Interpolation

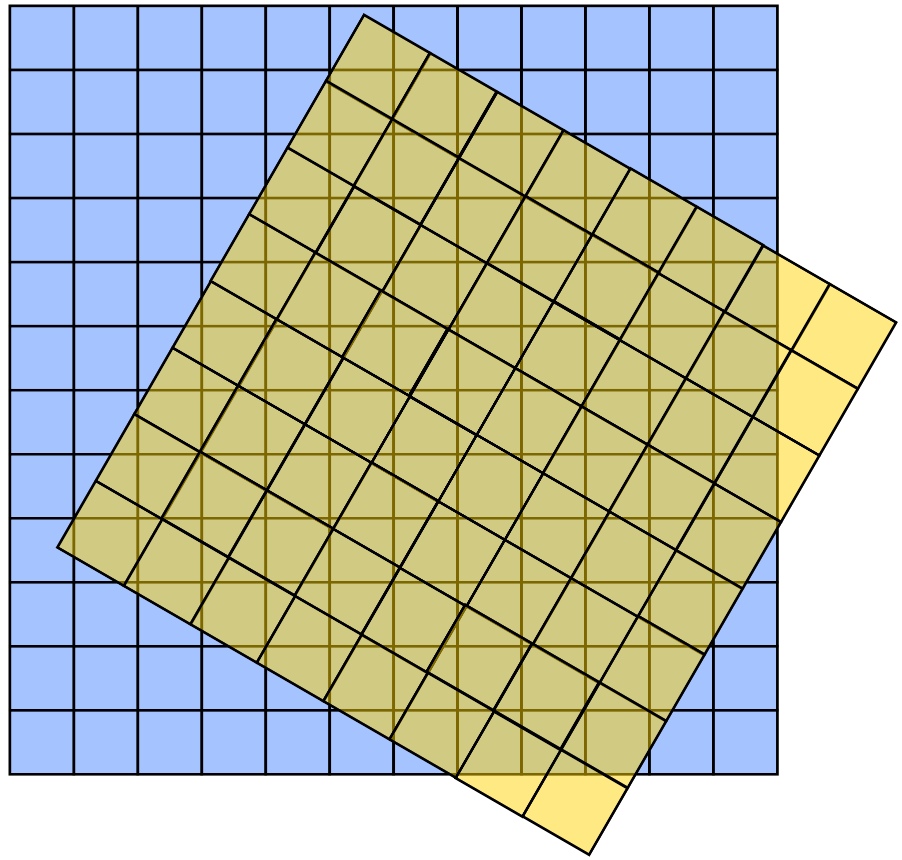
Lecture #2

January 28, 2013

Colorado State University

Image Transformation

- $I(x,y) = I'(G \bullet [x,y]^T)$
- Simple for continuous, infinite images
- Problematic for discrete, finite images



Source & Destination Images

- We apply a transformation to a source image to produce a destination image
- The role of source & destination are not symmetric
  - We need to know where every destination pixel came from in the source image
    - Typically, a non-integer location
  - We do not need to know where every source pixel went
    - It might be off the edge of the destination image, e.g.

Basic Transformation Algorithm

```
For every (x,y) in dest
{
  real (u,v) = G^-1 (x,y)
  pixel p = Interpolate(Src, u, v)
  dest(x,y) = p
}
```

Applying Transformations

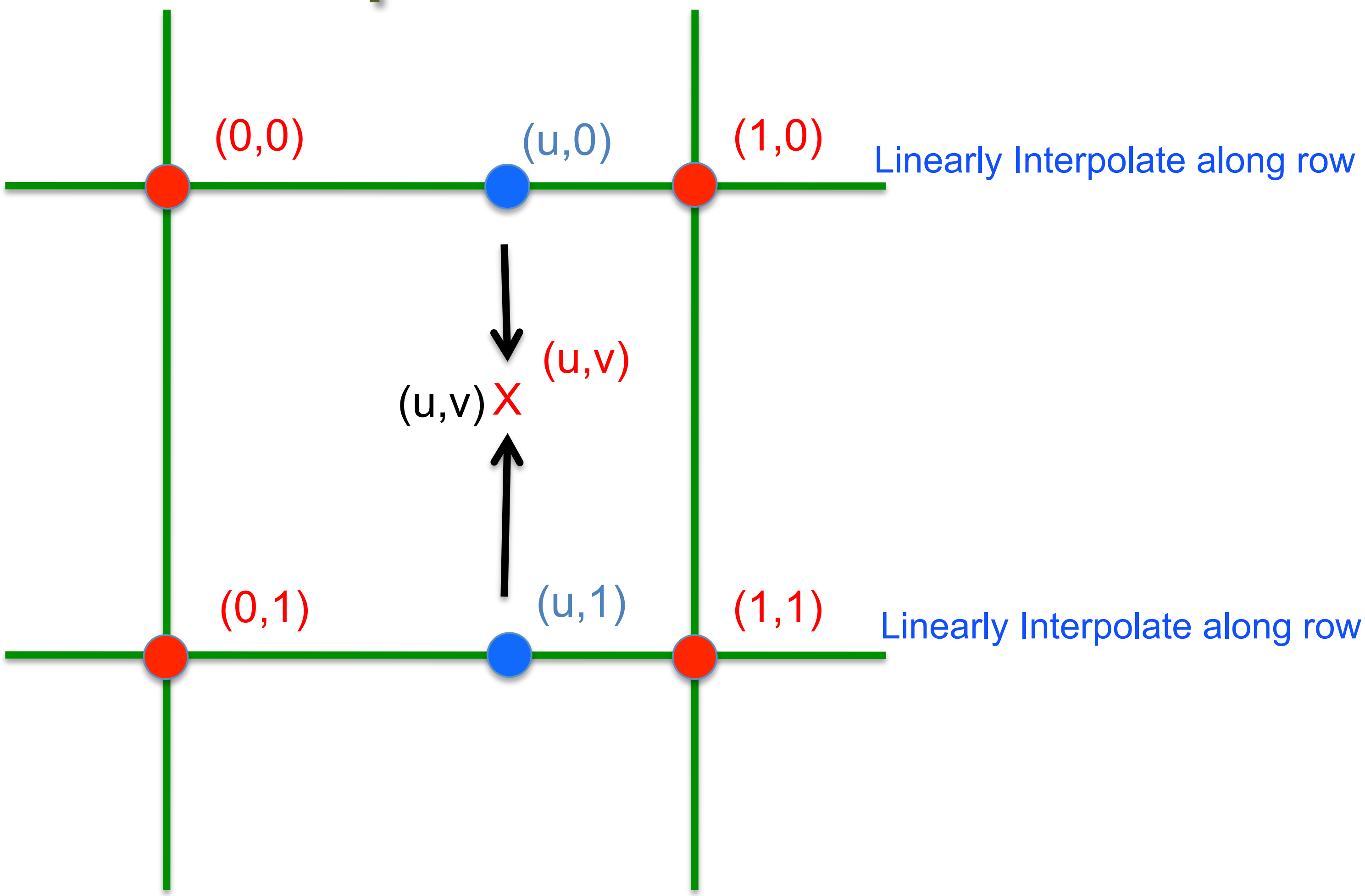
- I assume you can invert a 3x3 matrix
- So the trick is interpolation. 3 forms:
  - Nearest Neighbor (fast, bad)
  - Bilinear (less fast, good)
  - Bicubic (slowest, best)

Nearest Neighbor Interpolation

- $(u', v') = G^{-1}(x, y, 1)$
- $u = \text{round}(u')$
- $v = \text{round}(v')$
- Interpolate(Src, x, y) = Src[u,v]

For those who know Fourier Analysis, this is awful in the frequency domain

Bilinear Interpolation



Colorado State University

Bilinear Interpolation (II)

- Bilinear interpolation is actually a product of two linear interpolations
  - … and therefore non-linear
- Typical expression:

$$I(u,v) = I(0,0)(1-x)(1-y) + I(1,0)x(1-y) + I(0,1)(1-x)y + I(1,1)xy$$

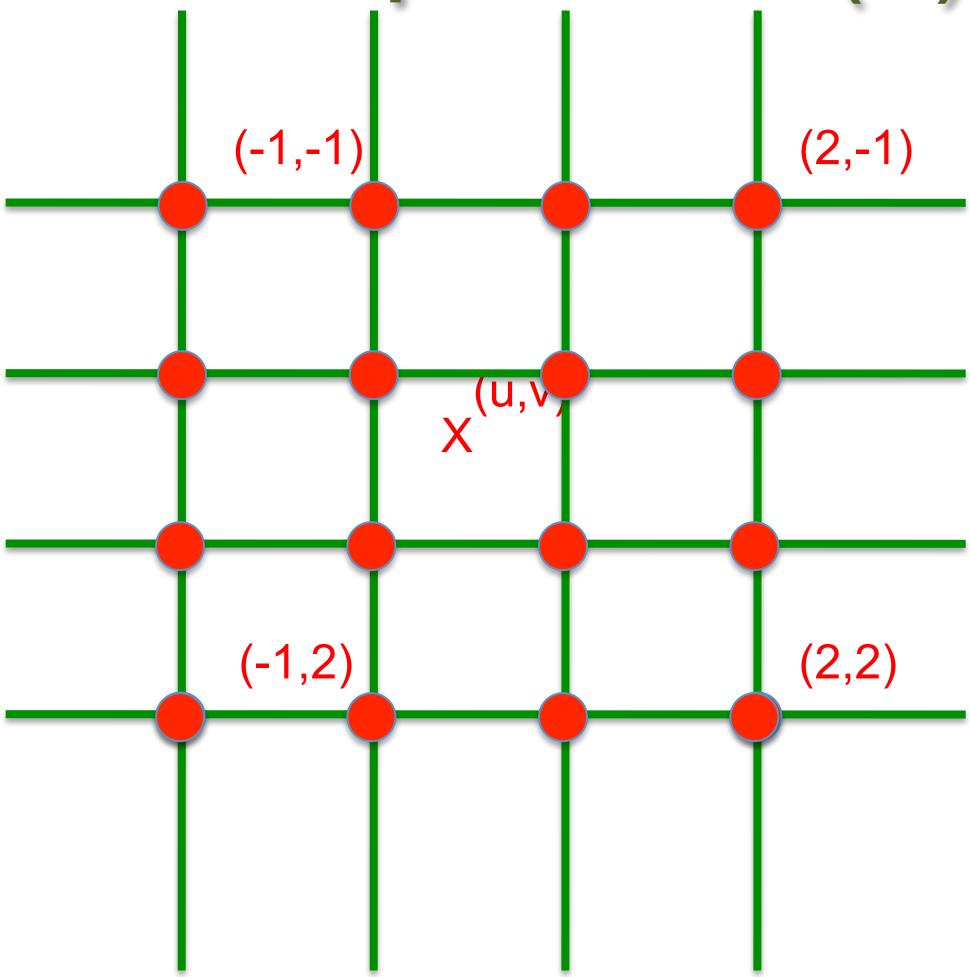
- Linear algebraic expression

$$I(x,y) = [1-x, x]\begin{bmatrix} I(0,0) & I(0,1) \\ I(1,0) & I(1,1) \end{bmatrix}\begin{bmatrix} 1-y \\ y \end{bmatrix}$$

Bicubic Interpolation

- Product of two cubic interpolations
  - 1 in x, 1 in y
- Based on a 4x4 grid of neighboring pixels
- In each dimension, create a cubic curve that exactly interpolates all four points
  - Similar to Bezier curves in graphics
  - Except curve passes through all 4 points

Bicubic Interpolation (II)



Colorado State University

Bicubic Interpolation (III)

- The equation of a cubic function is:

$$f(x) = ax^3 + bx^2 + cx + d$$

- This can be rewritten as:

$$f(x) = \begin{bmatrix} x^3 & x^2 & x & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

- We know the values of f at -1,0,1,2

Bicubic Interpolation (IV)

- Therefore:

$$\begin{bmatrix} f(-1) \\ f(0) \\ f(1) \\ f(2) \end{bmatrix} = \begin{bmatrix} -1 & 1 & -1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 8 & 4 & 2 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

- And:

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \frac{1}{12} \begin{bmatrix} -2 & 6 & -4 & 0 \\ 6 & -12 & -6 & 10 \\ -6 & -2 & 12 & 0 \\ 2 & 0 & -2 & 0 \end{bmatrix} \begin{bmatrix} f(-1) \\ f(0) \\ f(1) \\ f(2) \end{bmatrix}$$

Bicubic Interpolation (V)

- To interpolate a value:
  1. Interpolate along the four rows
     - Calculate a, b, c, d
     - Use a,b,c,d to calculate value at new x
  2. Interpolate the results vertically
- Each interpolation is a matrix/vector multiply
  - 20 mults, 15 adds per interpolation
  - 100 mults, 75 adds overall

The Anti-climax

- You don't need to implement geometric transformations of interpolations
- OpenCV supports geometric transformations
  - warpAffine applies an affine transformation
  - warpPerspective applies a perspective transformation
  - Both give you the option of interpolation technique
    - Nearest Neighbor
    - Bilinear
    - Bicubic
- The point of these lectures is so that you would know what was happening when you used them