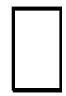
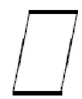
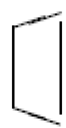
Robert Collins
CSE486, Penn State

Lecture 14
Parameter Estimation

Readings T&V Sec 5.1 - 5.3

Summary: Transformations

Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\, I \,\middle|\, t \,\right]_{2\times 3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\left[\, R \,\middle|\, t \,\right]_{2\times 3}$	3	lengths $+\cdots$	
similarity	$\left[\, sR \,\middle|\, t \,\right]_{2\times 3}$	4	angles $+\cdots$	
affine	$\left[\, A \,\right]_{2\times 3}$	6	parallelism $+\cdots$	
projective	$\left[\, H \,\right]_{3\times 3}$	8	straight lines	

$$\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$ **Euclidean**

$$\begin{bmatrix} sR & t \\ 0 & 1 \end{bmatrix}$$ **similarity**

$$\begin{bmatrix} A & b \\ 0 & 1 \end{bmatrix}$$ **affine**

$$\begin{bmatrix} A & b \\ c^T & d \end{bmatrix}$$ **projective**

Parameter Estimation

We will talk about estimating parameters of

1) Geometric models (e.g. lines, planes, surfaces)
2) Geometric transformations (any of the parametric transformations we have been talking about)

Least-squares is a general strategy to address both!

Parameter Estimation: Fitting Geometric Models

General Idea:

- **Want to fit a model to raw image features (data)**
  (the features could be points, edges, even regions)
- **Parameterize model such that**
  model instance is an element of R^n
  i.e. model instance = $(a_1, a_2, \ldots, a_n)$
- **Define an error function $E(\text{model}_i\text{, data})$ that measures how well a given model instance describes the data**
- **Solve for the model instance that minimizes E**

Example : Line Fitting

General Idea:

- **Want to fit a model to raw image features (data)**
  (the features could be points, edges, even regions)
- **Parameterize model such that**
  model instance is an element of R^n
  i.e. model instance = $(a_1, a_2, \ldots, a_n)$
- **Define an error function $E(\text{model}_i$, data) that measures how well a given model instance describes the data**
- **Solve for the model instance that minimizes E**

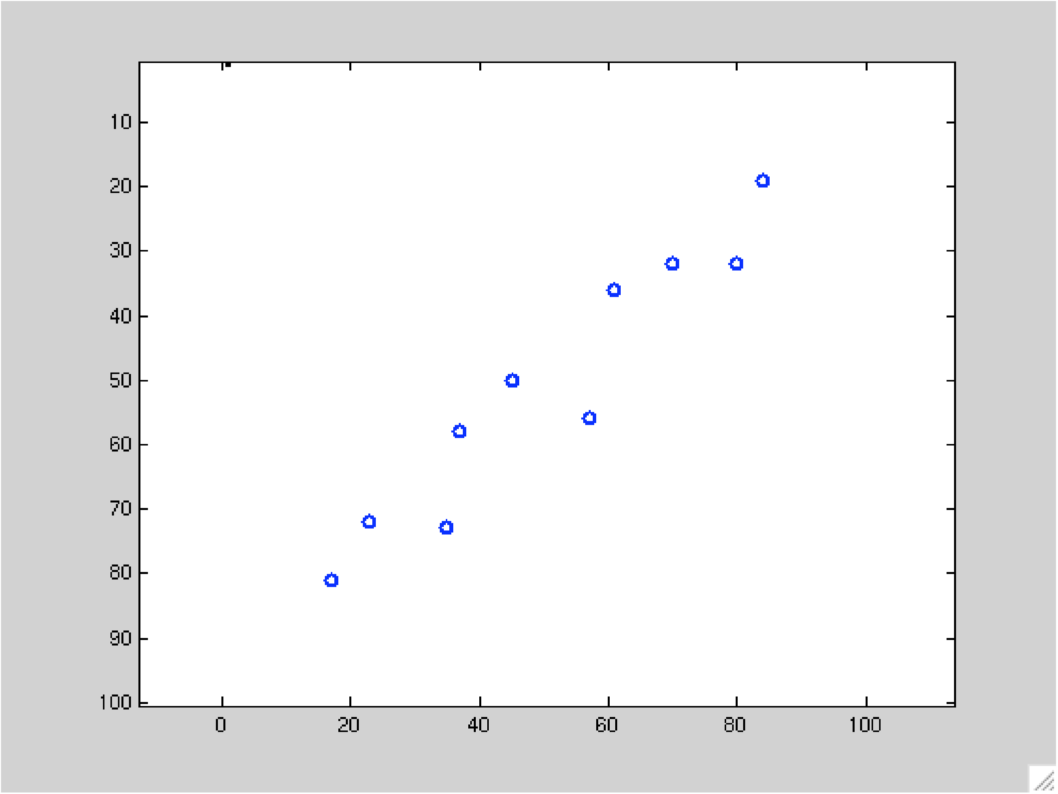
Robert Collins
CSE486, Penn State

Point Feature Data

Point features = $\{(x_i,y_i) \mid i = 1,\ldots,n\}$

$$
\text{pts} = [\ldots
\begin{array}{cc}
17 & 81; \\
23 & 72; \\
35 & 73; \\
37 & 58; \\
45 & 50; \\
57 & 56; \\
61 & 36; \\
70 & 32; \\
80 & 32; \\
84 & 19]
\end{array}
$$

x y



Robert Collins
CSE486, Penn State

Example : Line Fitting

General Idea:

- **Want to fit a model to raw image features (data)**
  (the features could be points, edges, even regions)
- **Parameterize model such that**
  model instance is an element of R^n
  i.e. model instance = $(a_1,a_2,\ldots,a_n)$
- **Define an error function $E(\text{model}_i, \text{data})$ that measures how well a given model instance describes the data**
- **Solve for the model instance that minimizes E**

Line Parameterization

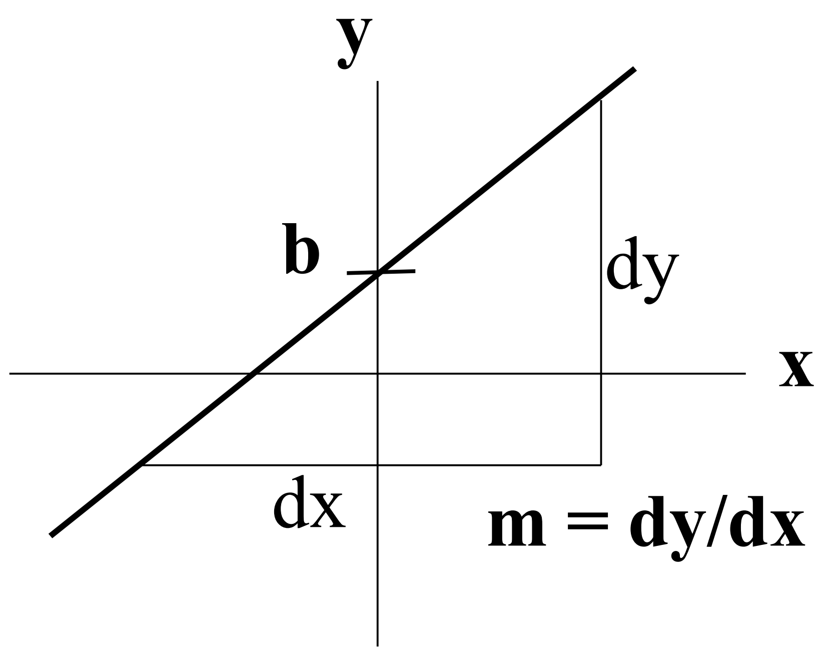
$y = m * x + b$

m = slope

b = y-intercept

(where b crosses the y axis)

Model instance = (m,b)



(The astute student will note a problem with representing vertical lines)

Example : Line Fitting

General Idea:

- **Want to fit a model to raw image features (data)**
  (the features could be points, edges, even regions)
- **Parameterize model such that**
  model instance is an element of R^n
  i.e. model instance = $(a_1, a_2, \ldots, a_n)$
- **Define an error function E($model_i$, data) that measures how well a given model instance describes the data**
- **Solve for the model instance that minimizes E**

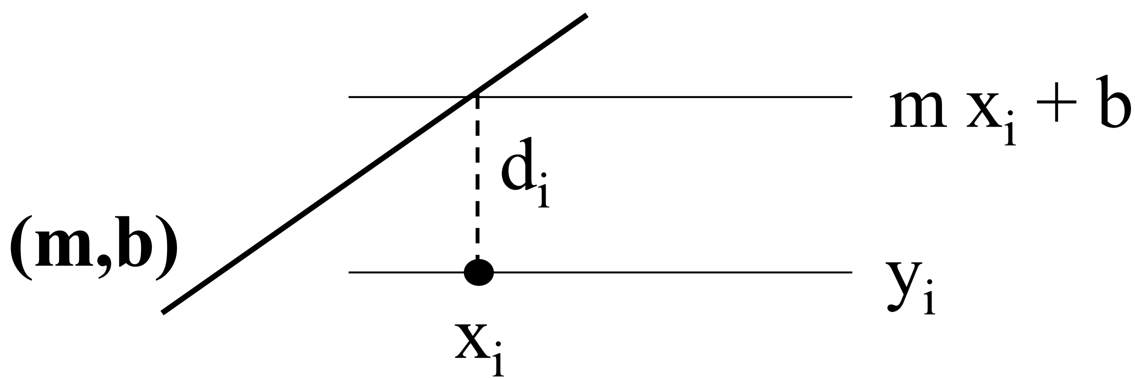
Least Squares (L.S. is just one type of error function)

- **Given line (m,b)**
- **distance of point (x_i,y_i) to line is vertical distance**

$$d_i = ((mx_i + b) - y_i)$$

- **E is sum of squared distances over all points**

$$E = \sum_{i=1}^{n} d_i^2 = \sum_{i=1}^{n} ((mx_i + b) - y_i))^2$$



Robert Collins
CSE486, Penn State

Example : Line Fitting

General Idea:

- **Want to fit a model to raw image features (data)**
  (the features could be points, edges, even regions)
- **Parameterize model such that**
  model instance is an element of R^n
  i.e. model instance = $(a_1, a_2, \ldots, a_n)$
- **Define an error function $E(\text{model}_i, \text{data})$ that measures how well a given model instance describes the data**
- **Solve for the model instance that minimizes E**

Calculus to Find Extrema

- Take first derivatives of E with respect to m, b
- Set equations to zero

$$\frac{d}{dm}E(m,b) = 0$$
$$\frac{d}{db}E(m,b) = 0$$

- Solve for m, b

Note equivalence to linear regression

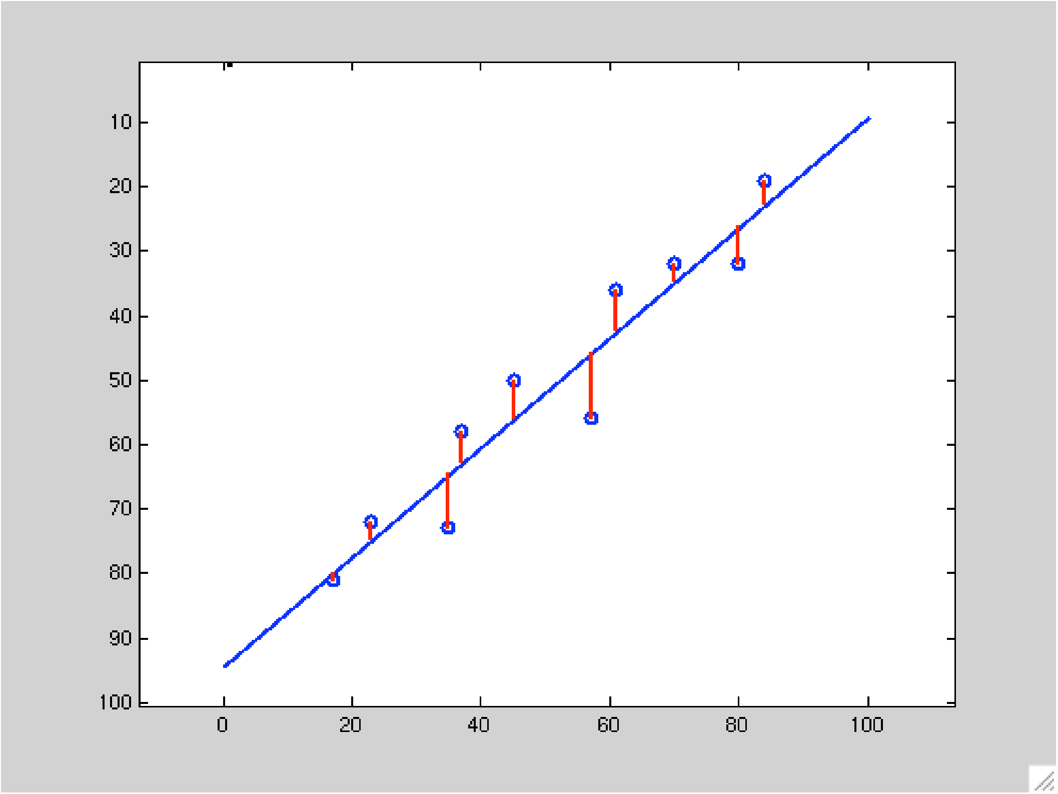
DOING DERIVATION ON THE BOARD

Robert Collins
CSE486, Penn State

Least Squares Solution

$$\begin{bmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & \sum 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} \sum x_i y_i \\ \sum y_i \end{bmatrix}$$

m = -0.8528
b = 94.3059



Problem with Parameterization

- There is a problem with our parameterization, namely (m,b) is undefined for vertical lines

- More general line parameterization

$$ax + by + c = 0$$

such that

$$a^2 + b^2 + c^2 = 1$$

Self-Study!

- Question for class: why do we need the quadratic constraint?
- Another question: what is relation of this to y=mx+b?

LS with Algebraic Distance

Self-Study!

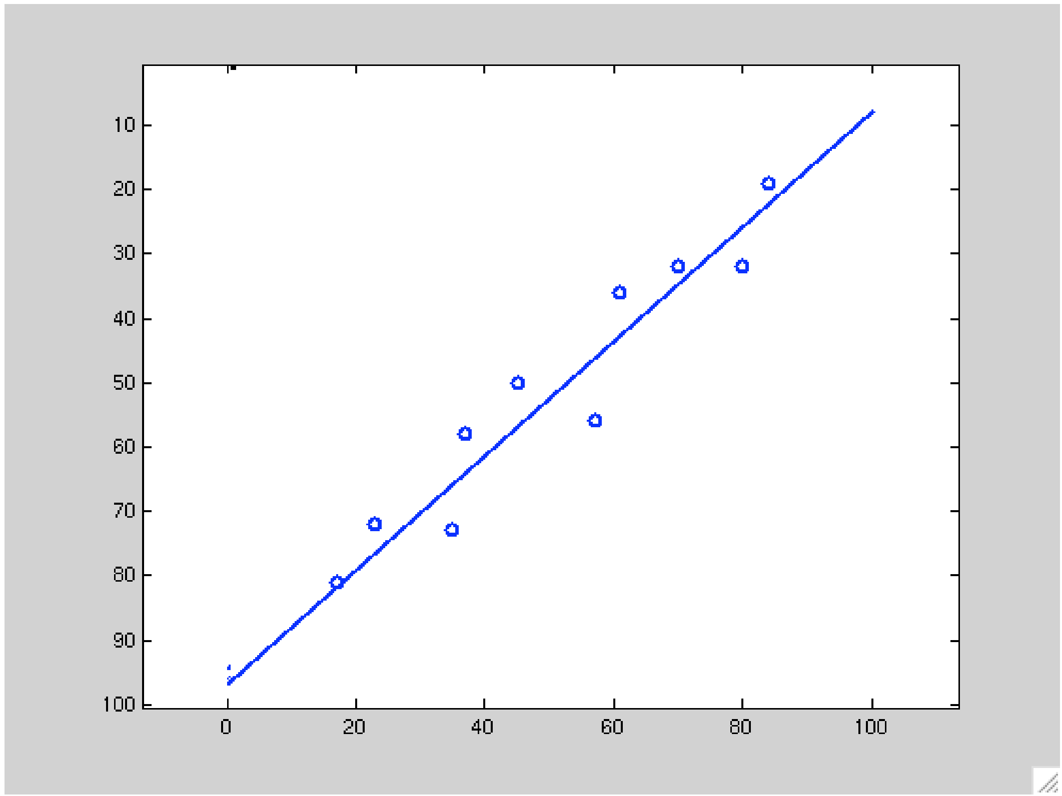
Algebraic Distance $d_i = (ax_i + by_i + c)$

Will derive on board.
Result:

$$\begin{bmatrix} \sum x_i^2 & \sum x_i y_i & \sum x_i \\ \sum x_i y_i & \sum y_i^2 & \sum y_i \\ \sum x_i & \sum y_i & \sum 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0$$

(a,b,c) is eigenvector associated with smallest eigenvalue (it will only be 0 if there is no noise, and therefore all the points lie exactly on a line)

LS with Algebraic Distance



Note much different from linear regression line (in this case) but we can rest assured that our program won't blow up when it sees a vertical line!

Algebraic Least Squares Issues

Self-Study!

- Note: I didn’t draw the error vectors on the plot
- That’s because I don’t know what to draw…
- Main problem with algebraic distances: Hard to say precisely what it is you are minimizing, since algebraic distances are rarely physically meaningful quantities

Orthogonal Least Squares

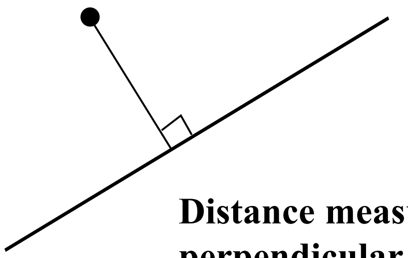
Self-Study!

- Minimize orthogonal (geometric) distance.
- Makes sense physically, but harder to derive
- Representation:

$$ax + by + c = 0$$

such that

$$a^2 + b^2 = 1$$



(compare with algebraic distance)

Orthogonal Least Squares

Self-Study!

Harder to derive.

Key insight: best fit line must pass through the center of mass of the set of points!

Move center of mass to the origin.

This reduces the problem to finding a unit vector normal to the line: (a,b) s.t. $a^2+b^2=1$

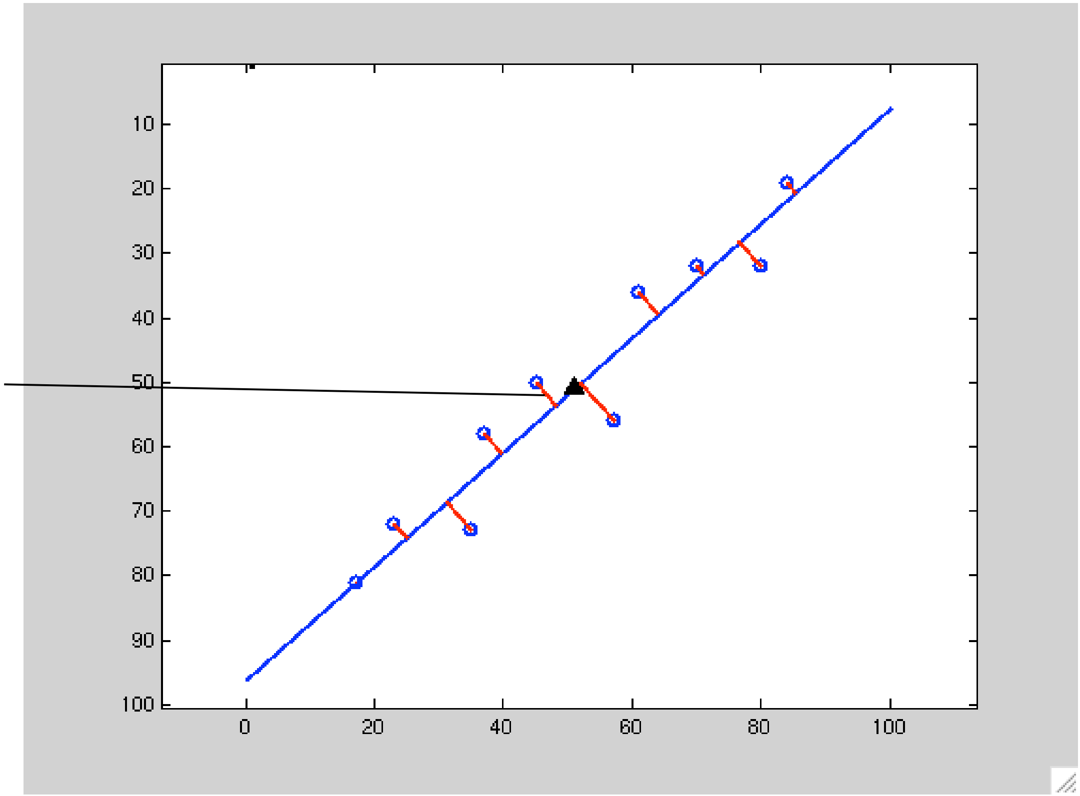
This will be minimum eigenvector of scatter matrix of the points.

Finally, solve for c

Orthogonal Least Squares Solution

Self-Study!

center
of mass



"distance" means what we intuitively expect (i.e. distance to closest point on line, or minimum distance to line)

Parameter Estimation: Estimating a Transformation

Let's say we have found point matches between two images, and we think they are related by some parametric transformation (e.g. translation; scaled Euclidean; affine). How do we estimate the parameters of that transformation?

General Strategy

- Least-Squares estimation from point correspondences

Two important (related) questions:

- How many degrees of freedom?
- How many point correspondences are needed?

Example: Translation Estimation

equations

$$x' = x + t_x$$
$$y' = y + t_y$$

matrix form

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

How many degrees of freedom?

How many independent variables are there? **Two**

How many point correspondences are needed?

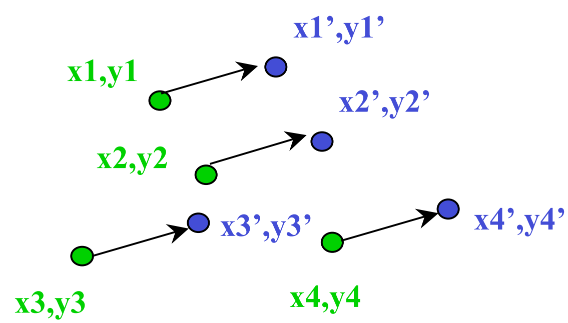
Each correspondence (x,y)=>(x',y') provides two equations

$$\frac{\textbf{DoF}}{\textbf{2}} = \mathbf{2/2 = 1}$$

Example: Translation Estimation

equations

$$x' = x + t_x$$
$$y' = y + t_y$$



Least Squares Estimation:

Minimize $E = \sum_{i=1}^{n} \left((x_i + t_x - x_i')^2 + (y_i + t_y - y_i')^2 \right)$ **wrt t_x, t_y**

$$\frac{\partial E}{\partial t_x} = \sum_{i=1}^{n} 2(x_i + t_x - x_i') = 0 \qquad t_x = \sum_{i=1}^{n} (x_i' - x_i)/n$$

$$\frac{\partial E}{\partial t_y} = \sum_{i=1}^{n} 2(y_i + t_y - y_i') = 0 \qquad t_y = \sum_{i=1}^{n} (y_i' - y_i)/n$$

Let's try another example

Similarity transformation

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

note: $a = s\cos\theta$
$b = s\sin\theta$

ON THE BOARD

Practical Issue

Once we have estimated a transformation, how can we (un)warp image pixel values to produce a new picture.

Warping & Bilinear Interpolation

Given a transformation between two images, (coordinate systems) we want to "warp" one image into the coordinate system of the other.

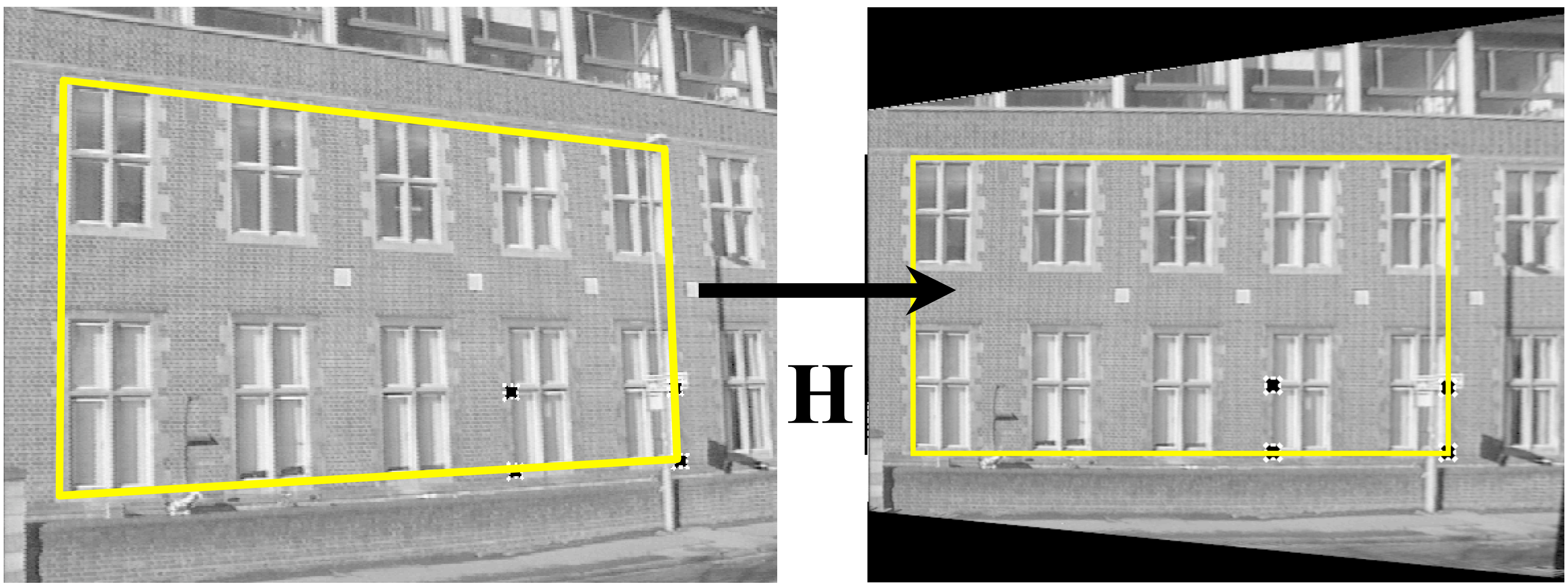
We will call the coordinate system where we are mapping from the "source" image

We will call the coordinate system we are mapping to the "destination" image.

Robert Collins
CSE486, Penn State

Warping Example

Transformation in this case is a projective transformation (general 3x3 matrix, operating on homogeneous coords)

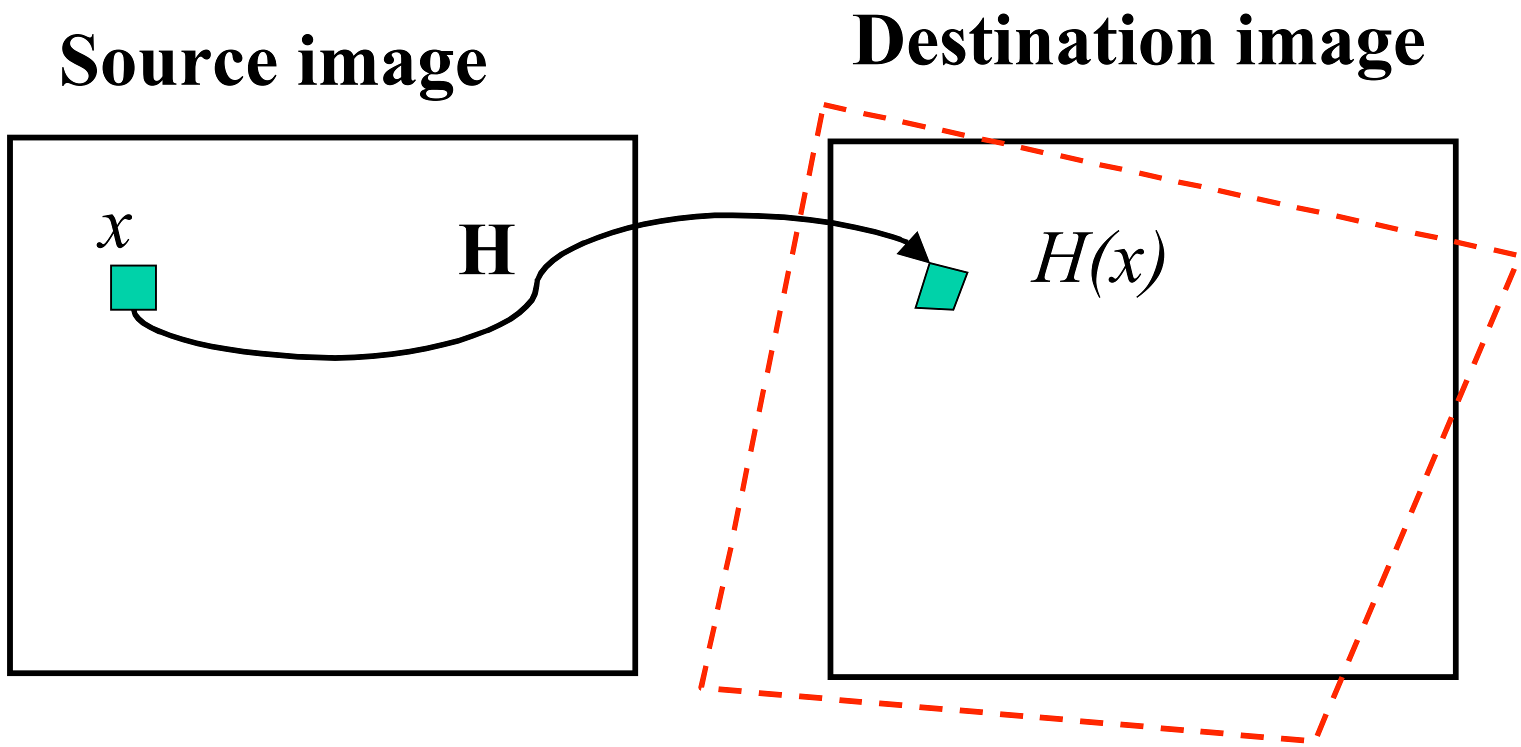


from Hartley & Zisserman

Source Image **Destination image**

We will have a lot more to say about this is a future lecture.

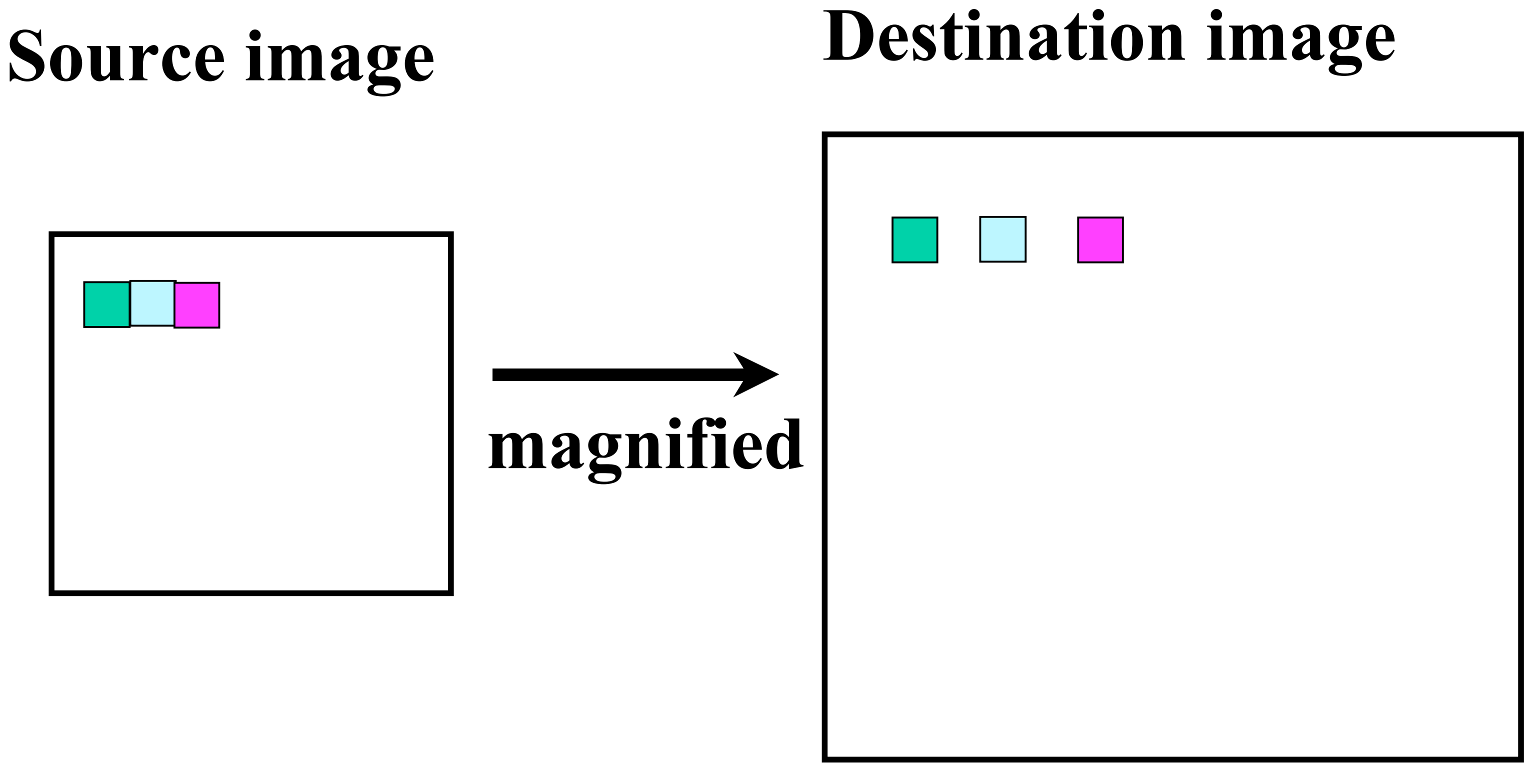
Forward Warping



- For each pixel *x* in the source image
- Determine where it goes as *H(x)*
- Color the destination pixel

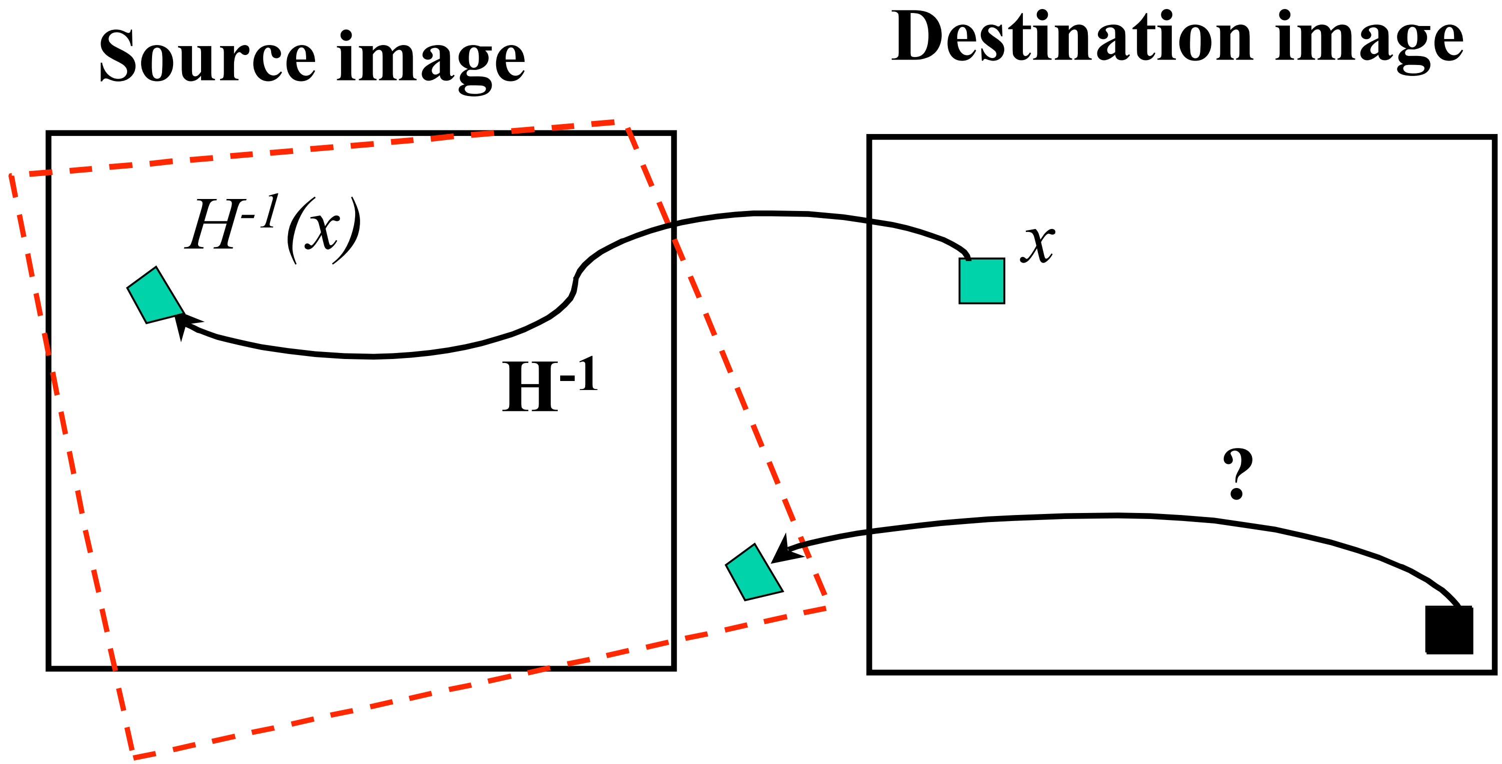
Problems?

Forward Warping Problem



Can leave gaps!

Backward Warping (No gaps)

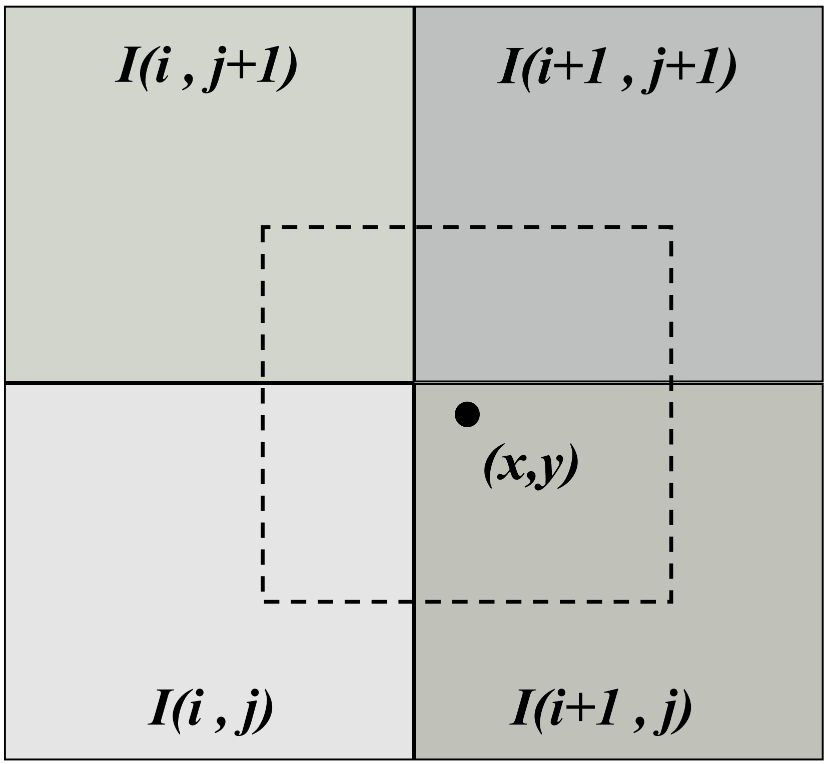


- For each pixel x in the destination image
- Determine where it comes from as $H^{-1}(x)$
- Get color from that location

Interpolation

What do we mean by "get color from that location"?

Consider grey values. What is intensity at (x,y)?



Nearest Neighbor:

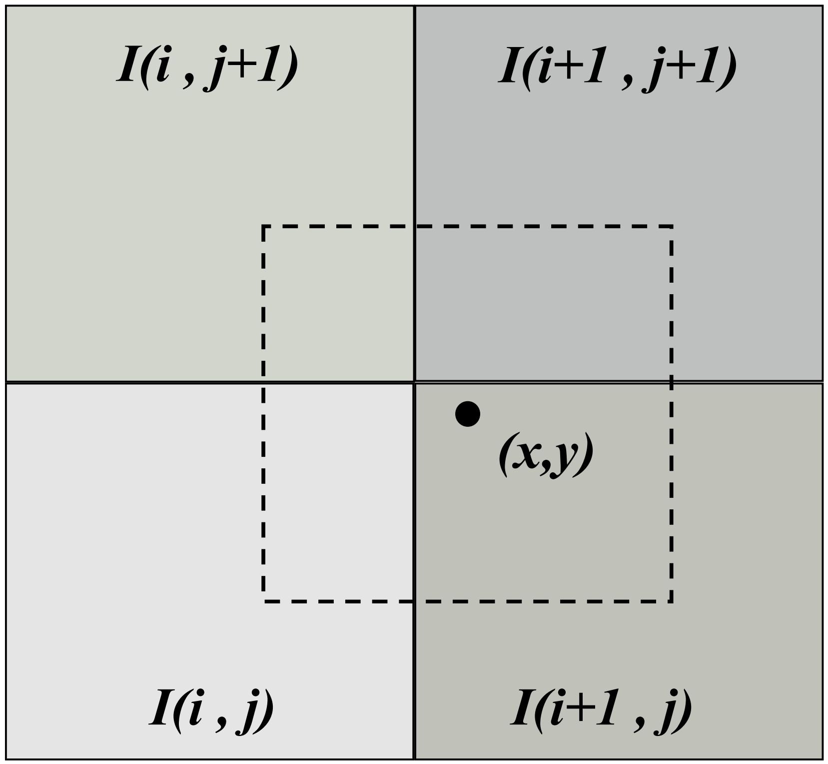
Take color of pixel with closest center.

$$I(x,y) = I(i+1,j)$$

Bilinear interpolation

What do we mean by “get color from that location”?

Consider grey values. What is intensity at (x,y)?



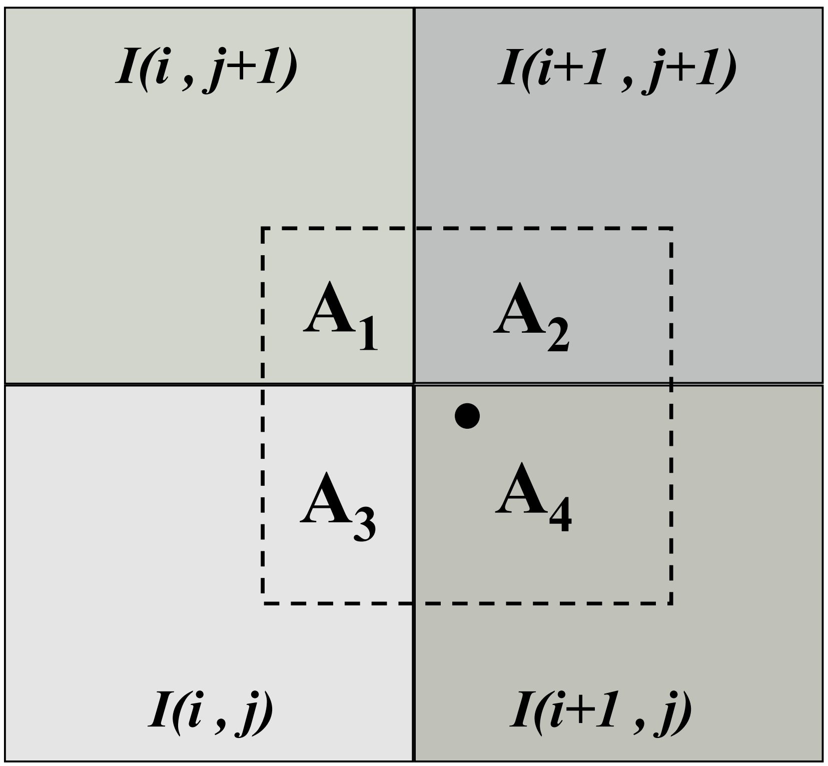
Bilinear Interpolation:

Weighted average

Bilinear interpolation

What do we mean by "get color from that location"?

Consider grey values. What is intensity at (x,y)?



Bilinear Interpolation:

Weighted average

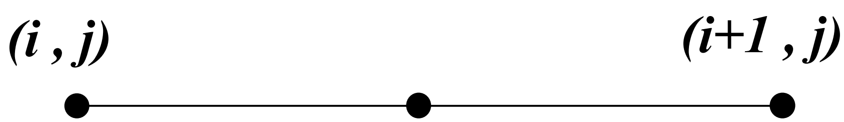
$$\begin{aligned} I(x,y) = {} & A3*I(i,j) \\ & + A4*I(i+1,j) \\ & + A2*I(i+1,j+1) \\ & + A1*I(i,j+1) \end{aligned}$$

Bilinear Interpolation, Math

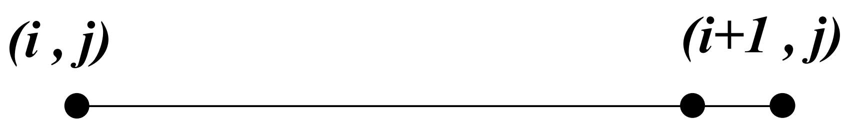
First, consider linear interpolation

Initiution: Given two pixel values, what should the value be at some intermediate point between them?

If close to (i,j), should be intensity similar to I(i,j)

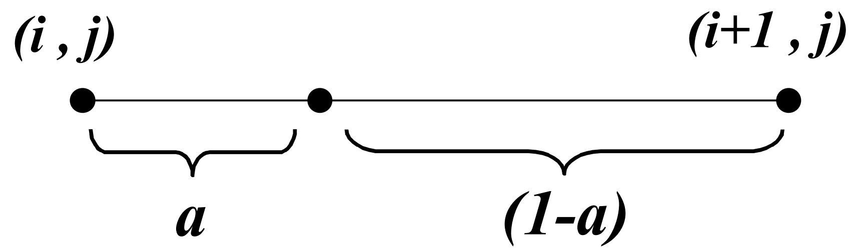


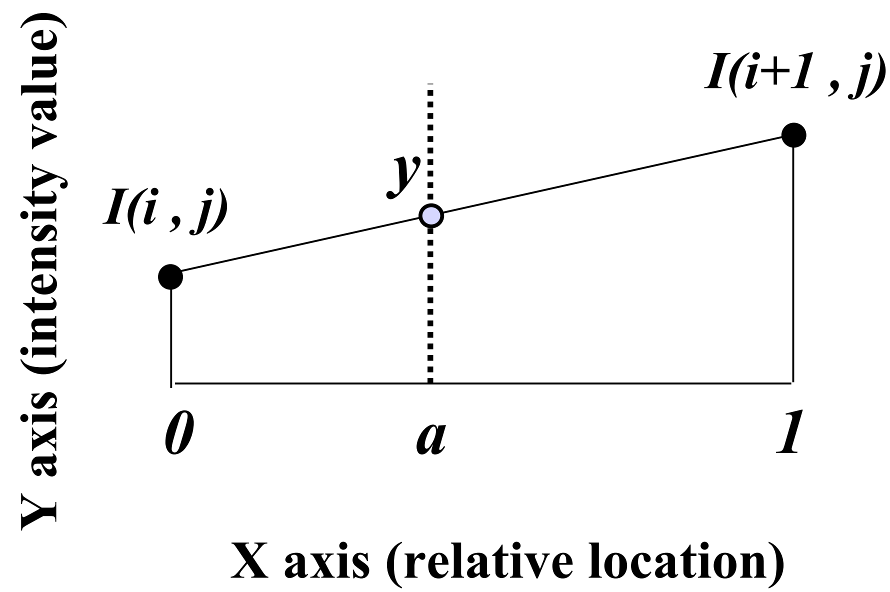
If equidistant from both, should be average of the two intensities



If close to (i+1,j), should be intensity similar to I(i+1,j)

Linear Interpolation





Recall:

$$y - y_1 = \frac{(y_2 - y_1)}{(x_2 - x_1)}(x - x_1)$$

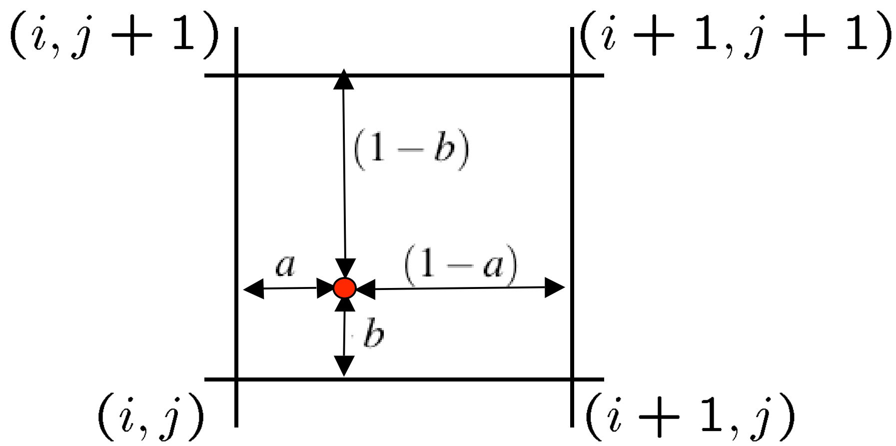
Instantiate

$$y - I(i,j) = \frac{(I(i+1,j) - I(i,j))}{(1-0)}(a-0)$$

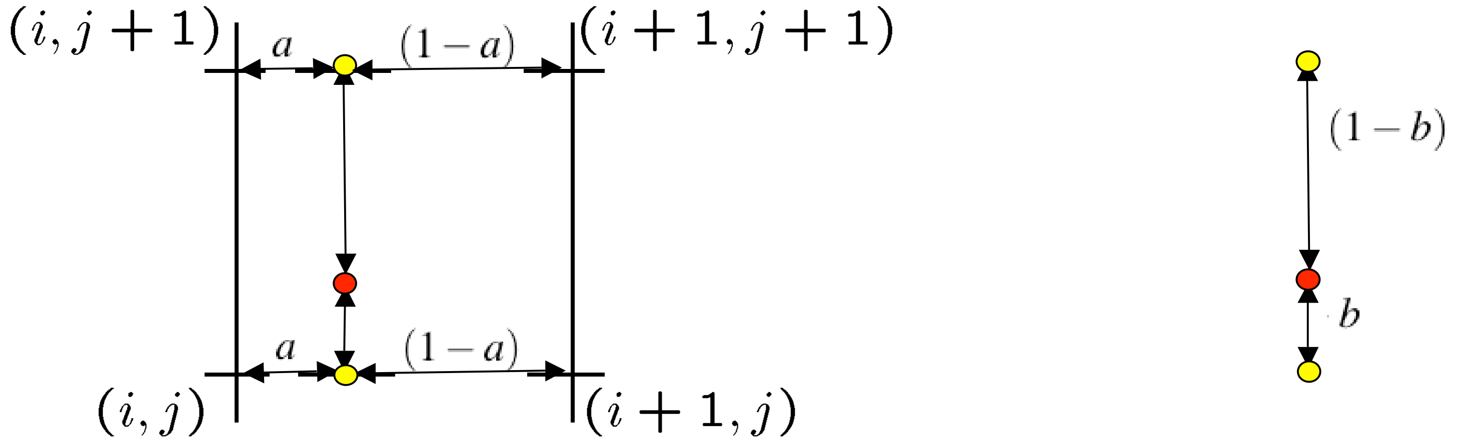
Solve

$$y = (1-a)\, I(i,j) + a\ I(i+1,j)$$

Bilinear Interpolation, Math



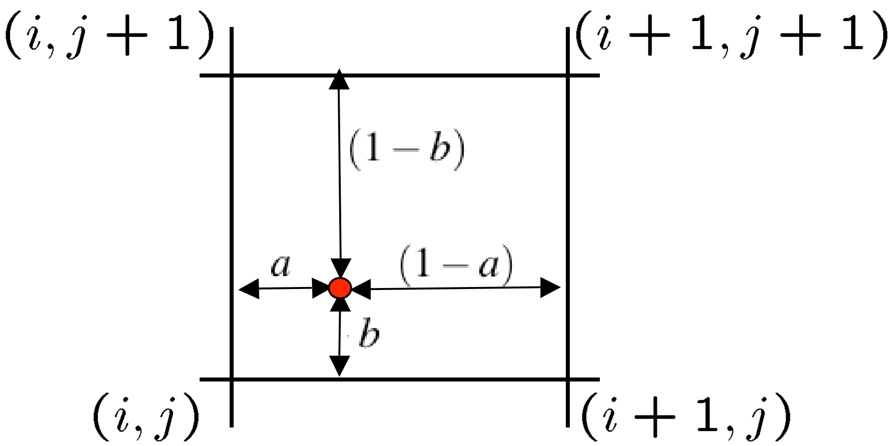
Bilinear Interpolation, Math



$$(1-b)\,[\,(1-a)\,I(i,j) + a\;I(i+1,j)\,]$$

$$+\;\; b\,[\,(1-a)\,I(i,j+1) + a\;I(i+1,j+1)\,]$$

Bilinear Interpolation, Math

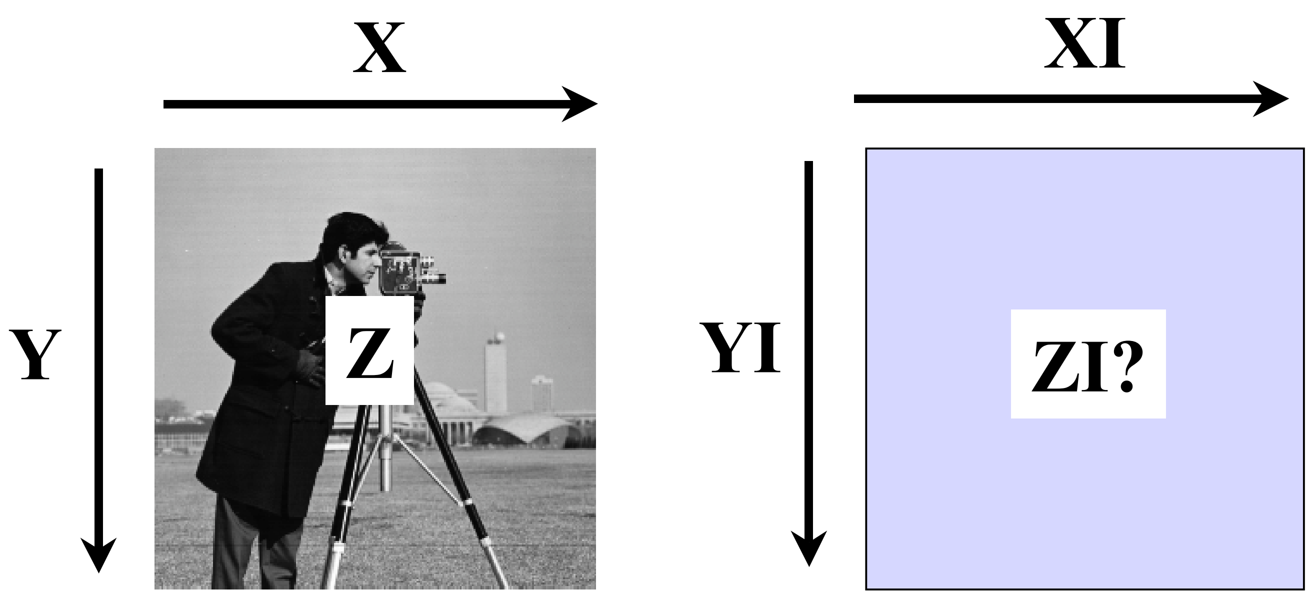


$$
\begin{aligned}
\mathbf{I} = \quad & (1-a)(1-b)\ I(i,j) \\
& + a\ (1-b)\ I(i+1,j) \\
& + (1-a)\ b\ I(i,j+1) \\
& + a\ b\ I(i+1,j+1)
\end{aligned}
$$

Image Warping in Matlab

interp2 is Matlab's built-in function for image warping

Usage: interp2(X,Y,Z,XI,YI)

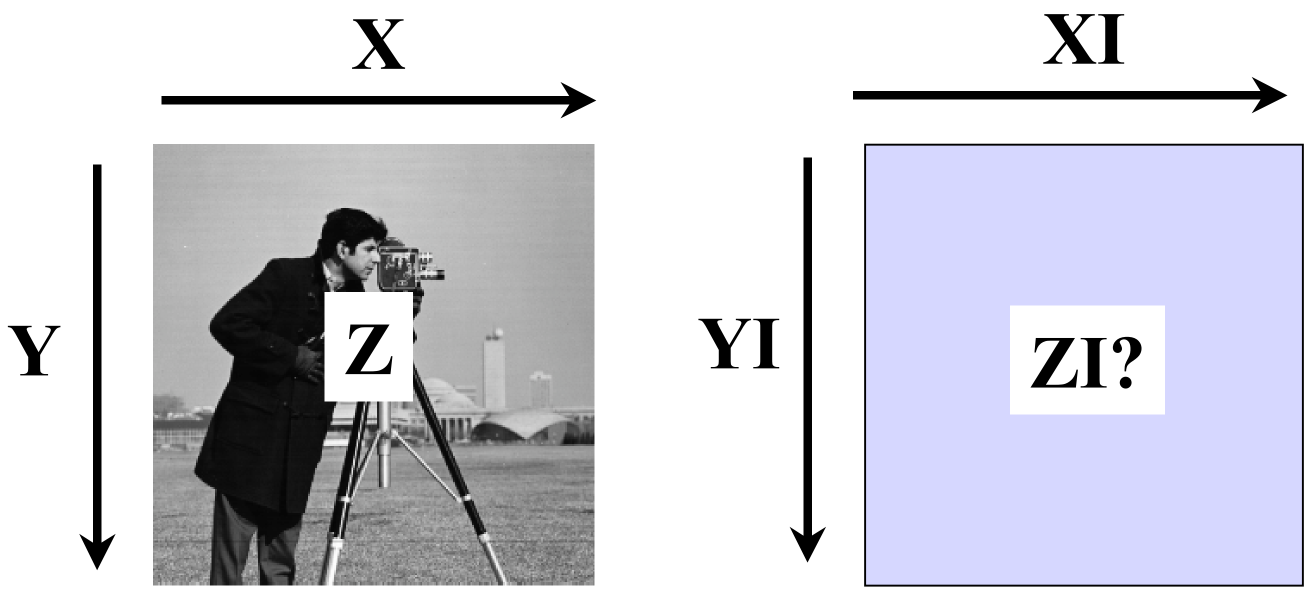


Robert Collins
CSE486, Penn State

Tips on Using Interp2

For our purposes, we can assume X and Y are just normal image pixel coords.

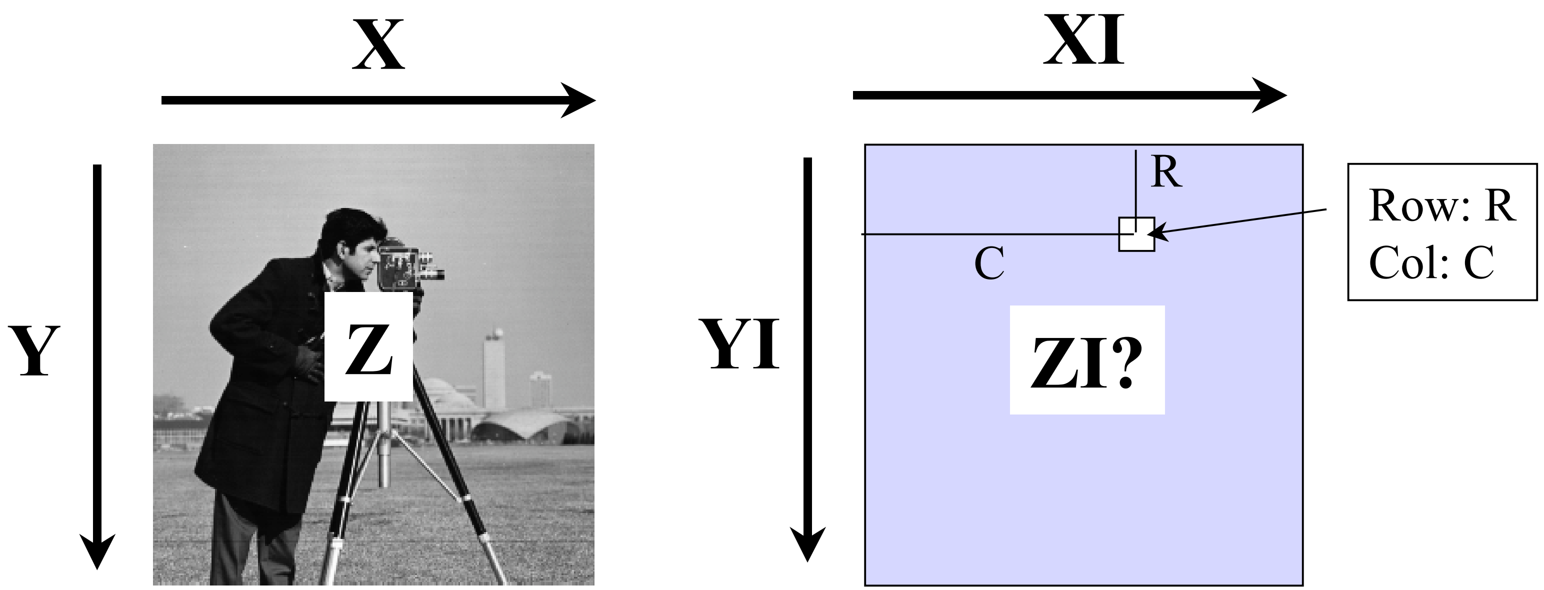
Simpler Usage: interp2(Z,XI,YI)



Tips on Using Interp2 (cont)

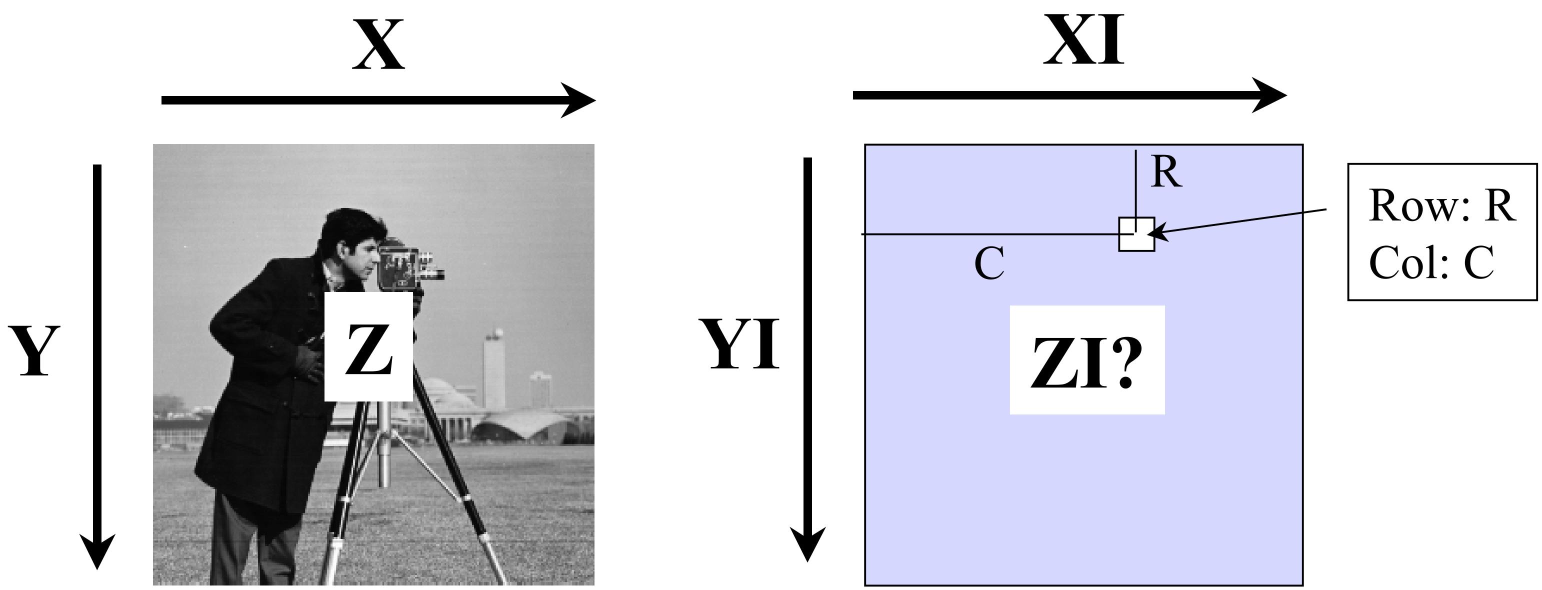
How does it work?

Consider computing the value of ZI at row R and col C.



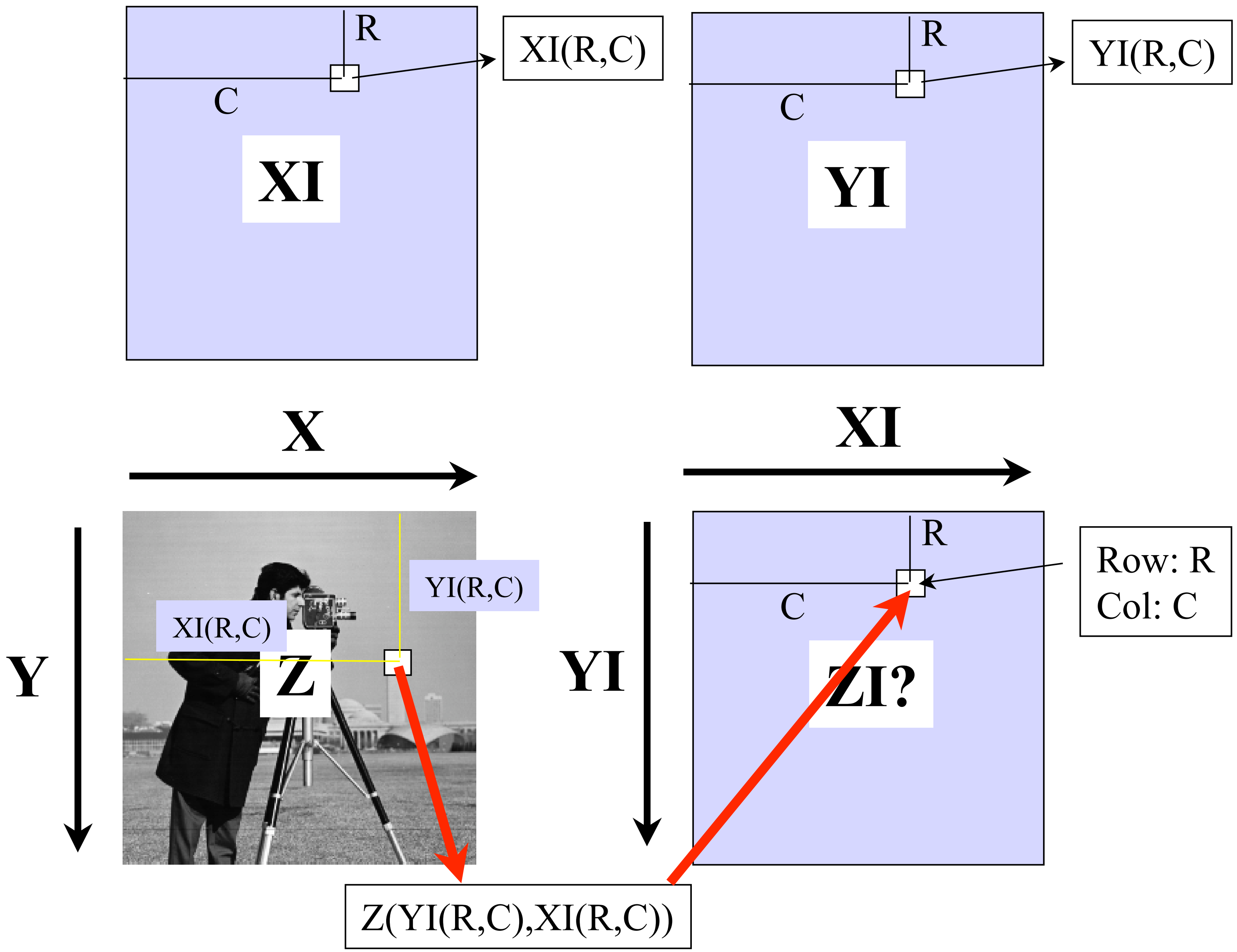
Tips on Using Interp2 (cont)

XI and YI are two arrays the same size as ZI.
For a given row, col (R,C), the value (XI(R,C), YI(R,C)) tells which (X,Y) coord of the orig image to take.
That is: Z(YI(R,C), XI(R,C)).



Robert Collins
CSE486, Penn State

Tips on Using Interp2 (cont)



Tips on Using Interp2 (cont)

interp2 takes care of bilinear interpolation for you, in case YI(R,C) and XI(R,C) are not integer coords.

There are optional arguments to interp2 to change the way the interpolation is calculated. We can go with the default, which is bilinear interp.

Robert Collins
CSE486, Penn State

Meshgrid

A useful function when using interp2 is **meshgrid**

```
>> [xx,yy] = meshgrid(1:4,1:3)

xx =

     1     2     3     4
     1     2     3     4
     1     2     3     4
```

An array suitable for use as XI

```
yy =

     1     1     1     1
     2     2     2     2
     3     3     3     3
```

An array suitable for use as YI

Interp2 Examples

```
im = double(imread('cameraman.tif'));
size(im)
ans =
   256   256

[xi, yi] = meshgrid(1:256, 1:256);

foo = interp2(im, xi, yi);
Imshow(uint8(foo));
```

Result: just a copy of the image.

Robert Collins
CSE486, Penn State

Interp2 Examples

```
im = double(imread('cameraman.tif'));
size(im)
ans =
   256   256

[xi, yi] = meshgrid(1:256, 1:256);

foo = interp2(im, xi/2, yi/2);
Imshow(uint8(foo));
```

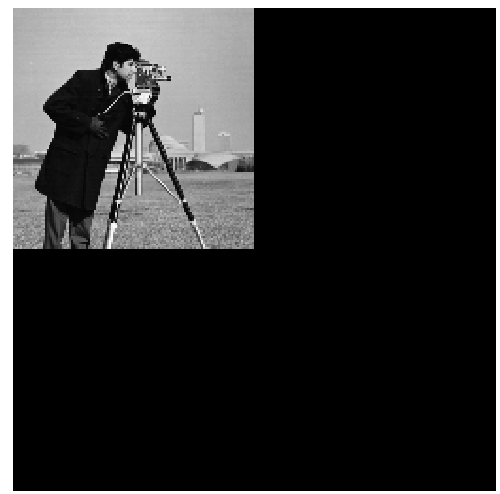
Result: scale up by 2
(but stuck in 256x256 image)

Interp2 Examples

```
im = double(imread('cameraman.tif'));
size(im)
ans =
   256   256

[xi, yi] = meshgrid(1:256, 1:256);

foo = interp2(im, 2*xi, 2*yi);
Imshow(uint8(foo));
```



Result: scale down by 2
(within 256x256 image, pad with 0)

Interp2 Examples

Confusion alert!

```
foo = interp2(im, xi/2, yi/2);
```

Divide coords by 2 to scale up by 2

```
foo = interp2(im, 2*xi, 2*yi);
```

Multiply coords by 2 to reduce up by 2

Interp2 wants the inverse coordinate transforms to the geometric operation you want (it uses backward warping)

Interp2 Examples

A more complicated example: scale down by 2, but around center of image (128,128), not (0,0)

Recall concatenation of transformation matrices:

$$H = \underbrace{\begin{bmatrix} 1 & 0 & 128 \\ 0 & 1 & 128 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Bring origin back to (128,128)}} \underbrace{\begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Scale down by 2 around origin}} \underbrace{\begin{bmatrix} 1 & 0 & -128 \\ 0 & 1 & -128 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Bring (128,128) to origin}}$$

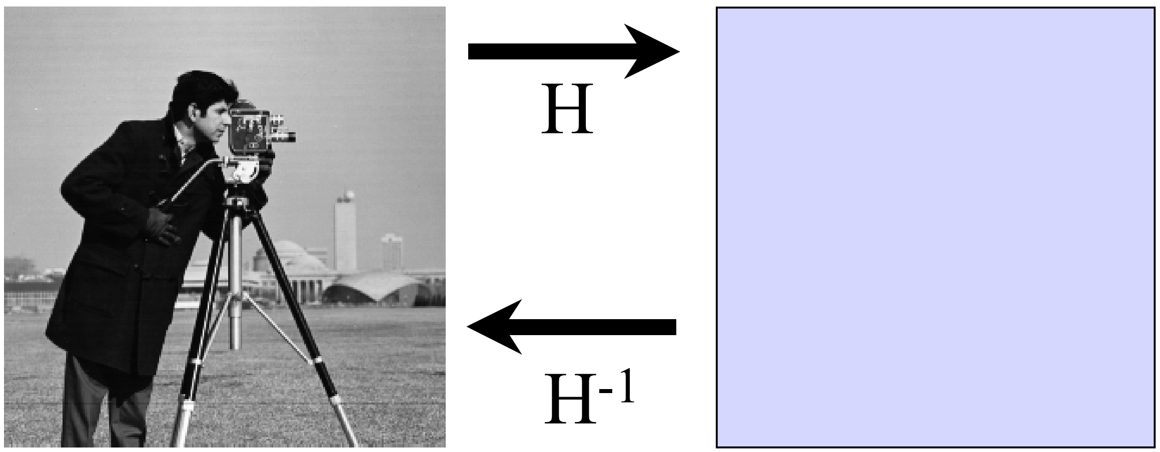
$$H = \begin{bmatrix} .5 & 0 & 64 \\ 0 & .5 & 64 \\ 0 & 0 & 1 \end{bmatrix}$$

BUT, BE CAREFUL….

Interp2 Examples

$$H = \begin{bmatrix} .5 & 0 & 64 \\ 0 & .5 & 64 \\ 0 & 0 & 1 \end{bmatrix}$$

This specifies how we want the original image to map into the new image.



Interp2 wants to know how new image coords map back to the original image coords.

$$H^{-1} = \begin{bmatrix} 2 & 0 & -128 \\ 0 & 2 & -128 \\ 0 & 0 & 1 \end{bmatrix}$$

Interp2 Examples

```
im = double(imread('cameraman.tif'));
size(im)
ans =
   256   256

[xi, yi] = meshgrid(1:256, 1:256);

foo = interp2(im, 2*xi-128, 2*yi-128);
imshow(uint8(foo));
```

$$H = \begin{bmatrix} 2 & 0 & -128 \\ 0 & 2 & -128 \\ 0 & 0 & 1 \end{bmatrix}$$

Result

Robert Collins
CSE486, Penn State

Interp2 Examples

More generally
(e.g. for any 3x3 transformation
matrix in homogeneous coords)

$$H = \begin{bmatrix} 1 & 0 & 128 \\ 0 & 1 & 128 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -128 \\ 0 & 1 & -128 \\ 0 & 0 & 1 \end{bmatrix}$$

$$H = H^{-1}$$

```
im = double(imread('cameraman.tif'));
size(im)
ans =
   256   256

[xi, yi] = meshgrid(1:256, 1:256);

h = [1 0 128;0 1 128; 0 0 1] * [1/2 0 0; 0 1/2 0; 0 0 1] * [1 0 -128; 0 1 -128; 0 0 1];
h = inv(h);   %TAKE INVERSE FOR USE WITH INTERP2
xx = (h(1,1)*xi+h(1,2)*yi+h(1,3))./(h(3,1)*xi+h(3,2)*yi+h(3,3));
yy = (h(2,1)*xi+h(2,2)*yi+h(2,3))./(h(3,1)*xi+h(3,2)*yi+h(3,3));
foo = uint8(interp2(im,xx,yy));
figure(1); imshow(foo)
```

Result