Lecture 25: Affine Transformations and Barycentric Coordinates

ECE 417: Multimedia Signal Processing
Mark Hasegawa-Johnson

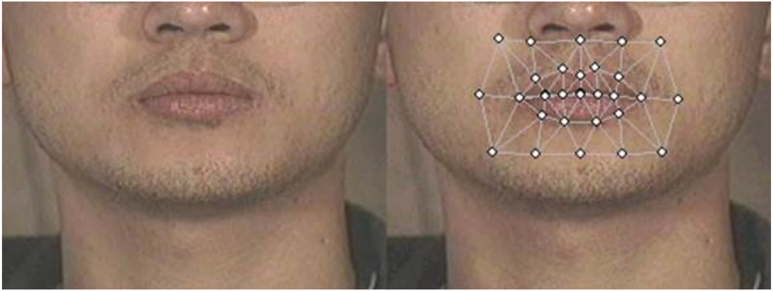
University of Illinois

11/28/2017

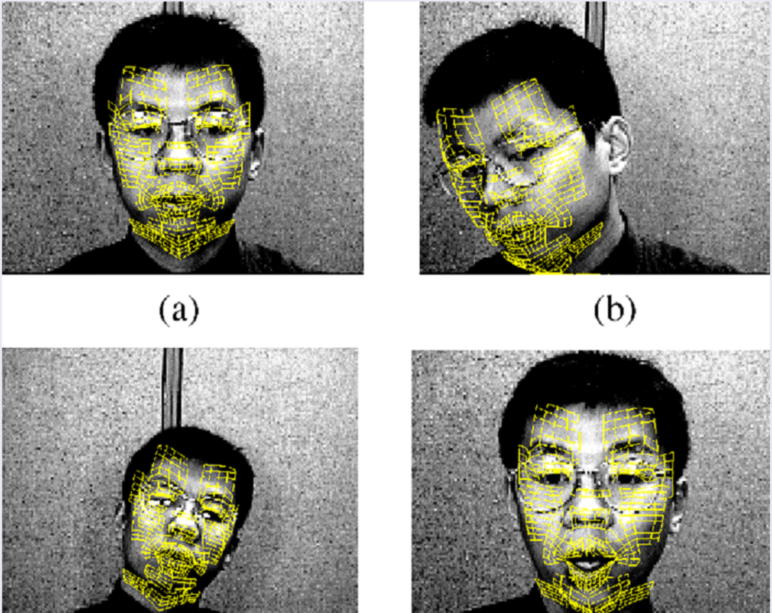
Outline

Moving Points Around

First, let's suppose that somebody has given you a bunch of points:



. . . and let's suppose you want to move them around, to create new images. . .



Moving One Point

- Your goal is to synthesize an output image, $I(x, y)$, where $I(x, y)$ might be intensity, or RGB vector, or whatever.
- What you have available is:
  - An input image, $I_0(u, v)$
  - Knowledge that the input point at (u, v) has been **moved** to the output point at (x, y).
- Therefore:

$$I(x, y) = I_0(u, v)$$

Non-Integer Input Points

- Usually, we can't make both (x, y) and (u, v) to be integers.
- The easiest thing is to make (x, y) be integers. In other words, create the output image as "*for x=1:M, for y=1:N, I(x,y) = I0(u,v);*"
- In that case, (u, v) are not integers. So what is $I_0(u, v)$?

Non-Integer Input Points

Suppose that $p = \text{int}(u)$, and $q = \text{int}(v)$. Then:

- Piece-wise constant interpolation:

$$I(x,y) = I_0(p,q)$$

- Bilinear interpolation:

$$I(x,y) = \sum_{m=-1}^{0} \sum_{n=-1}^{0} h[m,n] I_0(p-m, q-n)$$

where $\sum_m \sum_n w[m,n] = 1$.

- General interpolation (e.g., spline, sinc):

$$I(x,y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} h[m,n] I_0(p-m, q-n)$$

Outline

How do we find (u, v)?

Now the question: how do we find (u, v)?
We're going to assume that this is a piece-wise affine transformation.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a_k & b_k \\ d_k & e_k \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c_k \\ f_k \end{bmatrix}$$

where a_k etc. depend on which region (x, y) is in.

How do we find (u, v)?

Piece-wise affine means:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a_k & b_k \\ d_k & e_k \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c_k \\ f_k \end{bmatrix}$$

A much easier to write this is by using extended-vector notation:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a_k & b_k & c_k \\ d_k & e_k & f_k \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

It's convenient to define $\vec{u} = [u, v, 1]^T$, and $\vec{x} = [x, y, 1]^T$, so that for any $\vec{x}$ in the k^{th} region of the image,

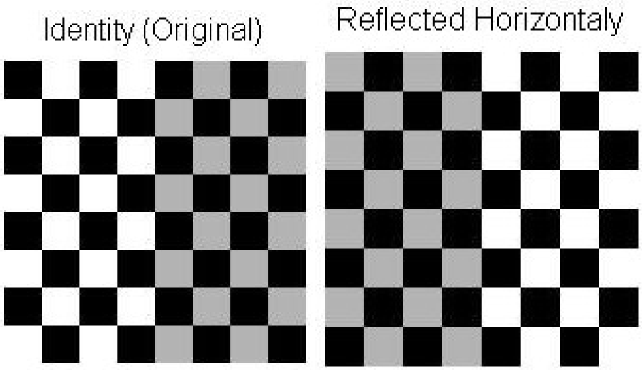
$$\vec{u} = A_k \vec{x}$$

Affine Transforms

Affine transforms can do the following things:

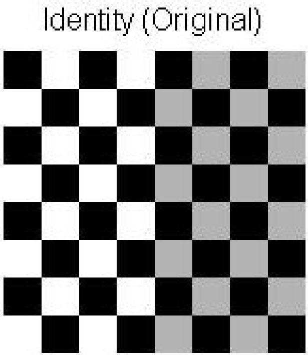
- Shift the input (to left, right, up, down)
- Reflect the input (through any line of reflection)
- Scale the input (separately in x and y directions)
- Rotate the input
- Skew the input

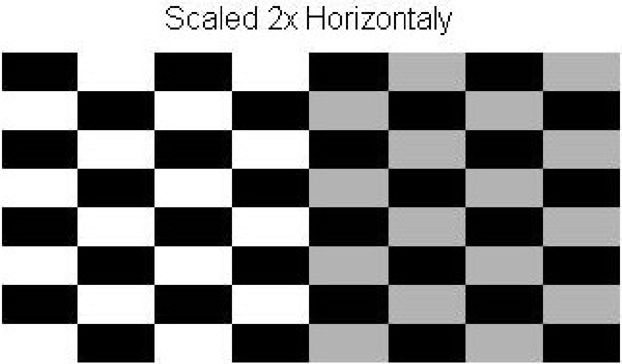
Example: Reflection



$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

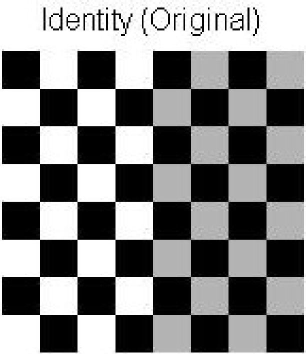
Example: Scale

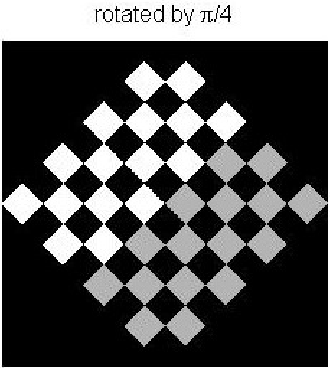




$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

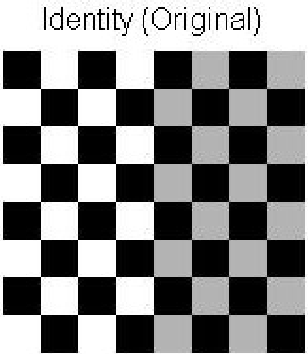
Example: Rotation

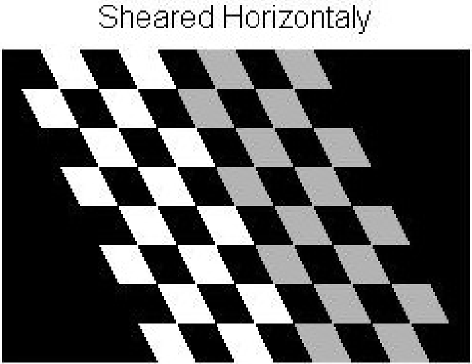




$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Example: Shear





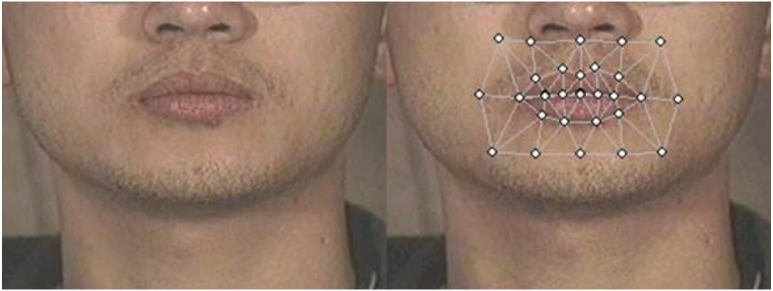
$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$


Affine Transformations
Affine Transformations
* Combines linear transformations, and Translations
TECH
TECH
x'
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
HD

Outline

How do we find the parameters?

- OK, so somebody's given us a lot of points, arranged like this in little triangles.
- We know that we want to scale, shift, rotate, and shear each triangle separately with its own 6-parameter affine transform matrix A_k.
- How do we find A_k for each of the triangles?



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

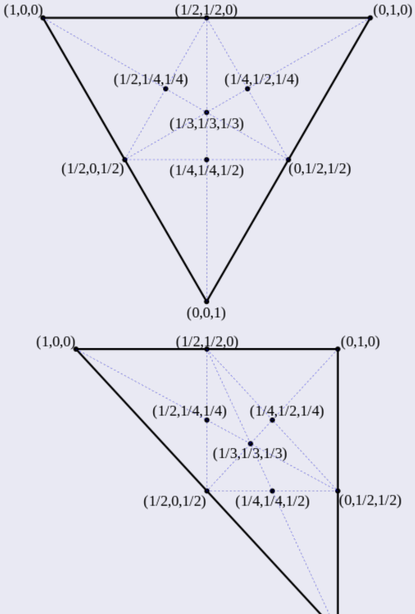
$$\vec{x} = \lambda_1\vec{x}_1 + \lambda_2\vec{x}_2 + \lambda_3\vec{x}_3$$

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.

How to find Barycentric Coordinates

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

$$\vec{x} = \lambda_1\vec{x}_1 + \lambda_2\vec{x}_2 + \lambda_3\vec{x}_3 = [\vec{x}_1, \vec{x}_2, \vec{x}_3]\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}$$

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.

Outline

Conclusion: The Whole Method

To construct the animated output image frame $I(x, y)$, we do the following things:

- First, for each of the reference triangles U_k in the input image $I_0(u, v)$, decide where that triangle should move to. Call the new triangle location X_k.
- Second, for each output pixel $I(x, y)$:
  - 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 $I_0(u, v)$ using piece-wise constant interpolation, or bilinear interpolation, or spline or sinc interpolation.
  - Set $I(x, y) = I_0(u, v)$.