2. Histogram Equalization

. (a) Write a function hist_eq.m that performs histogram equalization on an intensity image. The function should take as inputs an intensity image and the number of gray level value bins. Create a separte m-file for this function.

```
function [ g ] = hist_eq( image, graylevels )
%HISTOEQ Equalizes the histogram of an input image
%   Detailed explanation goes here


% Variables
rk = zeros(512,512);
sk = zeros(graylevels,1);
nk = zeros(graylevels,1);
enk = zeros(graylevels,1);
prk = zeros(graylevels,1);
psk = zeros(graylevels,1);
img_size = size(image,1) * size(image,2);

%Normalize



% Find the number of pixels that have intensity
% rk. (Generate Histogram)
nk = imhist(image(:,:,1),graylevels);
for i = 1:graylevels
   %nk(i,1) = sum(sum(image(:,:,1) == i));
   prk(i,1) = nk(i,1)/img_size;
end

% Perform transformation on input image
for k = 1:graylevels
    sk(k,1) = round((graylevels - 1) * dsum(prk, 1, k));
    enk(sk(k,1)+1,1) = enk(sk(k,1)+1,1) + nk(k,1);
end


% Equalize Histogram
for k = 1:graylevels
     psk(k,1) = enk(k,1)/img_size;
end

% Create new image
for row = 1 : 512
    for col = 1 : 512
        norm_value = normalize(double(image(row,col,1)),0,255,graylevels);
        rk(row,col) = denormalize(sk(norm_value,1),0,255,graylevels);
    end
end


g = uint8(rk);

end

%*********************************************
function [ y ] = normalize(x, A, B, range)
%Y Normalized value

    y = uint8(1 + ((x - A) * (range - 1))/(B-A));
end

%*********************************************

function [ x ] = denormalize(y, A, B, range)
```

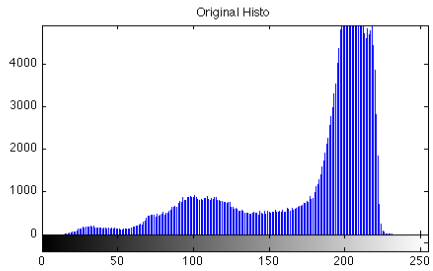
```
%X Denormalized value

    x = uint8(A + ((B-A)*(y-1))/(range - 1));
end
%*********************************************

function [ sum ] = dsum(f,a,b)
%DSUM Performs a definite summation
%   Detailed explanation goes here

sum = 0;
for j = a:b
    sum = sum + f(j);
end

end
```

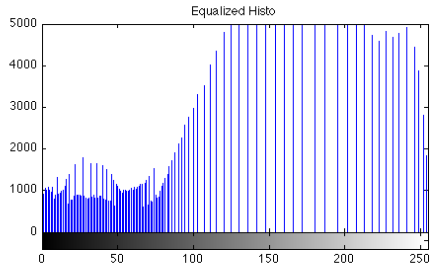
. (b) Perform histogram equalization on the jetplane image using 256, 128, and 64 bins. Com- pare the original image and the histogram equalized images by plotting the corresponding histograms and images side-by-side in a 2 × 2 subplot matrix.

L = 256

```
EDU>> f = imread('jetplane.tif');
EDU>> g = hist_eq(f,256);
EDU>> subplot(2,2,1), imhist(f(:,:,1));
      subplot(2,2,2), imshow(f(:,:,1));
      subplot(2,2,3), imhist(g);
      subplot(2,2,4), imshow(g);
```

Output:

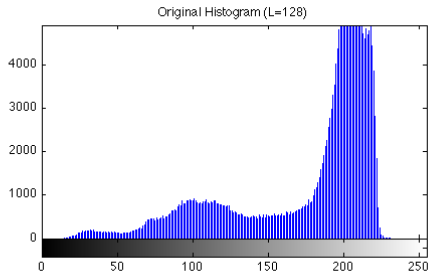


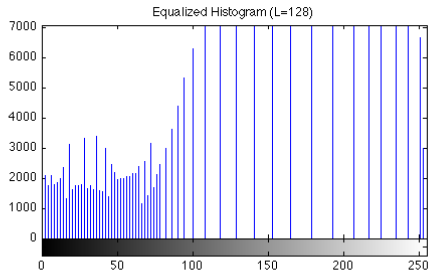


L = 128

```
EDU>> f = imread('jetplane.tif');
EDU>> g = hist_eq(f,128);
EDU>> subplot(2,2,1), imhist(f(:,:,1));
      subplot(2,2,2), imshow(f(:,:,1));
      subplot(2,2,3), imhist(g);
      subplot(2,2,4), imshow(g);
```

Output:

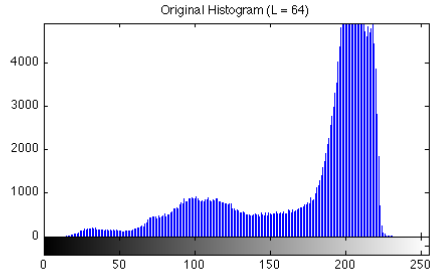


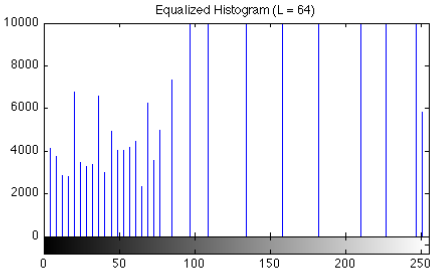


L = 64

```
EDU>> f = imread('jetplane.tif');
EDU>> g = hist_eq(f,64);
EDU>> subplot(2,2,1), imhist(f(:,:,1));
      subplot(2,2,2), imshow(f(:,:,1));
      subplot(2,2,3), imhist(g);
      subplot(2,2,4), imshow(g);
```

Output:

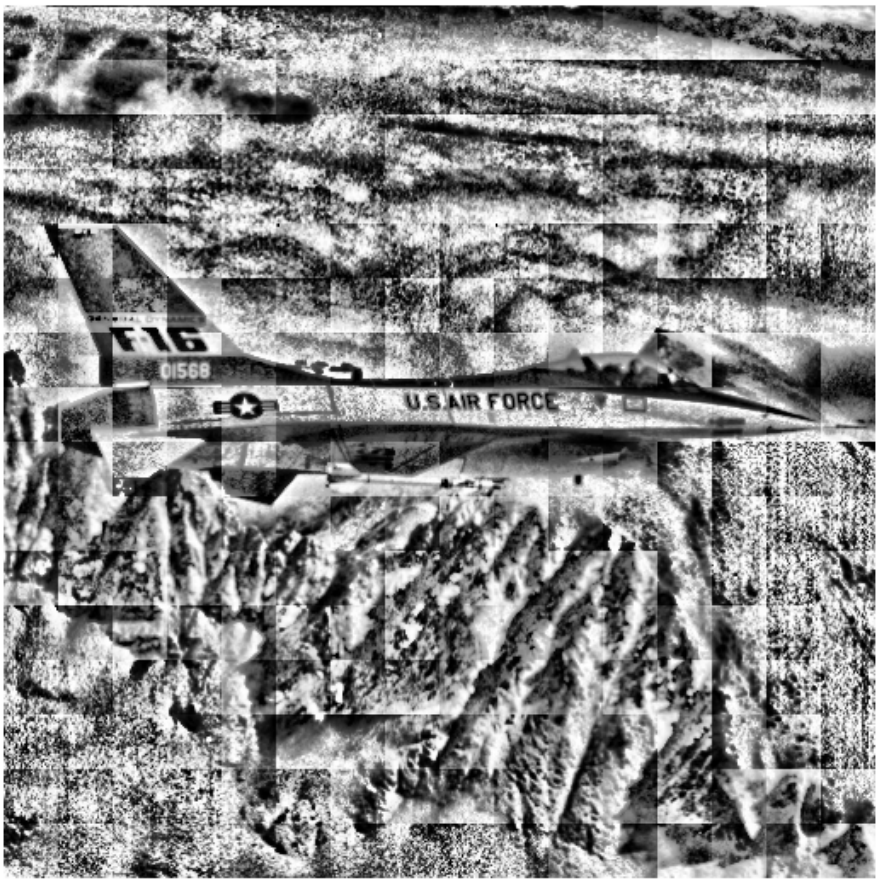




. (c) Perform the equalization in 32 × 32 blocks. Display the output image. You will find blockproc.m useful.

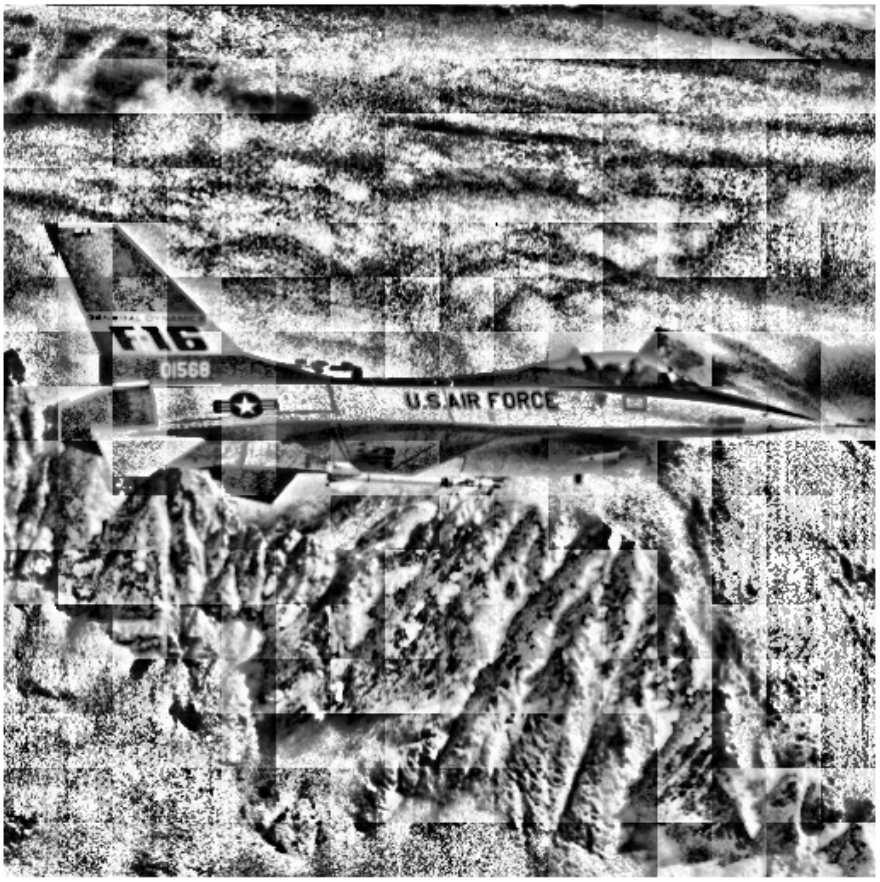
L = 256

```
EDU>> fun = @(block_struct) hist_eq(block_struct.data, 256);
EDU>> f = imread('jetplane.tif');
EDU>> b = blockproc(f(:,:,1),[32 32],fun);
EDU>> imshow(b);
```



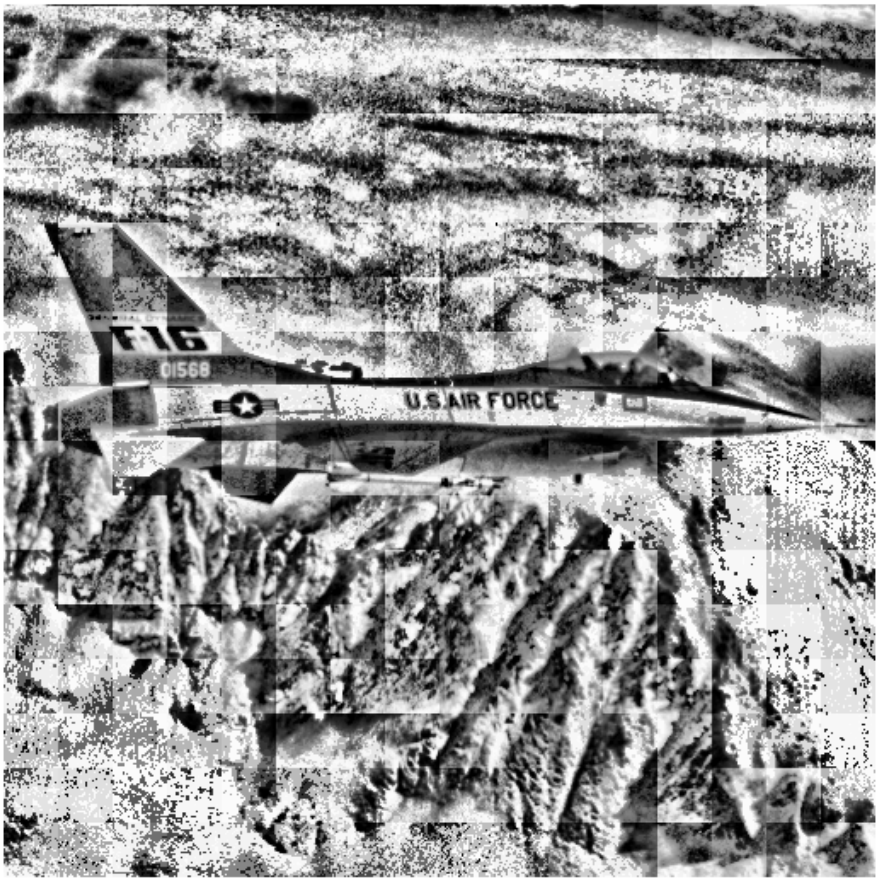
L = 128

```
EDU>> fun = @(block_struct) hist_eq(block_struct.data, 128);
EDU>> f = imread('jetplane.tif');
EDU>> b = blockproc(f(:,:,1),[32 32],fun);
EDU>> imshow(b);
```



L = 64

```
EDU>> fun = @(block_struct) hist_eq(block_struct.data, 64);
EDU>> f = imread('jetplane.tif');
EDU>> b = blockproc(f(:,:,1),[32 32],fun);
EDU>> imshow(b);
```



3. Morphology

. (a) Threshold the image SJEarthquakesteampic.jpg to detect faces. Be sure to describe how you obtained your threshold. You may find this is easier in another colorspace such as HSV.

```
function [ g ] = imthreshold( f, T1 )
%G Provides a thresholded version of the input
%   Based on a threshold value, the input image
%   is rasterized pixel-by-pixel, generating an
%   output image that's either 0(Black) or 1(White)

%Grab some stats on image
ROWS = size(f,1);
COLS = size(f,2);
%RGB = size(f,3);
%IMG_SIZE = ROWS * COLS;

hsv_img = rgb2hsv(f);
bw = zeros(ROWS,COLS);

%Threshold Input
for row = 1:ROWS
    for cols = 1:COLS
        if(hsv_img(row,cols,1) < T1 )
            bw(row,cols) = 1;
        else
            bw(row,cols) = 0;
        end
    end

end


%MN = [3,3];
%S = strel('rectangle', MN);

%Pattern that resembles the makeup of a face
MN = [0, 0, 1, 1, 1, 1, 1, 1, 0, 0;
      0, 1, 1, 1, 1, 1, 1, 1, 1, 0;
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1;
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1;
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1;
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1;
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1;
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1;
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1;
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1;
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1;
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1;
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1;
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1;
      0, 1, 1, 1, 1, 1, 1, 1, 1, 0;
      0, 0, 1, 1, 1, 1, 1, 1, 0, 0;
      0, 0, 0, 1, 1, 1, 1, 0, 0, 0;
      0, 0, 0, 0, 1, 1, 0, 0, 0, 0;
      0, 0, 0, 0, 0, 1, 0, 0, 0, 0;
      0, 0, 0, 0, 0, 1, 0, 0, 0, 0];
S = strel(MN);
eroded_output = imerode(bw,S);
opened_output = imdilate(eroded_output,S);
dilated_output = imdilate(opened_output,S);
imshow(dilated_output);
%imshow(bw);


%Look for connected components and bound (NOT WORKING)
cc = bwconncomp(dilated_output);
```

```
stats = regionprops(cc, 'Area', 'BoundingBox');
bb = cat(1, stats.BoundingBox);
idx = find([stats.Area] > 900);
bw2 = ismember(labelmatrix(cc), idx);
imshow(bw2);
hold on
for elements = 1:numel(idx)
    %bb = find([stats.BoundingBox] == idx(elements));
    plot(bb(idx(elements),1), bb(idx(elements),2), bb(idx(elements),3),
bb(idx(elements),4), 'b*');
end
hold off
L = labelmatrix(cc);
L2 = bwlabel(bw);
g = L2;
end
```

Threshold was attained by evaluating the hue levels on the image histogram of the input image and determining what value best corresponds with the skin color of the players. I settled for T = 0.1.
The code above isn't quite complete as I was trying to explore some of the different properties of the regionprops method.

. (b) Use morphological operations to clean the image and count the number of players in the image.

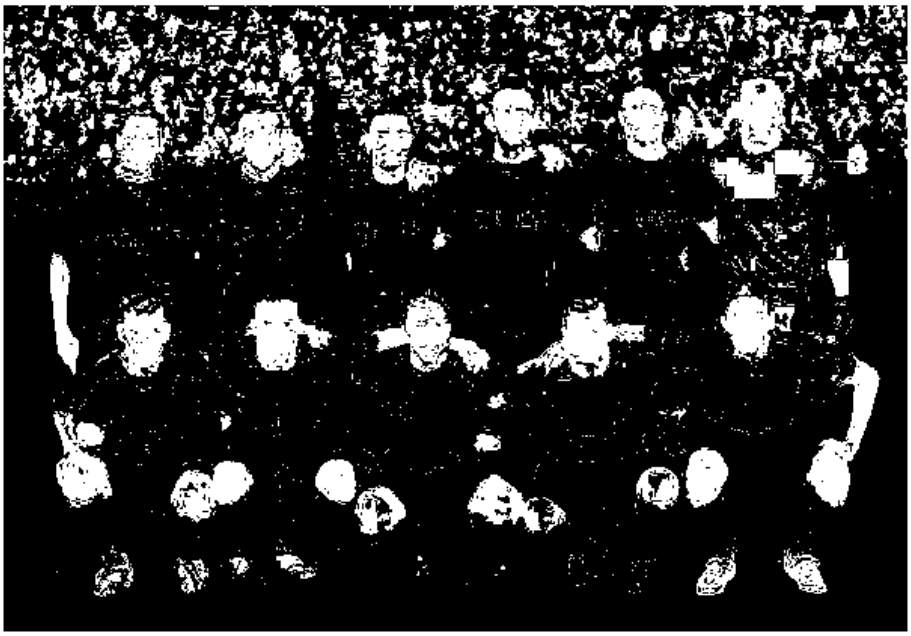


Figure 1 - Thresholded Image at T = 0.1

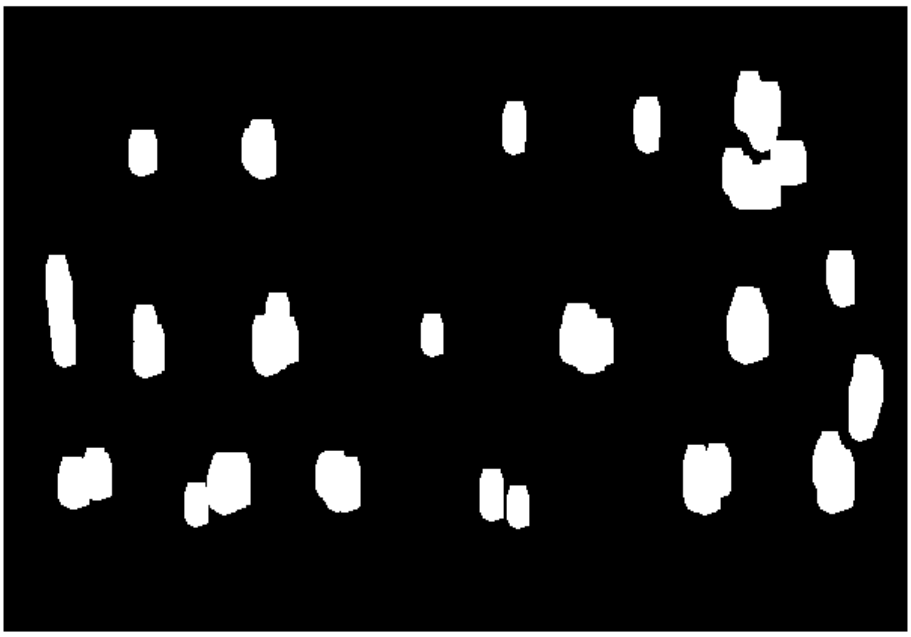


Figure 2 - Tinkered with Opening and then dilating the image using an SE that sort of matches a facial pattern. You can see some of the players heads, upper and lower limbs.

Figure 3 - Wanted to see if I could label the components. This image was taken with the same threshold, but you'll notice the pattern is different. Some of the faces and limbs are not there as depicted in Figure 2. Went more for 'V' pattern on the face.

. (c) Create an output image that has a bounding box around each face. Use regionprops.m.

Sorry, didn't have much luck here. ☹

4. Filtering

. (a) Consider image DSCN0479-001.JPG as a perfect image. Add white Gaussian noise with variance 0.005.

Code:

```
function [ g ] = box_filter( f, MN, v )
%G Summary of this function goes here
%   Detailed explanation goes here


%Add Gaussian Noise to image
img = imnoise(f,'gaussian', 0, v);


%Perform MxN Filtering
filter = fspecial('average',MN);
fimg = imfilter(img, filter, 'replicate');


g = fimg;

end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ g ] = median_filter( f, MN, v )
%G Summary of this function goes here
%   Detailed explanation goes here

%Add Gaussian Noise to image
img = imnoise(f,'gaussian', 0, v);

%Perform MxN Median Filtering
fimg = medfilt2(rgb2gray(img), MN);

g = fimg;

end
```

Smooth with a 3 × 3 and 7 × 7 box filter and a median filter.

3x3 Box

```
EDU>> g = box_filter(f,[3 3],0.005);
EDU>> imshow(g)
```

7x7 Box

```
EDU>> g = box_filter(f,[7 7],0.005);
EDU>> imshow(g)
```

3x3 Median

```
EDU>> g = median_filter(f, [3 3],0.005);
EDU>> imshow(g)
```

7x7 Median

```
EDU>> g = median_filter(f, [7 7],0.005);
EDU>> imshow(g)
```

Compute the mean squared error (MSE) and the peak signal-to-noise ratio (PSNR) for the noise reduced images. Which filter has the best results based on the error measures? How do the results compare visually?

```
function [ e ] = mse(f1,f2)
%E Mean Squared Error
%   Detailed explanation goes here


[M N] = size(img1);
diff = (double(f1) - double(f2)).^2;
MSE = mean(mean(diff,2),1)/(M * N);

e = MSE;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ res ] = psnr( mse )
%RES Peak Signal To Noise Ratio
%   Detailed explanation goes here

res = 20 * log10(255/sqrt(mse));

end
```

Results starting on next page:

3x3 Box

```
EDU>> f = imread('DSCN0479-001.jpg');
EDU>> g = box_filter(f,[3 3],0.005);
EDU>> e = mse(f,g);
EDU>> e

e(:,:,1) =

   2.8967e-04


e(:,:,2) =

   2.7165e-04


e(:,:,3) =

   2.6750e-04

EDU>> psnr = psnr(e);
EDU>> psnr

psnr(:,:,1) =

   83.5118


psnr(:,:,2) =

   83.7907


psnr(:,:,3) =

   83.8575
```

7x7 Box

```
EDU>> f = imread('DSCN0479-001.jpg');
EDU>> g = box_filter(f,[7 7],0.005);
EDU>> e = mse(f,g);
EDU>> e

e(:,:,1) =

   6.1865e-04


e(:,:,2) =

   5.9147e-04


e(:,:,3) =

   5.9707e-04

EDU>> psnr = psnr(e);
EDU>> psnr
ans(:,:,1) =

   80.2163


ans(:,:,2) =

   80.4115


ans(:,:,3) =

   80.3706
```

3x3 Median

```
EDU>> f = imread('DSCN0479-001.jpg');
EDU>> g = median_filter(f, [3 3],0.005);
EDU>> e = mse(f,g);
EDU>> e = mse(f(:,:,1),g);
EDU>> e

e =

    0.0066
EDU>> psnr = psnr(e);
EDU>> psnr

psnr =

   69.9477
```

7x7 Median

```
EDU>> g = median_filter(f, [7 7],0.005);
EDU>> e = mse(f(:,:,1),g);
EDU>> e

e =

    0.0073
EDU>> psnr = psnr(e);
EDU>> psnr
ans =
     69.5041
```

. (b) Repeat (a) with salt and pepper noise with noise density 0.05.

```
function [ g ] = box_filterSP( f, MN, d )
%G Summary of this function goes here
%   Detailed explanation goes here


%Add Gaussian Noise to image
img = imnoise(f,'salt & pepper', d);


%Perform MxN Filtering
filter = fspecial('average',MN);
fimg = imfilter(img, filter, 'replicate');


g = fimg;

end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ g ] = median_filterSP( f, MN, d )
%G Summary of this function goes here
%   Detailed explanation goes here

%Add Gaussian Noise to image
img = imnoise(f,'salt & pepper', d);

%Perform MxN Median Filtering
fimg = medfilt2(rgb2gray(img), MN);

g = fimg;

end
```

3x3 Box

```
EDU>> g = box_filterSP(f, [3 3], 0.05);
EDU>> imshow(g);
```


```
EDU>> e = mse(f,g);
EDU>> psnr = psnr(e);
EDU>> e

e(:,:,1) =

   4.4759e-04

e(:,:,2) =

   4.1406e-04

e(:,:,3) =

   4.1651e-04

EDU>> psnr

psnr(:,:,1) =

   81.6220

psnr(:,:,2) =

   81.9602

psnr(:,:,3) =

   81.9345
```

7x7 Box

```
EDU>> g = box_filterSP(f, [7 7], 0.05);
EDU>> imshow(g);
```


```
EDU>> e = mse(f,g);
EDU>> e

e(:,:,1) =

   6.6548e-04

e(:,:,2) =

   6.3068e-04

e(:,:,3) =

   6.4416e-04
EDU>> psnr = psnr(e);
EDU>> psnr
ans(:,:,1) =

   79.8994

ans(:,:,2) =

   80.1327

ans(:,:,3) =

   80.0408
```

3x3 Median

```
EDU>> g = median_filterSP(f, [3 3], 0.05);
EDU>> imshow(g);
```


```
EDU>> e = mse(f(:,:,1),g);
EDU>> e

e =

    0.0065

EDU>> psnr = psnr(e);
EDU>> psnr
ans =

   70.0307
```

7x7 Median

```
EDU>> g = median_filterSP(f, [7 7], 0.05);
EDU>> imshow(g);
```


```
EDU>> e = mse(f(:,:,1),g)
e =
        0.0073
EDU>> psnr = psnr(e)
ans =
        69.5176
```

. (c) Do the filtering again but this time on a real noisy image DSCN0482-001.JPG obtained at higher ISO. Compare the results visually only this time. Which filter works best for "real" noise?

. **3x3 Box**

```
EDU>> f = imread('DSCN0482-001.jpg');
EDU>> g = box_filter(f,[3 3], 0);
EDU>> imshow(g)
EDU>> e = mse(f,g);
EDU>> e

e(:,:,1) =

   1.9546e-04


e(:,:,2) =

   1.7728e-04


e(:,:,3) =

   1.7465e-04

EDU>> psnr = psnr(e)
ans(:,:,1) =

   85.2202


ans(:,:,2) =

   85.6441


ans(:,:,3) =

   85.7090
```

. **7x7 Box**

```
EDU>> g = box_filter(f,[7 7], 0);
EDU>> e = mse(f,g);
EDU>> e

e(:,:,1) =

   4.0753e-04


e(:,:,2) =

   3.8336e-04


e(:,:,3) =

   3.9420e-04
EDU>> psnr = psnr(e)
```

```
ans(:,:,1) =

   82.0292


ans(:,:,2) =

   82.2947


ans(:,:,3) =

   82.1736
```

. **3x3 Median**

```
EDU>> g = median_filter(f, [3 3], 0);
EDU>> e = mse(f,g);
EDU>> e = mse(f(:,:,1),g);
EDU>> e

e =

    0.0055

EDU>> 20 * log10(255/sqrt(e))

ans =

   70.7040
```

.

7x7 Median

```
EDU>> g = median_filter(f, [7 7], 0);
EDU>> e = mse(f(:,:,1),g);
EDU>> e

e =

    0.0059

EDU>> 20 * log10(255/sqrt(e))

ans =

   70.4405
```

3x3 Box has a higher PSNR for this noisy image, thus performing better for real noise.