L}_y^1 - \lfloor \mathbf{L}_y^1 \rfloor)\Delta t,
\end{aligned}$$

Outline

How to Make a Talking Head



$$\begin{aligned}\text{lip\_height,width} &= \text{NeuralNet}\,(\text{audio features})\\ \text{out\_triangs} &= \text{LinearlyInterpolate}\,(\text{inp\_triangs,lip\_height,width})\\ \text{inp\_coord} &= \text{BaryCentric}\,(\text{out\_coord,inp\_triangs,out\_triangs})\\ \text{out\_image} &= \text{BilinearInterpolate}\,(\text{inp\_coord,inp\_image})\end{aligned}$$

Barycentric Coordinates

- For each of the triangles, find $\vec{\lambda} = X_k^{-1}\vec{x}$.
- Choose the triangle for which all of the λ coefficients are $0 \le \lambda \le 1$.
- Find $\vec{u} = U_k\vec{\lambda}$.
- Estimate $I(v,u)$ using bilinear interpolation.

$$I(v,u) = \sum_m \sum_n I[n,m]h(v-n, u-m)$$

- Set $J[y,x] = I(v,u)$.

Deep Voxel Flow: PWL⇒End-to-end differentiable

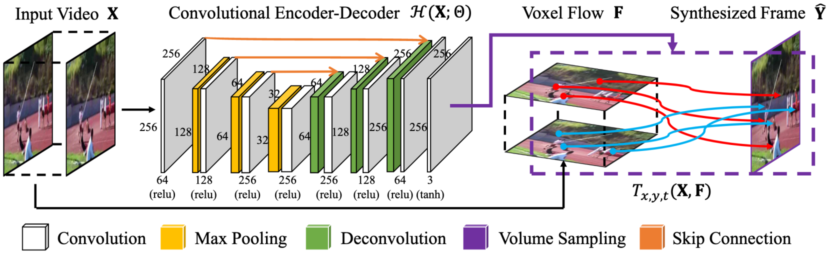


Image (c) ICCV and the authors