Lecture 21: Barycentric Coordinates and Deep Voxel Flow

Mark Hasegawa-Johnson
All content CC-SA 4.0 unless otherwise specified.

University of Illinois

ECE 417: Multimedia Signal Processing, Fall 2020
1. How to Make a Talking Head
2. Barycentric Coordinates
3. Deep Voxel Flow
4. Conclusion
Outline

1. How to Make a Talking Head
2. Barycentric Coordinates
3. Deep Voxel Flow
4. Conclusion
**Goal of MP4:** Generate video frames (right) by warping a static image (left).
Talking head, full outline

speech

Audio to visual mapping

Face animation

warping
How it is done

\[
\text{lip_height,width} = \text{NeuralNet (audio features)} \\
\text{out_triangs} = \text{LinearlyInterpolate (inp_triangs, lip_height, width)} \\
\text{inp_coord} = \text{BaryCentric (out_coord, inp_triangs, out_triangs)} \\
\text{out_image} = \text{BilinearInterpolate (inp_coord, inp_image)}
\]
Outline

1. How to Make a Talking Head
2. Barycentric Coordinates
3. Deep Voxel Flow
4. Conclusion
Affine Transformations

* Combines linear transformations, and Translations

\[
\begin{bmatrix}
    x' \\
    y' \\
    w'
\end{bmatrix} =
\begin{bmatrix}
    a & b & c \\
    d & e & f \\
    0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    x \\
    y \\
    w
\end{bmatrix}
\]

the ones we looked at, that were the you know the rotation scaling and
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

output point: \( \vec{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \), 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

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

Then:

\[
\vec{u} = A\vec{x} = \lambda_1 A\vec{x}_1 + \lambda_2 A\vec{x}_2 + \lambda_3 A\vec{x}_3
\]

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

In other words, once we know the \( \lambda \)’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 $\lambda_1$, $\lambda_2$, and $\lambda_3$?

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

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 $3 \times 3$ 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 $\lambda_1$, $\lambda_2$, and $\lambda_3$ 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 $\lambda$'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 \( \lambda \) coefficients are \( 0 \leq \lambda \leq 1 \).
  - Find \( \vec{u} = U_k \vec{\lambda} \).
  - Estimate \( I(u, v) \) using bilinear interpolation.
  - Set \( J[y, x] = I(v, u) \).
lip_height, width = NeuralNet (audio features)
out_triangs = LinearlyInterpolate (inp_triangs, lip_height, width)
inp_coord = BaryCentric (out_coord, inp_triangs, out_triangs)
out_image = BilinearInterpolate (inp_coord, inp_image)
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, $\hat{Y}(y, x, t)$, is generated as a linear convex interpolation between selected pixels of the two reference images, $X(y, x, 0)$ and $X(y, x, 1)$:

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

where $\Delta t \in (0, 1)$.
The voxel flow field is generated as

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

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 $\Delta x \in (-1, 1)$, $\Delta y \in (-1, 1)$. 
How to Make a Talking Head

Barycentric Coordinates

Deep Voxel Flow

Conclusion

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:

\[
\hat{Y}(y, x, t) = \sum_{i,j,k\in\{0,1\}} W_{ijk} \mathbf{X}(\mathbf{V}_{ijk}),
\]

where:

\[
\mathbf{V}^{000} = ([x - \Delta x], [y - \Delta y], 0)
\]

\[
\mathbf{V}^{100} = ([x - \Delta x], [y - \Delta y], 0)
\]

\vdots

\[
\mathbf{V}^{111} = ([x + \Delta x], [y + \Delta y], 1)
\]

and the weights \(W_{ijk}\) are constructed according to bilinear interpolation.
Because bilinear interpolation is a piece-wise linear function of $\Delta x$ and $\Delta y$, the error can be differentiated w.r.t. those parameters. From the original paper:

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

$$E^{000} = (1 - (L_y^0 - \lfloor L_y^0 \rfloor))(1 - \Delta t)$$
$$E^{100} = - (1 - (L_y^0 - \lfloor L_y^0 \rfloor))(1 - \Delta t)$$
$$\vdots$$
$$E^{011} = - (L_y^1 - \lfloor L_y^1 \rfloor) \Delta t$$
$$E^{111} = (L_y^1 - \lfloor L_y^1 \rfloor) \Delta t,$$
lip_height, width = NeuralNet (audio features)
out_triangs = LinearlyInterpolate (inp_triangs, lip_height, width)
inp_coord = BaryCentric (out_coord, inp_triangs, out_triangs)
out_image = BilinearInterpolate (inp_coord, inp_image)
Barycentric Coordinates

For each of the triangles, find $\vec{\lambda} = X_k^{-1} \vec{x}$.

Choose the triangle for which all of the $\lambda$ coefficients are $0 \leq \lambda \leq 1$.

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 $\Rightarrow$ End-to-end differentiable

Input Video $X$

Convolutional Encoder-Decoder $\mathcal{H}(X; \Theta)$

Voxel Flow $F$

Synthesized Frame $\hat{Y}$

$T_{x,y,t}(X, F)$

Convolution $\square$

Max Pooling $\square$

Deconvolution $\square$

Volume Sampling $\square$

Skip Connection $\square$

Image (c) ICCV and the authors