Project 1

Digital Image Acquisition

Md Zobaer Islam

PhD student, Oklahoma State University

CWID: A20270547

Explanation of submission files (only .m files in alphabetical order)

File name	What it is---
bicubic_downsize_upsize.m	Upsizing a downsized image using bicubic interpolation and calculating error
bicubic_interp_color.m	Bicubic interpolation code for a colored image
bicubic_interp_gray.m	Bicubic interpolation code for a grayscale (cameraman) image
image_registration_cameramanB_bicubic.m	Image registration of cameramanB image using bicubic interpolation
image_registration_cameramanB_linear.m	Image registration of cameramanB image using linear interpolation
image_registration_cameramanB_zoh.m	Image registration of cameramanB image using ZOH interpolation
image_registration_cameramanC_bicubic.m	Image registration of cameramanC image using bicubic interpolation
image_registration_cameramanC_linear.m	Image registration of cameramanC image using linear interpolation
image_registration_cameramanC_zoh.m	Image registration of cameramanC image using ZOH interpolation
linear_downsize_upsize.m	Upsizing a downsized image using linear interpolation and calculating error
linear_interp_color.m	Linear interpolation code for a colored image
linear_interp_gray.m	Linear interpolation code for a grayscale (cameraman) image

Explanation of submission files (only .m files in alphabetical order)

File name	What it is---
MexHat.m	Mexhat function
MSE_calc_for_imresize.m	For MSE calculation by comparing the original image and the downsized+upsized image using imresize() function
zoh_downsize_upsize.m	Upsizing a downsized image using zoh interpolation and calculating error
zoh_interp_color.m	ZOH interpolation code for a colored image
zoh_interp_gray.m	ZOH interpolation code for a grayscale (cameraman) image

Image Interpolation

Bicubic Image Interpolation in MATLAB

```
clc; clear; close all;
p=4; %zoom ratio
II=imread('cameraman.bmp'); %read the image
I=im2double(II); %Convert to double for floating point operation
[x,y] = size(I);
%New sizes of the image, rounding has been done because p can be fractional
X=round(x*p); Y=round(y*p);
%new pixel locations u in x-direction, v in y-direction
u=1:1/p:((X-1)/p+1); v=1:1/p:((Y-1)/p+1);
[XI,YI]=ndgrid(u,v); %Create the 2D grid of new pixels
UI=XI-floor(XI); VI=YI-floor(YI);
%map the new pixel locations u and v to the original pixel locations
X1=floor(u)-1; X2=floor(u); X3=floor(u)+1; X4=floor(u)+2;
Y1=floor(v)-1; Y2=floor(v); Y3=floor(v)+1; Y4=floor(v)+2;
%Handle the boundary cases
X4(find(X4>x))=x; Y4(find(Y4>y))=y; X3(find(X3>x))=x; Y3(find(Y3>y))=y;
X1(find(X1<1))=1; Y1(find(Y1<1))=1;
```

Continued at the next slide

Bicubic Image Interpolation in MATLAB

```
I1=I(X1,Y1); I2=I(X1,Y2); I3=I(X1,Y3); I4=I(X1,Y4); I5=I(X2,Y1); I6=I(X2,Y2);
I7=I(X2,Y3); I8=I(X2,Y4); I9=I(X3,Y1); I10=I(X3,Y2); I11=I(X3,Y3);
I12=I(X3,Y4); I13=I(X4,Y1); I14=I(X4,Y2); I15=I(X4,Y3); I16=I(X4,Y4);
%Distances (4 distances in row and column directions each)
Xa = UI+1; Xb = UI; Xc = 1-UI; Xd = 2-UI; Ya = VI+1; Yb = VI; Yc = 1-VI; Yd = 2-VI;
%Distances will be passed to mexhat function to have weights
Xamex = arrayfun(@MexHat,Xa); Yamex = arrayfun(@MexHat,Ya);
Xbmex = arrayfun(@MexHat,Xb); Ybmex = arrayfun(@MexHat,Yb);
Xcmex = arrayfun(@MexHat,Xc); Ycmex = arrayfun(@MexHat,Yc);
Xdmex = arrayfun(@MexHat,Xd); Ydmex = arrayfun(@MexHat,Yd);
%Weights calculation
c1 = Xamex.*Yamex; c2 = Xamex.*Ybmex; c3 = Xamex.*Ycmex; c4 = Xamex.*Ydmex;
c5 = Xbmex.*Yamex; c6 = Xbmex.*Ybmex; c7 = Xbmex.*Ycmex; c8 = Xbmex.*Ydmex;
c9 = Xcmex.*Yamex; c10 = Xcmex.*Ybmex; c11 = Xcmex.*Ycmex; c12 = Xcmex.*Ydmex;
c13 = Xdmex.*Yamex; c14 = Xdmex.*Ybmex; c15 = Xdmex.*Ycmex; c16 = Xdmex.*Ydmex;
B=c1.*I1 + c2.*I2 + c3.*I3 + c4.*I4 + c5.*I5 + c6.*I6 + c7.*I7 + c8.*I8 ...
+c9.*I9 + c10.*I10 + c11.*I11 + c12.*I12 + c13.*I13 + c14.*I14 + c15.*I15 + c16.*I16;
figure(1);imshow(B);
```

Comparison of Interpolated Images

P=4

ZOH

Linear

Bicubic

Comparison of Interpolated Images (Zoomed-in)

P=4

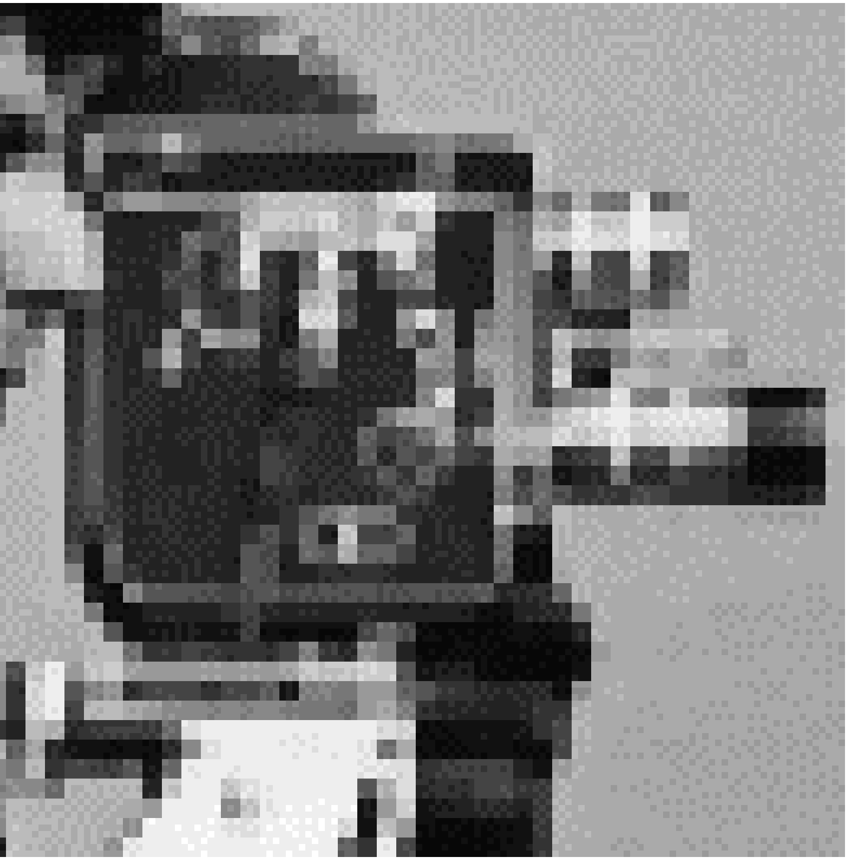
ZOH | Linear | Bicubic

Highly pixelated | Smoother | Sharper

Comparison of Interpolated Images (Zoomed-in)

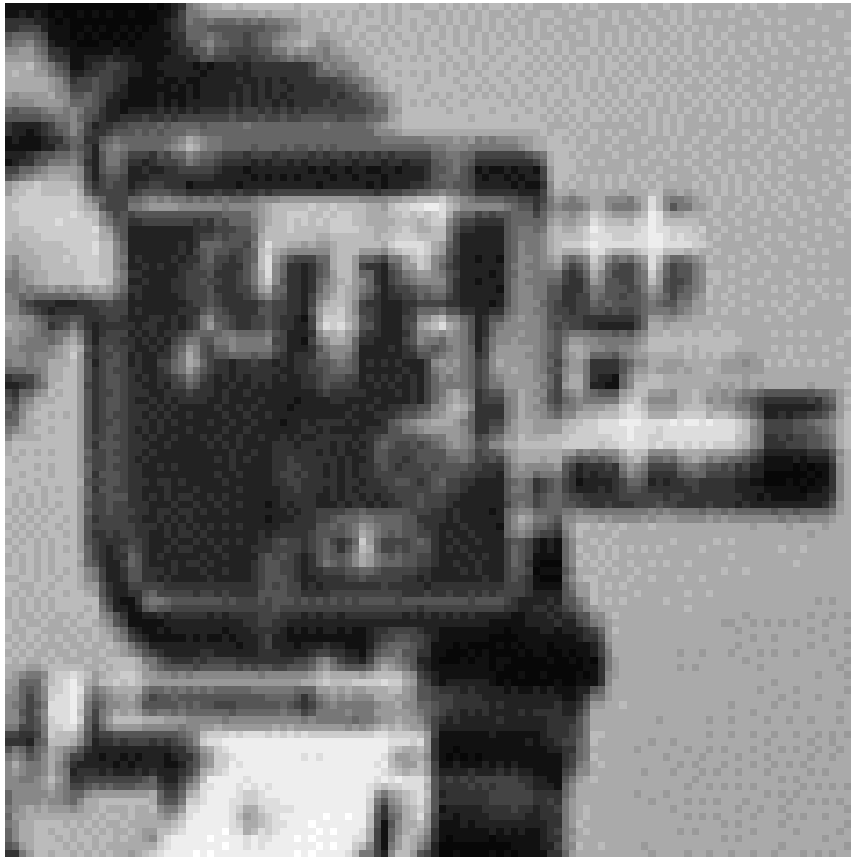
P=4

ZOH



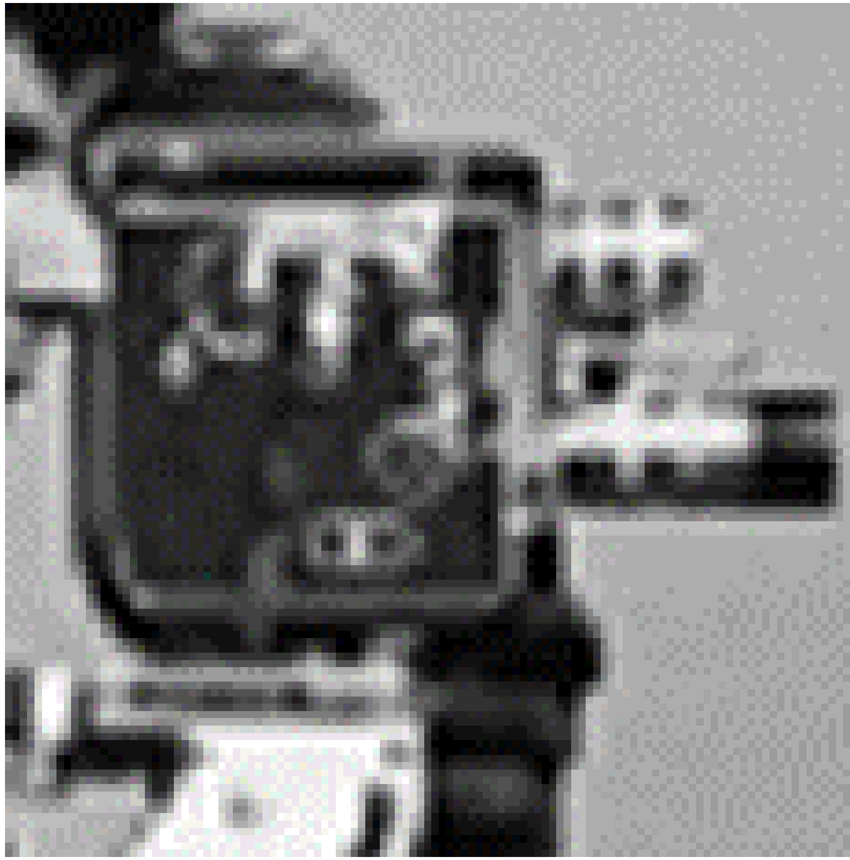
Highly pixelated

Linear



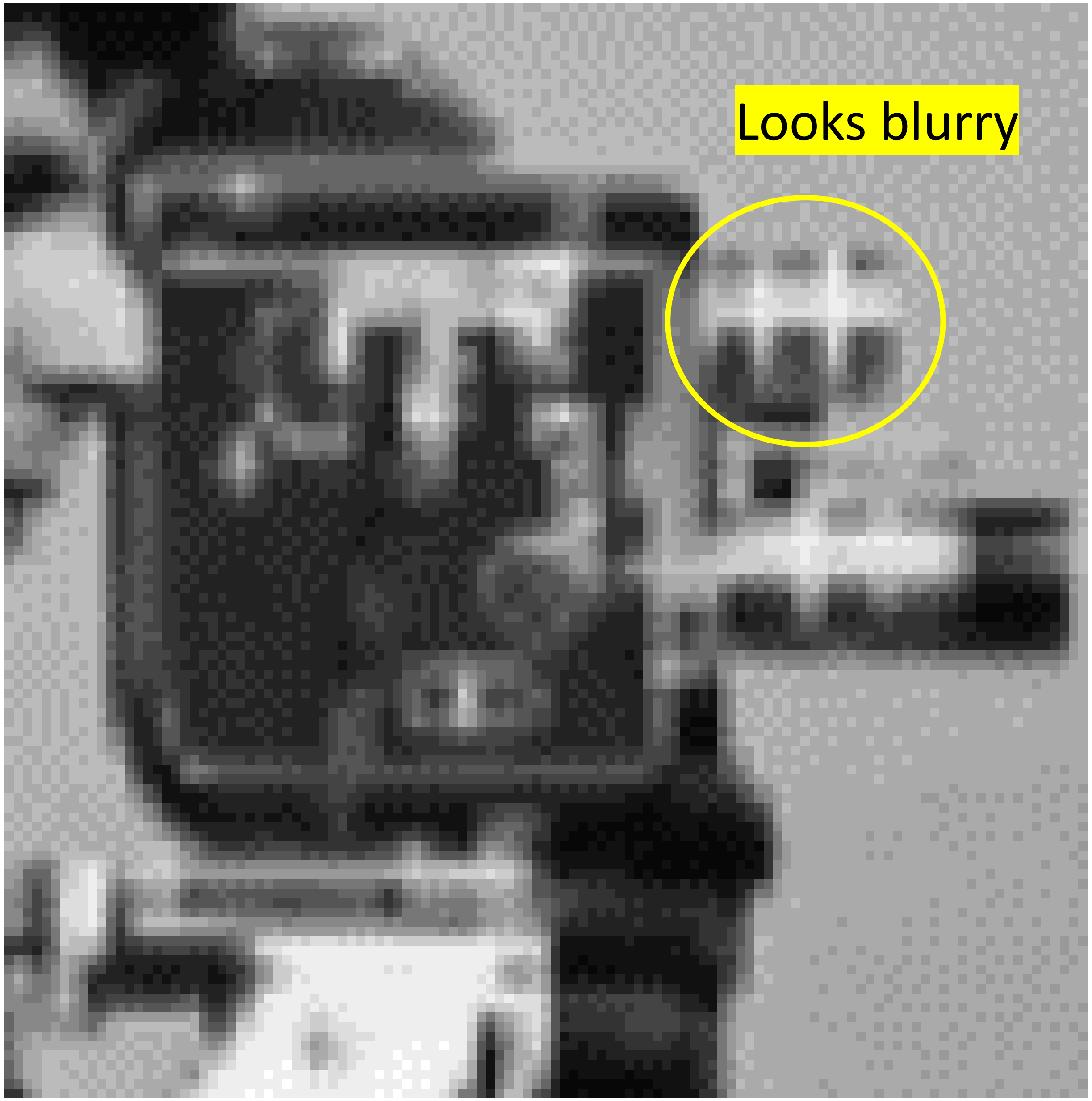
Smoother, but blurry

Bicubic

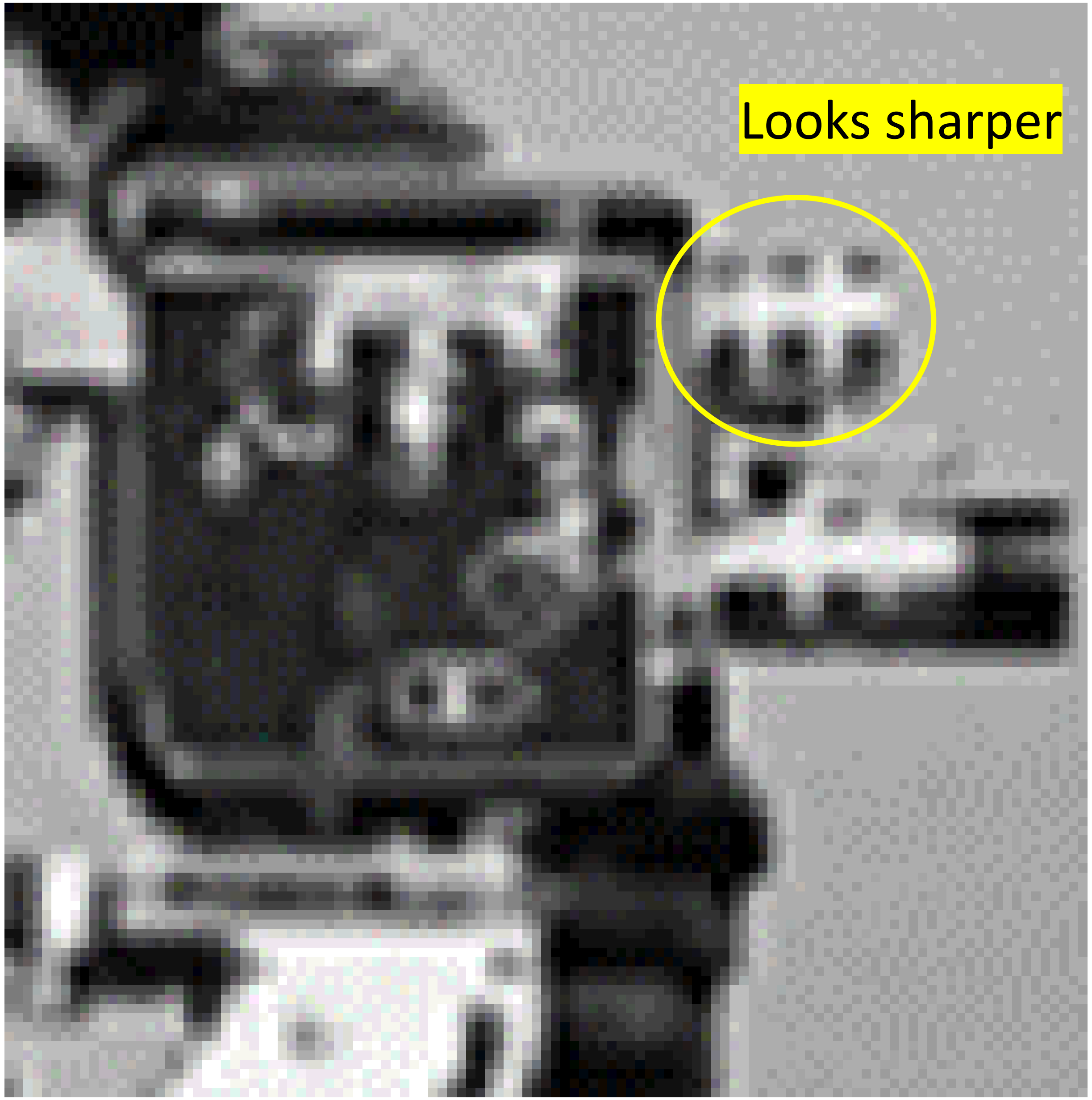


Smoother and Sharper

Comparison between Linear and Bicubic (Zoomed-in) P=4



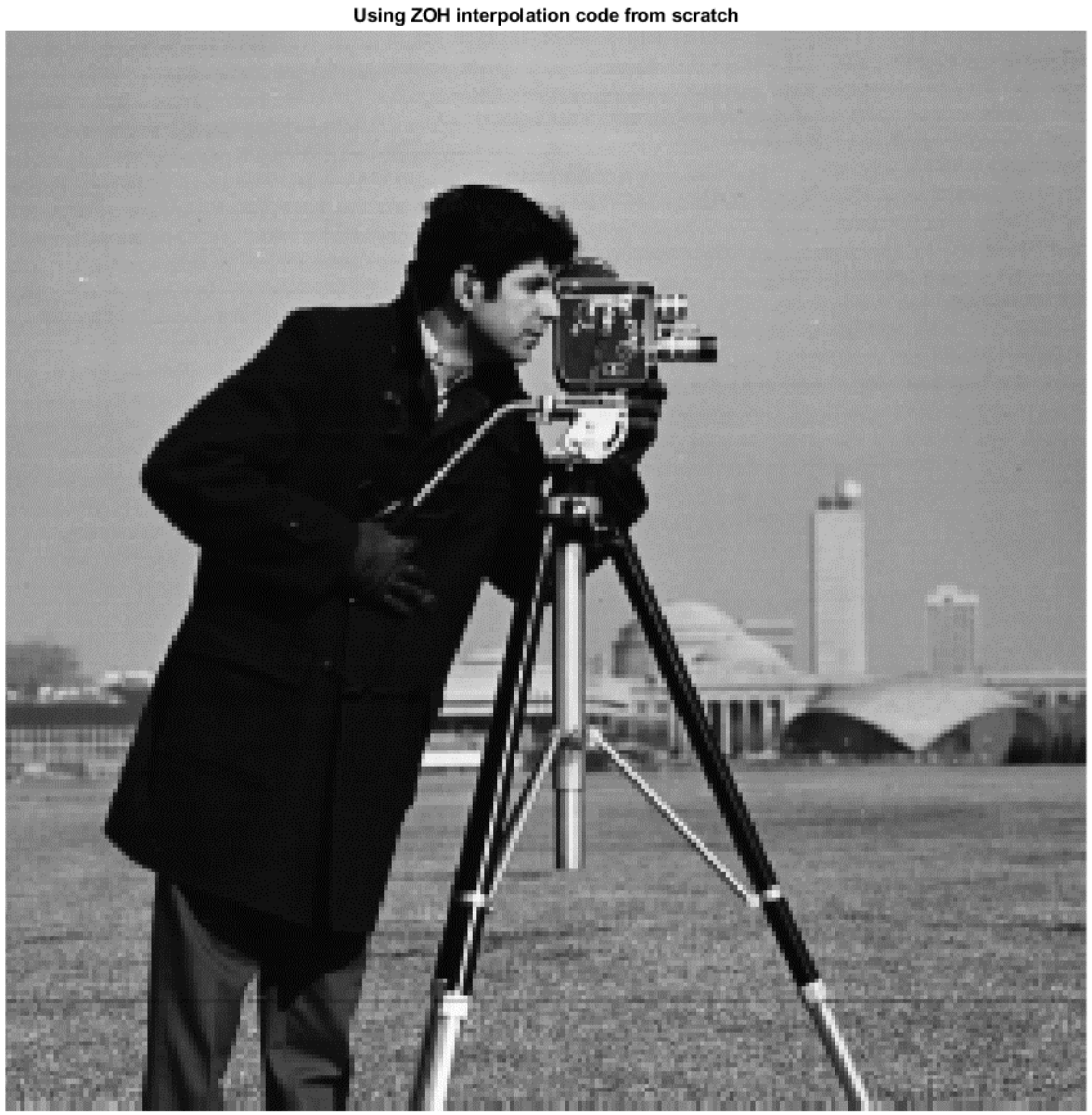
Linear



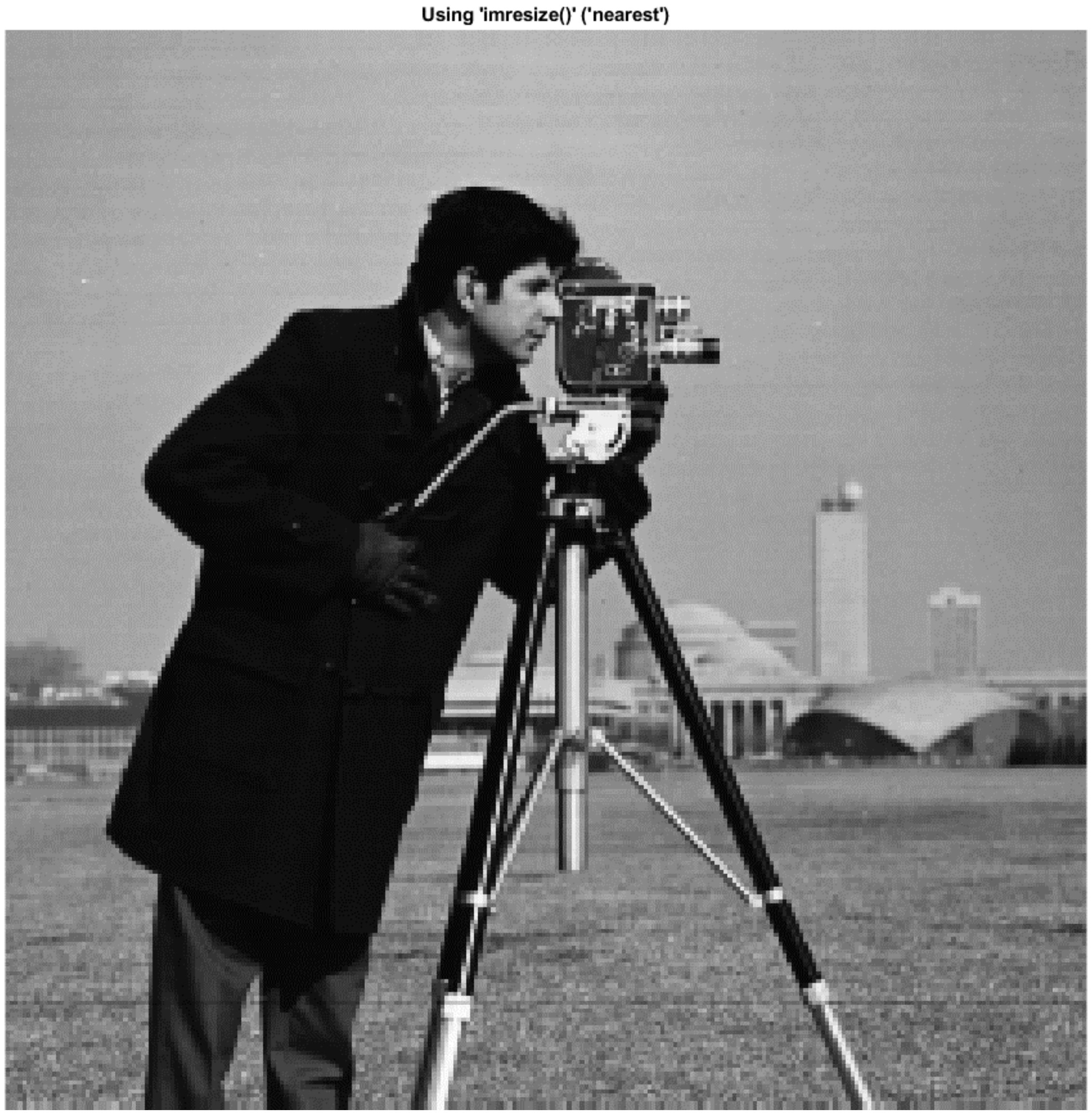
Bicubic

Comparison with imresize()

Interpolation method	Comparison with	MSE
ZOH	“nearest” in imresize()	.0281



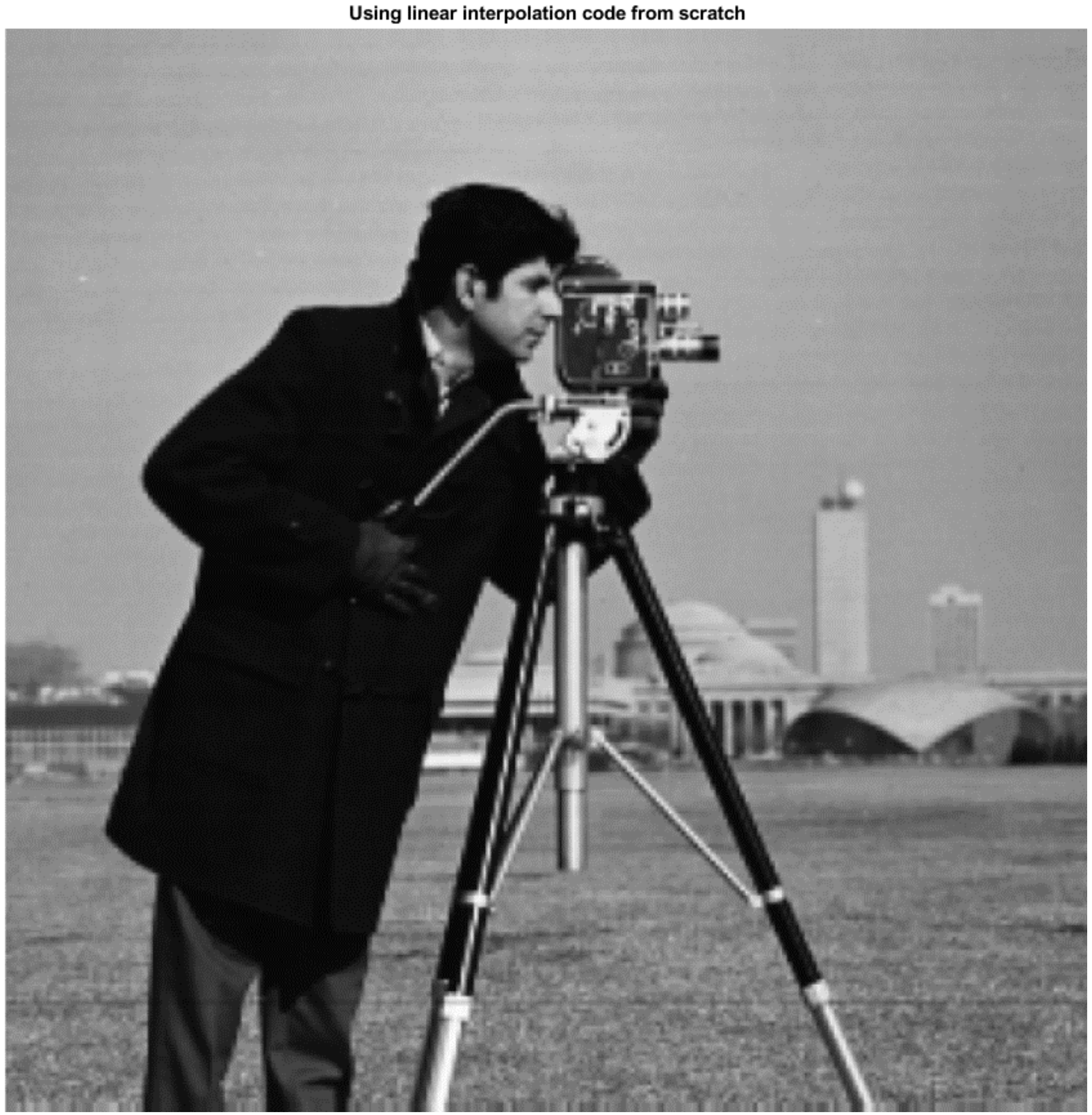
Using ZOH interp. Code from scratch



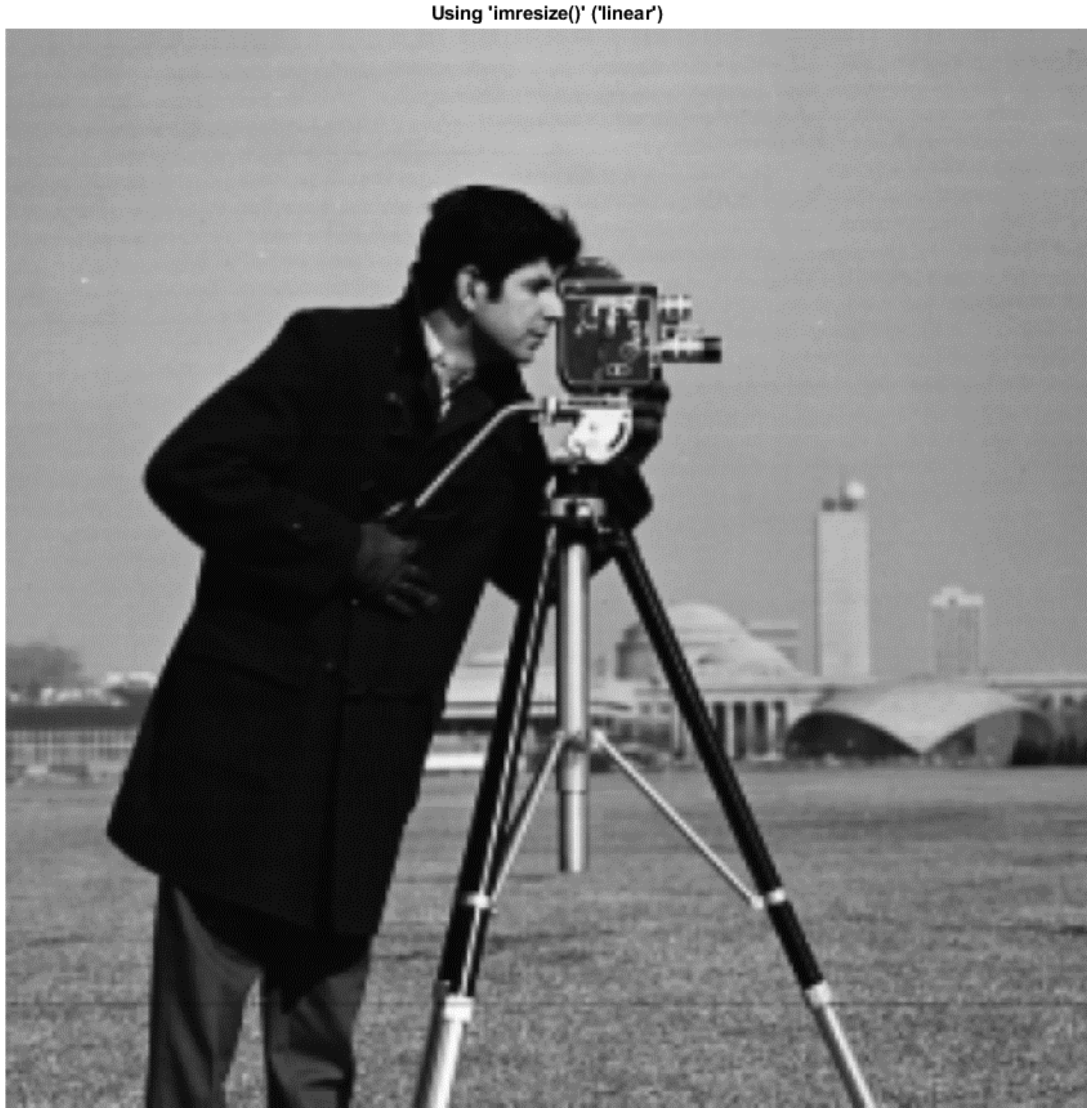
Using imresize (“nearest”)

Comparison with imresize()

Interpolation method	Comparison with	MSE
Linear	“bilinear” in imresize()	.0166



Using linear interp. Code from scratch



Using imresize (“bilinear”)

Comparison with imresize()

Interpolation method	Comparison with	MSE
Bicubic	"bicubic" in imresize()	.0202

Using bicubic interpolation code from scratch

Using bicubic interp. Code from scratch

Using 'imresize()' ('bicubic')

Using imresize ("bicubic")

Interpolation of color images

- The same codes were used to interpolate color images, but the final image was broken down into three parts and used as R,G and B channels to show the colored output image.

```
BB = B(:,1:s*p); %channel 1
BB2 = B(:,s*p+1:2*s*p);
BB3 = B(:,2*s*p+1:3*s*p);
BB(:,:,2) = BB2; %channel 2
BB(:,:,3)=BB3; %channel 3
size(BB)
figure(1);imshow(BB);
```

Original image (128*128)

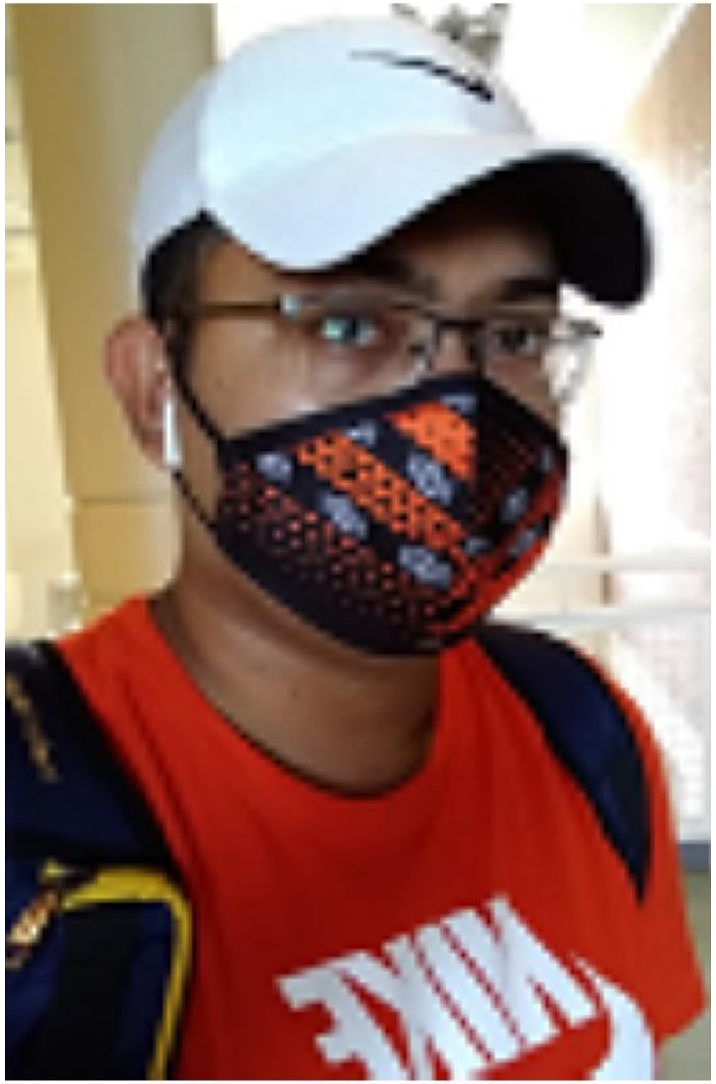
Here,

- B was the interpolated image initially generated where three channels were placed side by side (the whole matrix is 2-D).
- s is the y-dimension (horizontal) of the original image
- p is the zoom ratio.

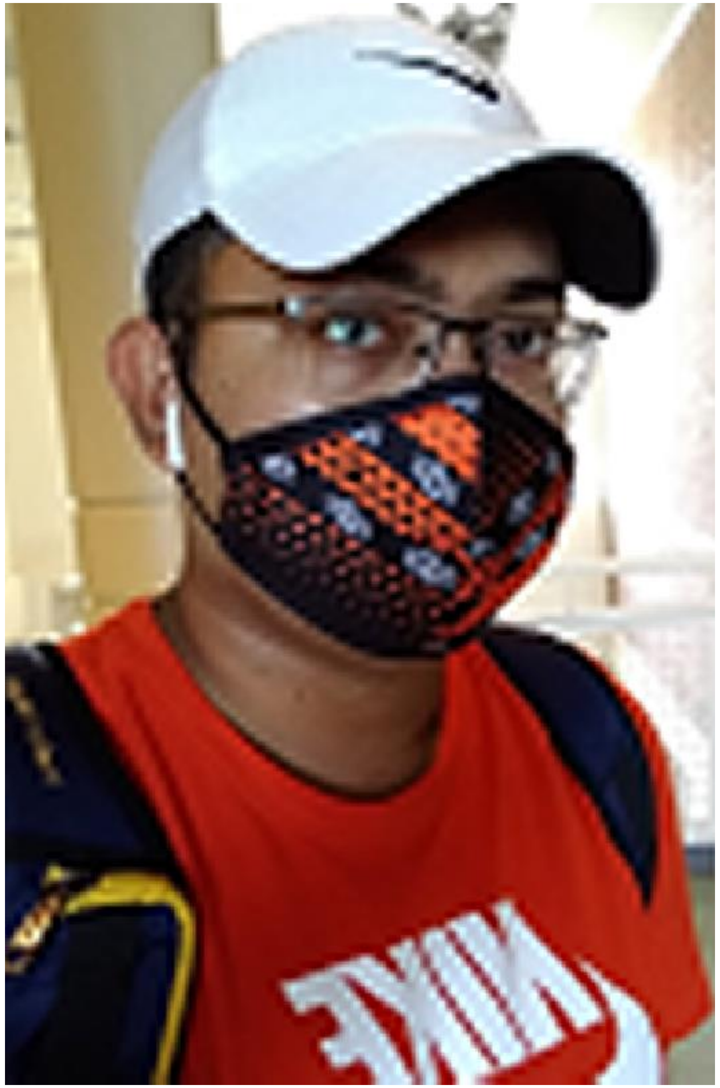
Interpolation of color images

P=4

ZOH



Linear



Bicubic

Image Interpolation (Upsizing and Downsizing)

- One 1024*1024 grayscale was picked ("cameraman.bmp")
- This image was downsized to 128*128

("cameraman_128by128.bmp") using some web-based tool.

- The downsized image was upsized to 1024*1024 using all three interpolation methods and MSE was calculated

```
MSE_scratch = immse(Iorig2,B)
```

- The downsized image was upsized to 1024*1024 again using all three interpolation methods using default function of MATLAB (imresize())and MSE was calculated.

Interpolation method	MSE after comparing with code from scratch	MSE after comparing with imresize()
ZOH	7.3597×10^{-3}	3.1553×10^{-3}
Linear	3.9125×10^{-3}	3.1858×10^{-3}
Bicubic	3.6025×10^{-3}	2.6431×10^{-3}

Image Interpolation (Upsizing and Downsizing)

- The MSEs after comparing with code from scratch match with our intuition with ZOH giving the maximum error while bicubic the minimum.

- While comparing with images formed by imresize(), the MSE after nearest interpolation became less than that of linear interpolation. In order to reverify this, a short code is developed to find MSE of all three methods (using imresize). Similar pattern in the results were found which are shown in the next slide with code. The values are different in the new code because im2double() was not used in that code, while the following values were found after using im2double.

Interpolation method	MSE after comparing with code from scratch	MSE after comparing with imresize()
ZOH	7.3597×10^{-3}	3.1553×10^{-3}
Linear	3.9125×10^{-3}	3.1858×10^{-3}
Bicubic	3.6025×10^{-3}	2.6431×10^{-3}

Image Interpolation (Upsizing and Downsizing)

MSE_calc_for_imresize.m

```
clc;clear; close all;
I = imread('cameraman.bmp'); %read the original image
II=imread('cameraman_128by128.bmp'); %read the downsized image
Ir_zoh = imresize(II,2,"nearest");
Ir_lin = imresize(II,2,"bilinear");
Ir_bicubic = imresize(II,2,"bicubic");
MSE_zoh = immse(I,Ir_zoh(:,:,1))
MSE_lin = immse(I,Ir_lin(:,:,1))
MSE_bicubic = immse(I,Ir_bicubic(:,:,1))
```

Command Window

New to MATLAB? See resources for Getting Started.

```
MSE_zoh =

  205.1735


MSE_lin =

  207.3166


MSE_bicubic =

  172.0469

>>
```

Image Interpolation (Downsizing and upsizing)

- Error images (reconstructed image using code from scratch subtracted from original image) were displayed

ZOH

Linear

Bicubic

Image Registration

Image Registration in MATLAB

- For “cameramanB.bmp” input image, the following control point pairs were chosen manually (these are fours corner points):

```
%manually selected control point pairs for cameramanB:
%%%%%%%%%%%%%%%%%%%%%%%%%
% Ref(x,y) % Inp(v,w) %
%%%%%%%%%%%%%%%%%%%%%%%%%
% (1,1) % (1,129) %
% (1,256) % (129,350) %
% (256,1) % (222,1) %
% (256,256) % (350,222) %
%%%%%%%%%%%%%%%%%%%%%%%%%
```

The matrices are:

```
A =

     1   129     1     0     0     0
     0     0     1     1   129     1
   129   350     1     0     0     0
     0     0     0   129   350     1
   222     1     1     0     0     0
     0     0     0   222     1     1
   350   222     1     0     0     0
     0     0     0   350   222     1


B =

     1
     1
     1
   256
   256
     1
   256
   256


t =

    0.8640
   -0.5004
   64.6904
    0.5004
    0.8640
 -110.9576


T =

    0.8640    0.5004         0
   -0.5004    0.8640         0
   64.6904 -110.9576    1.0000


Tinv =

    0.8667   -0.5020    0.0000
    0.5020    0.8667   -0.0000
   -0.3686  128.6353    1.0000
```

Image Registration in MATLAB

- For “cameramanC.bmp” input image, the following control point pairs were chosen manually (these are fours corner points):

```
%manually selected control point pairs for cameramanC:
%%%%%%%%%%%%%%%%%%%%%%%%%%
% Ref(x,y) % Inp(v,w) %%
%%%%%%%%%%%%%%%%%%%%%%%%%%
% (1,1) % (1,193) %
% (1,256) % (193,525) %
% (256,1) % (333,1) %
% (256,256) % (525, 333) %
%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The matrices are:

```
A =

     1   193     1     0     0     0
     0     0     0     1   193     1
   193   525     1     0     0     0
     0     0     0   193   525     1
   333     1     1     0     0     0
     0     0     0   333     1     1
   525   333     1     0     0     0
     0     0     0   525   333     1


B =

     1
     1
     1
   256
   256
     1
   256
   256


t =

    0.5756
   -0.3329
   64.6668
    0.3329
    0.5756
 -110.4186


T =

    0.5756    0.3329         0
   -0.3329    0.5756         0
   64.6668 -110.4186    1.0000


Tinv =

    1.3020   -0.7529    0.0000
    0.7529    1.3020    0.0000
   -1.0549  192.4510    1.0000
```

Image Registration in MATLAB

- As an example, the code for image registration using linear interpolation for CameramanB image is given below. Other codes are submitted with this ppt.

```
clc; clear; close all;
Iref=imread('cameraman.bmp'); Iref_size = size(Iref)
figure(1);subplot(121);imshow(Iref);
Iinp = imread('cameramanB.bmp'); Iinp_size = size(Iinp)
subplot(122);imshow(Iinp);
Iinp = im2double(Iinp);
%manually selected control point pairs for cameramanB:
% Ref(x,y) % Inp(v,w) %
% (1,1) % (1,129) %
% (1,256) % (129,350) %
% (256,1) % (222,1) %
% (256,256) % (350,222) %
z = [0 0 0];
vw1=[1 129 1]; vw2=[129 350 1]; vw3=[222 1 1]; vw4=[350 222 1];
A = [vw1 z; z vw1; vw2 z; z vw2; vw3 z; z vw3; vw4 z; z vw4]
B = [1 1 1 256 256 1 256 256]'
```

Continued at the next slide

Image Registration in MATLAB

```
t = linsolve(A,B)
T = [t(1:3) t(4:6) [0 0 1]']
Tinv = inv(T)
Tinv_size = size(Tinv);


n = Iref_size(1);
s = 1:n;
[X,Y] = ndgrid(s,s);
Xr = reshape(X,[],1); %reshape for doing matrix operation
Yr = reshape(Y,[],1);
xy1 = [Xr Yr ones(length(s)^2,1)];
xy1_size = size(xy1);
vw1 = xy1*Tinv;
vw1_size = size(vw1);
vw = vw1(:,1:2); %All new pixel locations in the input image,
%one co-ordinate per row.
```

Continued at the next slide

Image Registration in MATLAB

```
XI = reshape(vw(:,1),n,n); %Reshape back
YI = reshape(vw(:,2),n,n); %Reshape back
UI=XI-floor(XI); VI=YI-floor(YI);
c1=(1-UI).*(1-VI); c2=(1-UI).*VI; c3=UI.*(1-VI); c4=UI.*VI;
X1 = floor(XI);Y1=floor(YI); X2=floor(XI)+1; Y2=floor(YI)+1;
X1(find(X1>Iinp_size(1)))=Iinp_size(1);
Y1(find(Y1>Iinp_size(2)))=Iinp_size(2);
X2(find(X2>Iinp_size(1)))=Iinp_size(1);
Y2(find(Y2>Iinp_size(2)))=Iinp_size(2);
X1(find(X1<1))=1; Y1(find(Y1<1))=1;X2(find(X2<1))=1; Y2(find(Y2<1))=1;
for i=1:n
       for j = 1:n
              I1(i,j) = Iinp(X1(i,j),Y1(i,j));
              I2(i,j) = Iinp(X1(i,j),Y2(i,j));
              I3(i,j) = Iinp(X2(i,j),Y1(i,j));
              I4(i,j) = Iinp(X2(i,j),Y2(i,j));
       end
end
BB=c1.*I1+c2.*I2+c3.*I3+c4.*I4;
size(BB)
figure(2);imshow(BB);
```

Image Registration (CameramanB)

Reference

Input

Registered Images for CameramanB (zoh and linear - for comparison)

ZOH

Linear

Registered Images for CameramanB (linear and bicubic - for comparison)

Linear

Bicubic

Image Registration (CameramanC)

Reference

Input

Registered Images for CameramanC (zoh and linear – for comparison)

ZOH

Linear

Registered Images for CameramanC (linear and bicubic – for comparison)

Linear

Bicubic

Thanks