Geometric Image Manipulation

Lecture #4

Wednesday, January 24, 2018

Colorado State University

Programming Assignment #1

Colorado State University

Image Manipulation: Context

- To start with the obvious, an image is a 2D array of pixels
  - Pixel locations represent points on the image plane
  - Pixel values represent measurements of light
    - Color images : energies by frequency ranges (RGB: three overlapping ranges)
    - Intensity images : average energy across the visible range
  - Building ray tracers should have taught you about image formation

- To directly compare two images, they should be *registered*
  - Geometrically : image 1 should "lines up with" image 2
  - Photometrically : equal pixel values should imply equal energy

Geometric Registration

- It's not enough for two matching images to have the same set of pixel values
- They have to be in the same relative positions

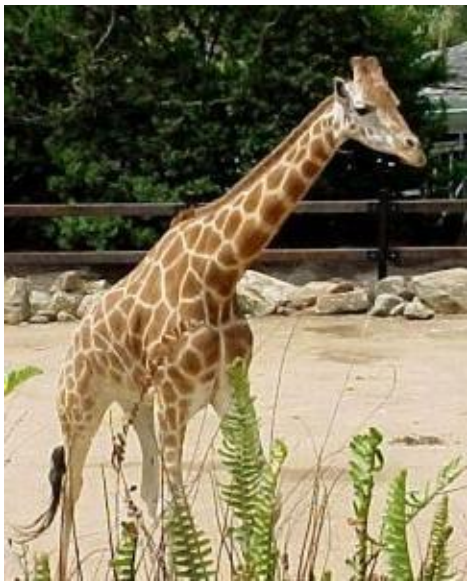
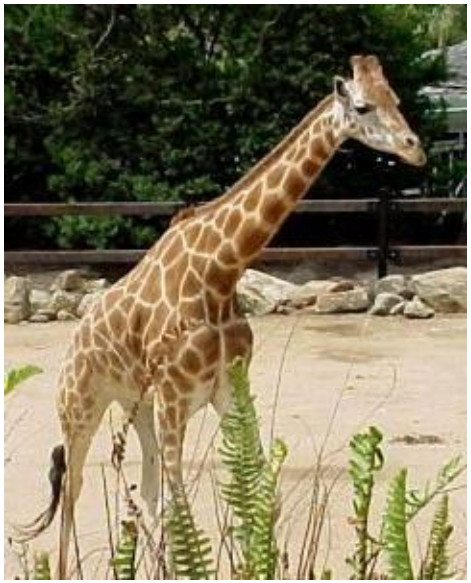


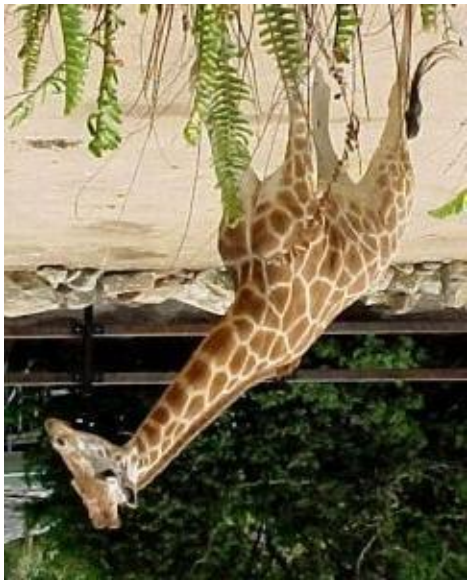
Image from CalTech256 data set

Otherwise, these two images match!

Geometric Registration (II)

- Geometric registration finds a mapping that maps one image onto the other
  - We will limit ourselves to linear transformation





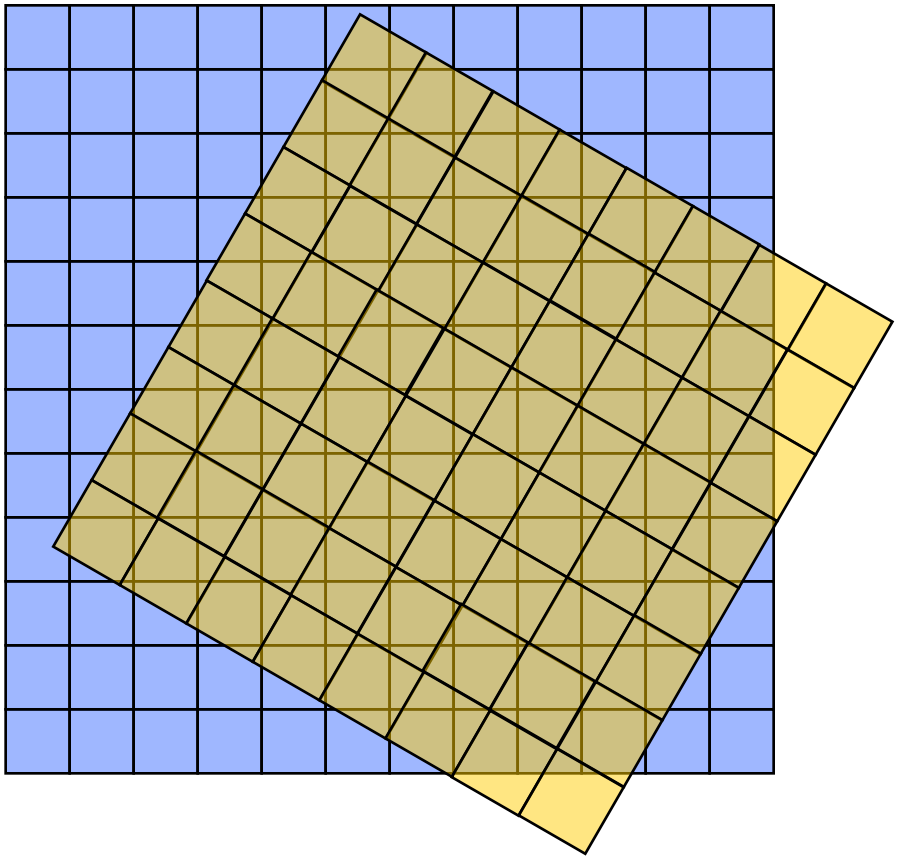
We should be able to register these…

Registration formalism

- We denote an image as a 2D function $I(x, y)$
- Or, in homogeneous coordinates, $I(x, y, w)$
- Given
  - two image I_i and I_j
  - Matching points {(u,v),...} & {(x,y),…}
  - Find G such that $I_i(u, v, w) \cong I_j\left(G \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}\right)$

Interpolation (foreshadow…)

- Seldom get integer-to-integer mappings.
- Geometry part computes real-valued positions of pixel centers.
- We will worry about how to interpolate values later.



Classes of Image Transformations

- Rigid transformations
  - Combine rotation and translation
  - Preserve relative distances and angles
  - 3 Degrees of freedom
- Similarity transformations
  - Add scaling to rotation and translation
  - Preserves relative angles
  - 4 Degrees of freedom

Building Blocks: Rotation

- Trigonometric version

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

- Projection onto basis vectors

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \hat{U}_1 & \hat{U}_2 & 0 \\ \hat{V}_1 & \hat{V}_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \hat{U} \cdot \hat{U} = 1 \quad \hat{V} \cdot \hat{V} = 1 \quad \hat{U} \cdot \hat{V} = 0$$

Building Blocks: Scaling

- Uniform scaling

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

- Non-uniform scaling

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

Building Blocks: Translation

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

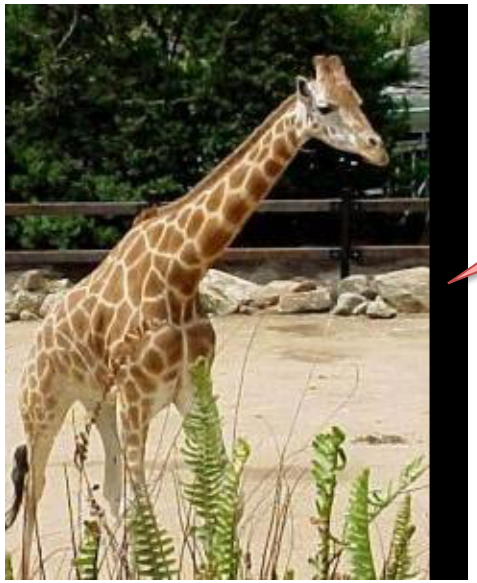
Recall how homogenous coordinates formulates translation as a matrix multiply

$$Q = M P$$

Translation Applied to Images

Translate 20 in x

$$\begin{bmatrix} 1 & 0 & 20 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



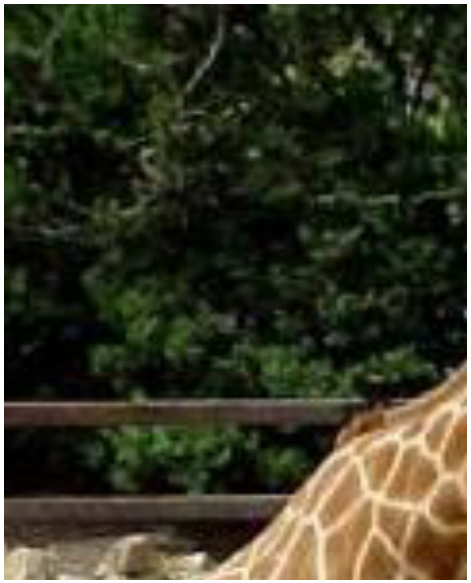
Why Black ?

Translate -20 in x

$$\begin{bmatrix} 1 & 0 & -20 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

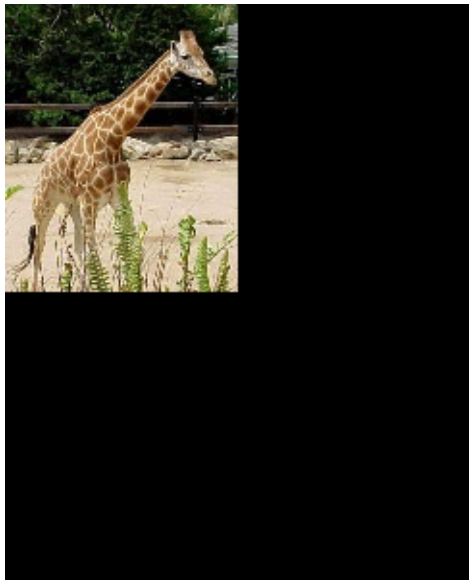
Scale Applied to Images

Note the origin



Scale Uniformly by 2

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

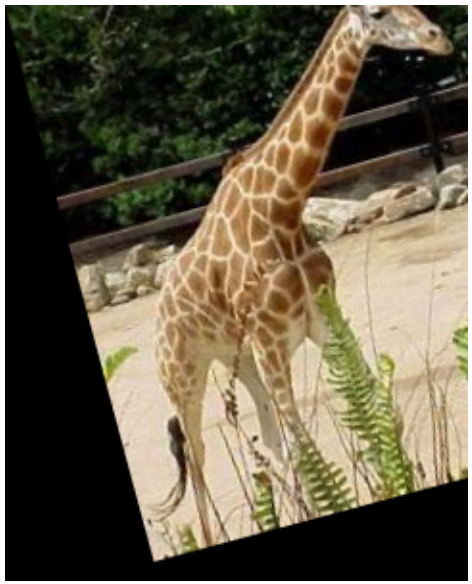


Scale Uniformly by 0.5

$$\begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation Applied to Images

Rotate by 15˚



Rotate by -15˚

Note that a positive rotation rotates the positive X axis toward the positive Y axis

Composition of Matrices

To rotate by θ around a point (x,y):

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

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

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & y\sin(\theta) - x\cos(\theta) + x \\ \sin(\theta) & \cos(\theta) & -x\sin(\theta) - y\cos(\theta) + y \\ 0 & 0 & 1 \end{bmatrix}$$

Affine Transformations

- All the similarity transforms can be combined into one generic matrix:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

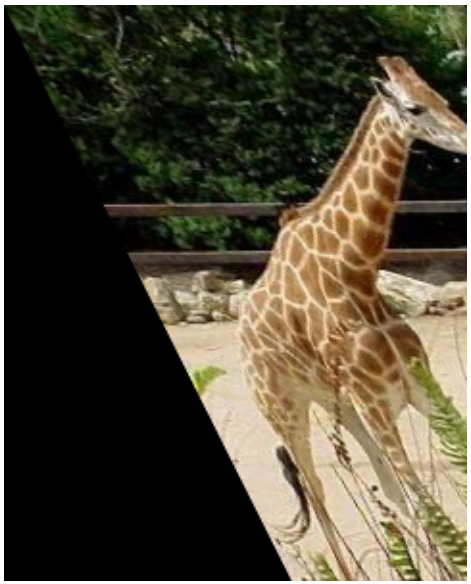
Hint: diagonal terms are not equal, and b ≠ -d.

- But! This matrix does more. What?
  - hint: 2 more transformations.
  - hint: 6 degrees of freedom.

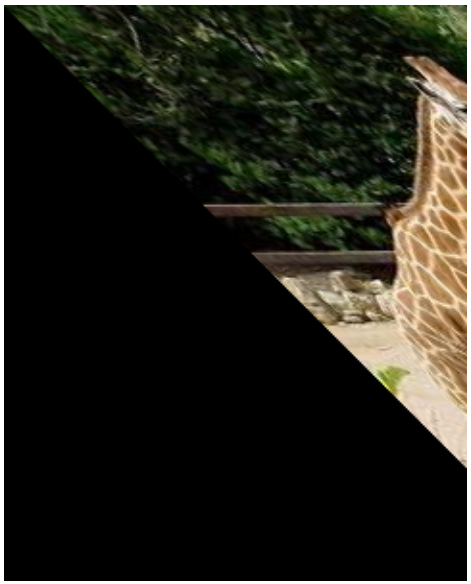
Equivalent to adding 2 shear parameters, or unequal scaling & 1 shear parameter.

- How can you specify this matrix?

Affine Examples: Shear



$$\begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Similarity vs. Affine Matrices

- Similarity : 4 DOF

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

- Affine : 6 DOF

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Specifying Affine Transformations

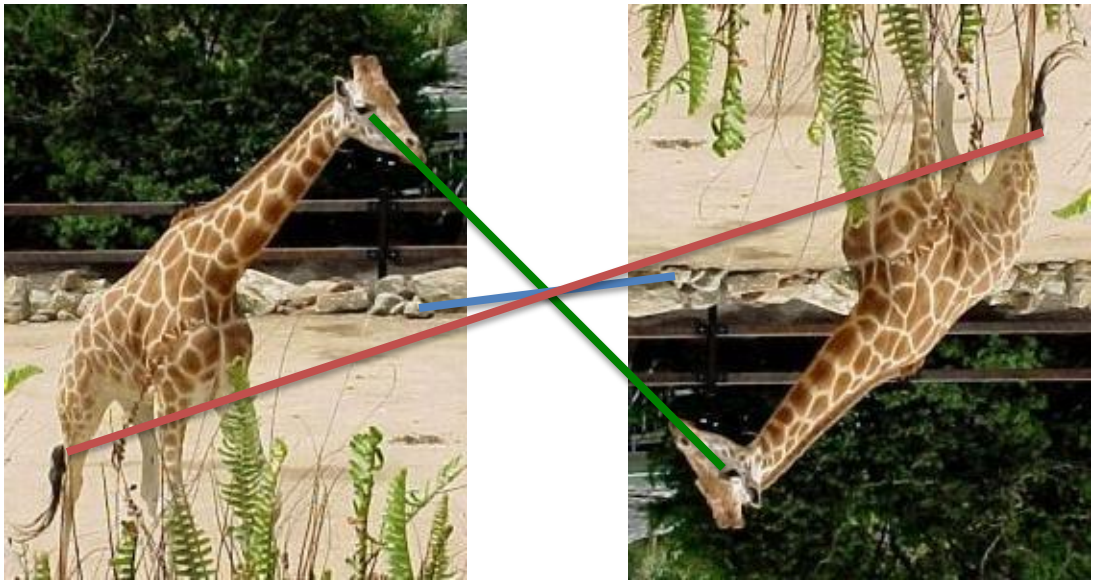
- There are six unknowns in the matrix (*a* through *f*)
- If you specify one point in the source image and a corresponding point in the target image, that yields two equations:

$$u_i = ax_i + by_i + c$$

$$v_i = dx_i + ey_i + f$$

- So providing three point-to-point correspondences specifies an affine matrix

Affine Specification: Example



There is one affine transformation that will map the green point on the right to the green point on the left, and align the red and blue points also.

Solving Affine Transformations

These linear equations can be easily solved:

- WLOG, assume $x_1=y_1=0$
- then $u_1= c$ and $v_1 = f$
- so:

Calculation of *a, b* & *c* is independent of calculation of *e, f* & *g.*

$$u_2 = ax_2 + by_2 + u_1$$

$$u_3 = ax_3 + by_3 + u_1$$

$$a = \frac{u_2 - u_1 - by_2}{x_2}$$

$$\frac{x_3(u_2 - u_1 - by_2)}{x_2} = u_3 - u_1 - by_3$$

$$\left(\frac{-x_3y_2}{x_2} - y_3\right)b = u_3 - u_1 - \frac{x_3}{x_2}(u_2 - u_1)$$

$$b = \frac{u_3 - u_1 - \frac{x_3}{x_2}(u_2 - u_1)}{\frac{-x_3y_2}{x_2} - y_3} = \frac{x_2(u_3 - u_2) - x_3(u_2 - u_1)}{-x_3y_2 - y_3x_2}$$

Solving Affine (cont.)

- This can be substituted in to solve for a
- The same process with y's solves for d,e,f
- About the WLOG:
  - It was true because you can translate the original coordinate system by $(-x_1, -y_1)$
  - So what do you do to compensate?
- Alternatively, set up a system of linear equations and solve...
  - Will show this for a harder case shortly….