

Yin, MST Spring 2012	
Teams	
Teams	Projects
Stewart Boling, Nicolas Pereira	
Michael Wisely, Mat Nuckolls	
Nate Eloie, Jacob Gardener	
Mingzhong Li, Wenyong Lu	
Catlin Derr, Chester Gregg	
Jason Hagerty, Steven Heitzman	
Cheng Lu, Nabin Mishra	
Arturo Rosas, Erik Oneal	
Tom Reese (AUD)	

Yin, MST  
Spring 2012

## Lecture 08

### Project 1: Computing Homography Matrix Warping Images and Image Mosaic

Yin, MST  
Spring 2012

### Computing Homography Matrix

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \sim \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Source  
 $p=(x,y)$

Destination  
 $p'=(x',y')$

Manually select correspondence points:

```

source = imread('sourceImg.jpg');
figure(1); imshow(source,[]);
[xpts,ypts] = ginput;
hold on;
plot(xpts, ypts, 'rs','Markersize',12);
text(xpts, ypts, num2str((1:length(xpts))),'Color','r')
hold off;

```

Yin, MST  
Spring 2012

### Computing Homography Matrix

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \sim \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Source  
 $p=(x,y)$

Destination  
 $p'=(x',y')$

**Your tasks: compute the Homography matrix using two methods:**

- Set  $h_{33}=1$
- Set  $\|h\| = 1$

Yin, MST  
Spring 2012

### Computing Homography Matrix

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \sim \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$h_{33}=1$			$\ h\ =1$		
2N x 8	8 x 1	2N x 1	2N x 9	9 x 1	2N x 1
Point 1 $\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1 y'_1 & -y_1 y'_1 \end{bmatrix}$ Point 2 $\begin{bmatrix} x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2 y'_2 & -y_2 y'_2 \end{bmatrix}$ Point 3 $\begin{bmatrix} x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3 y'_3 & -y_3 y'_3 \end{bmatrix}$ Point 4 $\begin{bmatrix} x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4 y'_4 & -y_4 y'_4 \end{bmatrix}$ additional points $\vdots$	$\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix}$	$\begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{bmatrix}$	4 x o n s	$\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$
Linear equations	$2N \times 8 \quad 8 \times 1 \quad 2N \times 1$		Homogeneous equations	$2N \times 9 \quad 9 \times 1 \quad 2N \times 1$	
Matlab:	$h = A \setminus b$		SVD of $A^T A = U \quad D \quad V^T$		
			Let $h$ be the column of $V$ associated with the smallest singular value in $D$ .		

Yin, MST  
Spring 2012

### Forward Warping

Source image

Destination image

$p' = H(p)$

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

Yin, MST  
Spring 2012

## Backward Warping (No gaps)

7

Source image

Destination image

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

Yin, MST  
Spring 2012

## Example: Forward Warping Using “for-loops”

8

```

[nrows, ncols] = size(imDest);
warpedSrc = zeros(nrows, ncols);
for x = 1:ncols
    for y = 1:nrows
        p = [x; y; 1];
        pprime = H*p;
        xprime = round(pprime(1)/pprime(3));
        yprime = round(pprime(2)/pprime(3));
        if xprime < 1 | xprime > ncols | ...
            yprime < 1 | yprime > nrows
            continue;
        end
        warpedSrc(xprime,yprime)=imSrc(y,x);
    end
end
figure; imshow(warpedSrc,[]);

```

Source image

Destination image

**Your pseudo task:**  
implement the forward warping

Yin, MST  
Spring 2012

## Backward Warping with Interpolation

9

```

[nrows, ncols] = size(imDest);
warpedSrc = zeros(nrows, ncols);
for x = 1:ncols
    for y = 1:nrows
        p = [x; y; 1];
        pprime = H*p;
        xprime = round(pprime(1)/pprime(3));
        yprime = round(pprime(2)/pprime(3));
        if xprime < 1 | xprime > ncols | ...
            yprime < 1 | yprime > nrows
            continue;
        end
        warpedSrc(xprime,yprime)=imSrc(y,x);
    end
end
figure; imshow(warpedSrc,[]);

```

Source image

Destination image

**Your tasks:**  
modify the codes as  
backward warping  
and  
try two interpolation  
methods

Yin, MST  
Spring 2012

## Interpolation Methods

10

### 1. Nearest neighbor

Take color of pixel with closest center.

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

### 2. Bilinear

Weighted average

$I = (1-a)(1-b) I(i, j) + a(1-b) I(i+1, j) + (1-a)b I(i, j+1) + ab I(i+1, j+1)$

Yin, MST  
Spring 2012

## Image Warping in Matlab

11

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

Usage:  $ZI = \text{interp2}(X, Y, Z, XI, YI)$

X

XI

Yin, MST  
Spring 2012

## Tips on Using Interp2

12

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

Simpler Usage:  $ZI = \text{interp2}(Z, XI, YI)$

X

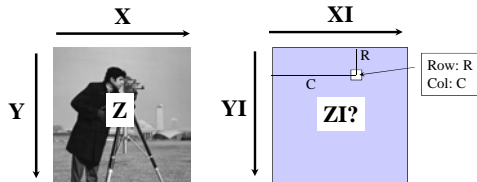
XI

## Tips on Using Interp2 (cont)

13

How does it work?

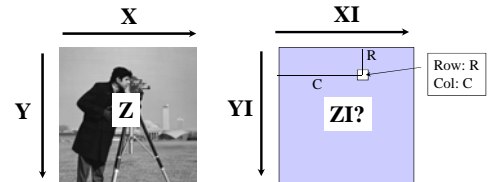
Consider computing the value of ZI at row R and col C.



## Tips on Using Interp2 (cont)

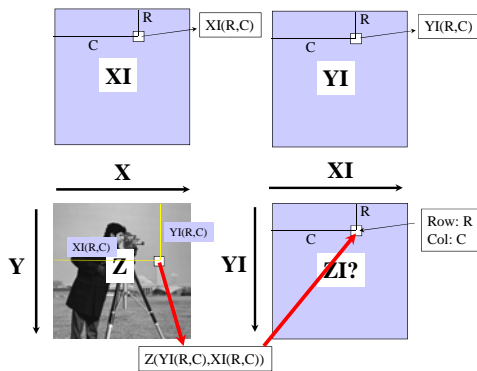
14

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:  $Z(YI(R,C), XI(R,C))$ .



## Tips on Using Interp2 (cont)

15



## Tips on Using Interp2 (cont)

16

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.

## Meshgrid

17

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

18

```
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.

Yin, MST  
Spring 2012


## Interp2 Examples

19

```
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)

Yin, MST  
Spring 2012


## Interp2 Examples

20

```
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)

Yin, MST  
Spring 2012


## Interp2 Examples

21

**Confusion alert!**

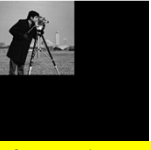
```
foo = interp2(im, xi/2, yi/2);
```

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)

Yin, MST  
Spring 2012

## Interp2 Examples

22

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

Recall concatenation of transformation matrices:

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

Bring (128,128) to origin  
Scale down by 2 around origin  
Bring origin back to (0,0)

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

BUT, BE CAREFUL....

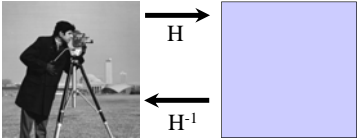
Yin, MST  
Spring 2012

## Interp2 Examples

23

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

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.

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

Yin, MST  
Spring 2012


## Interp2 Examples

24

```
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));
```



Result

Yin, MST  
Spring 2012
25

## 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


Yin, MST  
Spring 2012
26

## Your Pseudo Task: Use Interp2 for Backward Warping

```

im = double(imread('cameraman.tif'));

[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)

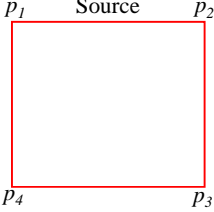
```

**Your Pseudo Tasks:**  
**Modify this code for backward mapping.**  
**Use different interpolation options in interp2() function.**  
**Compare with your for-loop method.**

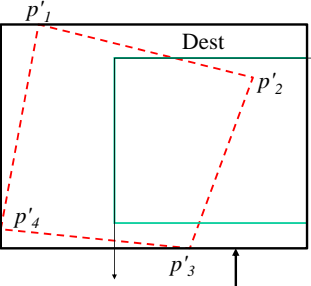
Yin, MST  
Spring 2012
27

## Image Mosaic

Source



Dest



How to define the size and pixel value of this image?

Yin, MST  
Spring 2012
28

## Matlab Demo