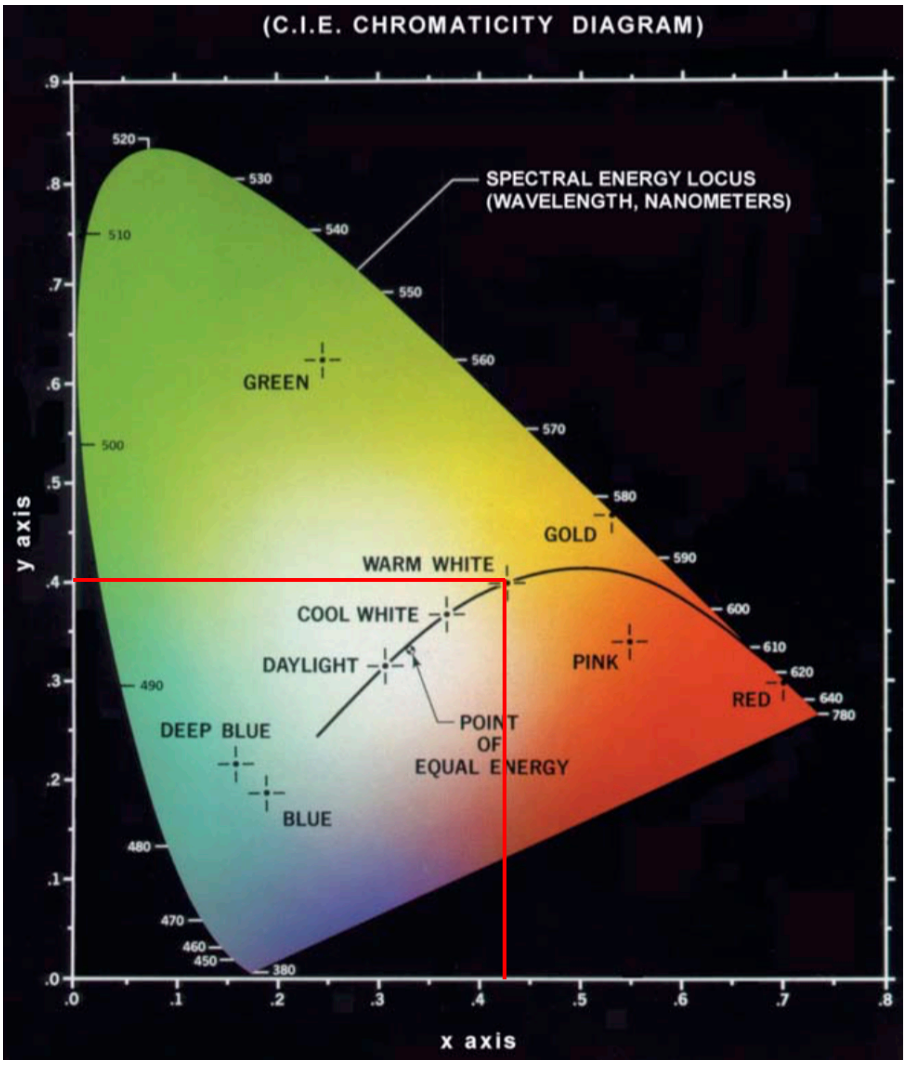
Ray Li
CSC 481
Dr. Furst
10/13/16

6.1 Give the percentages of red (X), green (Y), and blue (Z) light required to generate the point labeled "warm white" in Fig. 6.5.



As shown above, red is approximately 43%, green is approximately 40% and therefore, blue is approximately 17% (1 – (43% + 40%)).

6.2 Consider any two valid colors c1 and c2 with coordinates (x1, y1) and (x2, y2) in the chromaticity diagram of Fig. 6.5. Derive the necessary general expression(s) for computing the relative percentages of colors c1 and c2 composing a given color that is known to lie on the straight line joining these two colors.

Point C(x3, y3) is the color that lines between two known colors, A(x1, y1) and B(x2, y2). The distance between A and B is the Euclidean distance, d1, shown as

$$d1 = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}.$$

The distances between A and C is d2, as shown below,

$$d2 = \sqrt{(x3 - x1)^2 + (y3 - y1)^2}.$$

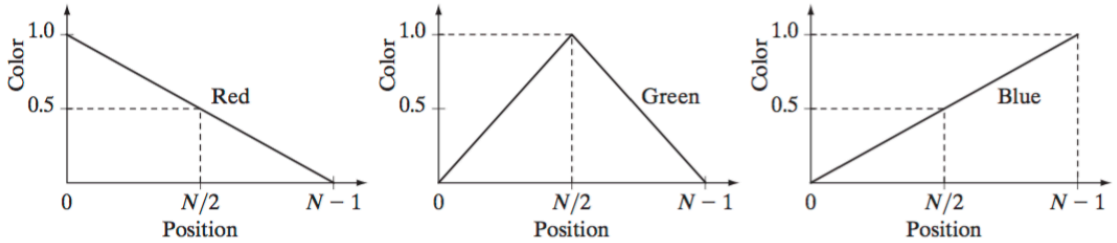
Therefore, the percentage of color A in C would be p1, calculated as

$$p1 = \frac{d1 - d2}{d1} \times 100\%,$$

and the percentage of color B in C would be p2, calculated as

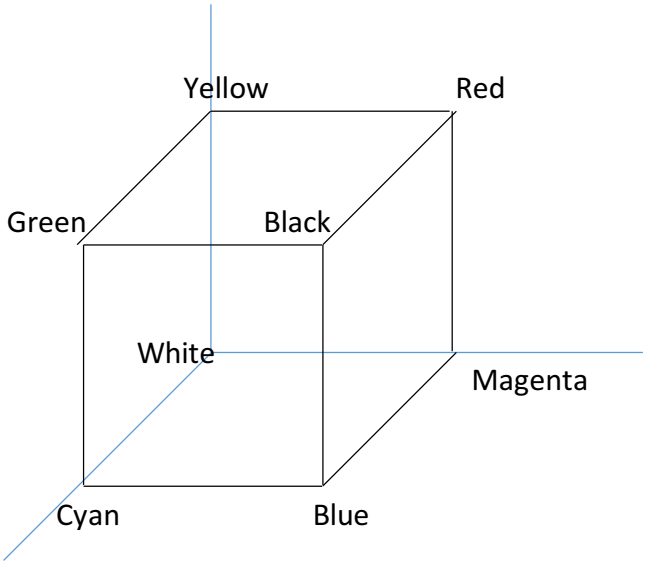
$$p2 = 1 - p1.$$

6.5 In a simple RGB image, the R, G, and B component images have the horizontal intensity profiles shown in the following diagram. What color would a person see in the middle column of this image?



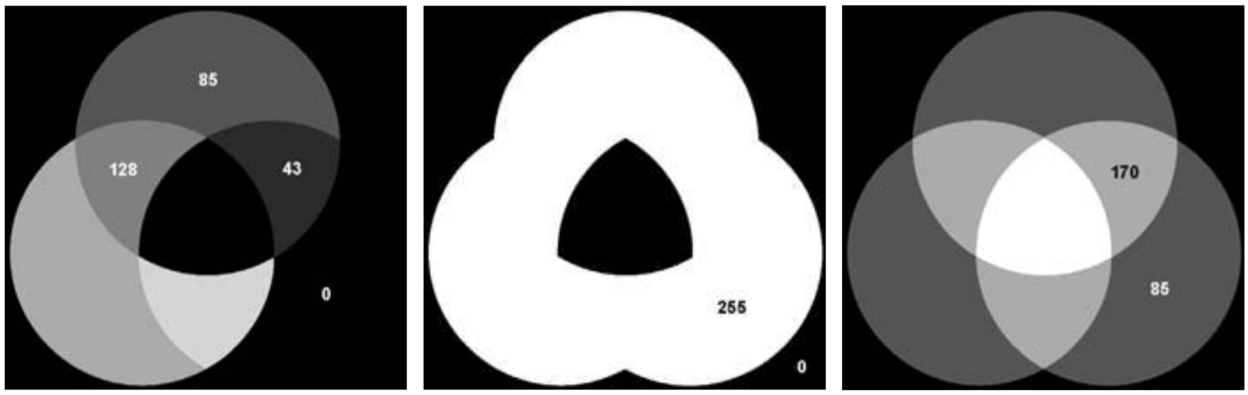
To comprehend this color, assuming all three color are at the same value, 0.5, which will give us a gray color. By increasing the green value to 1, it will add more green to the gray color. Therefore, a person would see a grayish green color in the middle column of this image.

6.8 Consider the RGB color cube shown in Fig 6.8, and answer each of the following:
(b) Suppose that we replace every color in the RGB cube by its CMY color. This new cube is displayed on an RGB monitor. Label with a color name the eight vertices of the new cube that you would see on the screen.



As shown on the left, here are the coordinates of each color:
(0, 0 ,0) – White
(1, 1, 1) – Black
(1, 0, 0) – Cyan
(0, 1, 0) – Magenta
(0, 0, 1) – Yellow
(1, 1, 0) – Blue
(1, 0, 1) – Green
(0, 1, 1) – Red

6.16 The following 8-bit images are (left to right) the H, S, and I component images from Fig. 6.16. The numbers indicate gray-level values. Answer the following question, explaining the basis for your answer. If it is not possible to answer a question based on the given information, state why you cannot do so.
a) Give the gray-level values of all regions in the hue image.



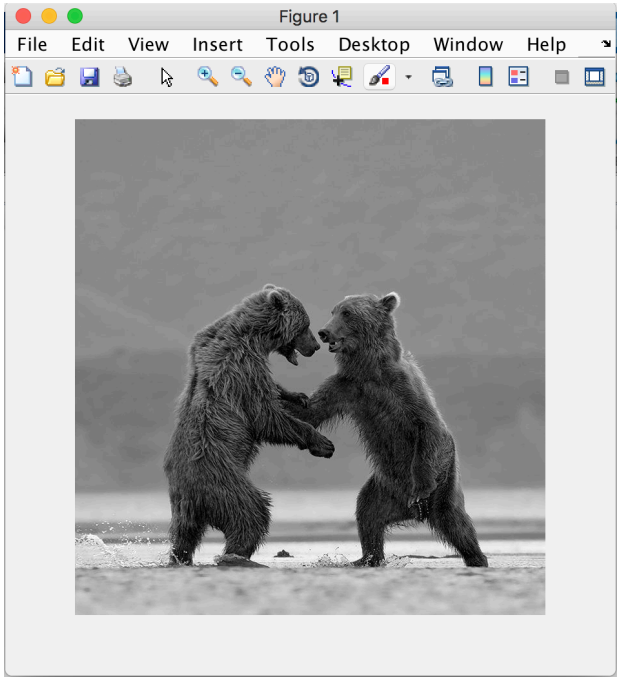
(a) (b) (c)

The gray level values in the white regions, which are the fully saturated color regions in the original image, are all 255 and the black region, which is the white region in the original image, in the center of the hue image, has a gray level value of 0.

Histogram Equalization

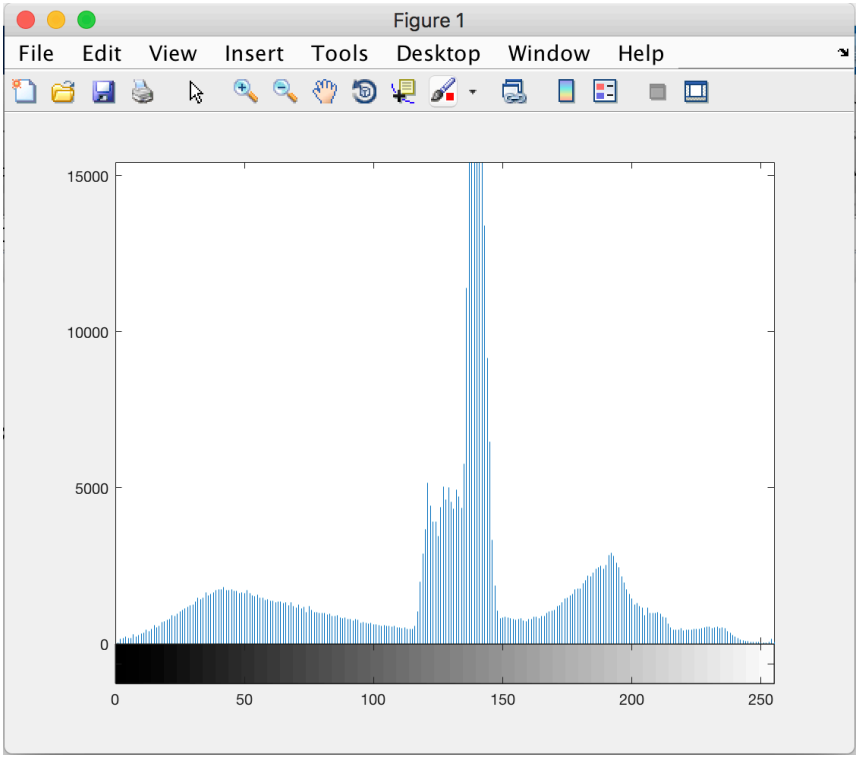
a) **Read and display an image of your choice. (To help you with the next problem, look for an image that has interesting objects of different intensities.)**

```
>> m = imread(bear.jpg');
>> m2 = rgb2gray(m);
>> imshow(m2);
```



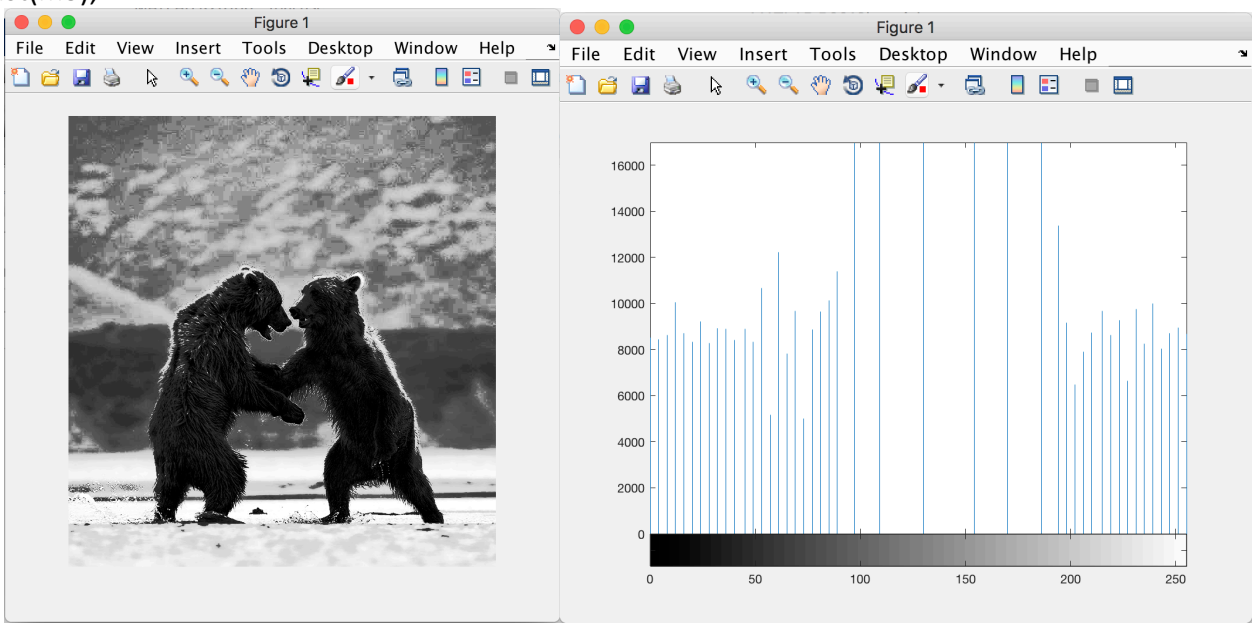
b) **Calculate and display the histogram of this image.**

```
>> imhist(m2);
```



c) Enhance the contrast of the intensity image using histogram equalization and display both the uniform histogram and the new enhanced intensity image.

```
>> m3 = histeq(m2);
>> imshow(m3);
>> imhist(m3);
```



d) Explain why the two histograms (of the original image and of the enhanced image) are different.

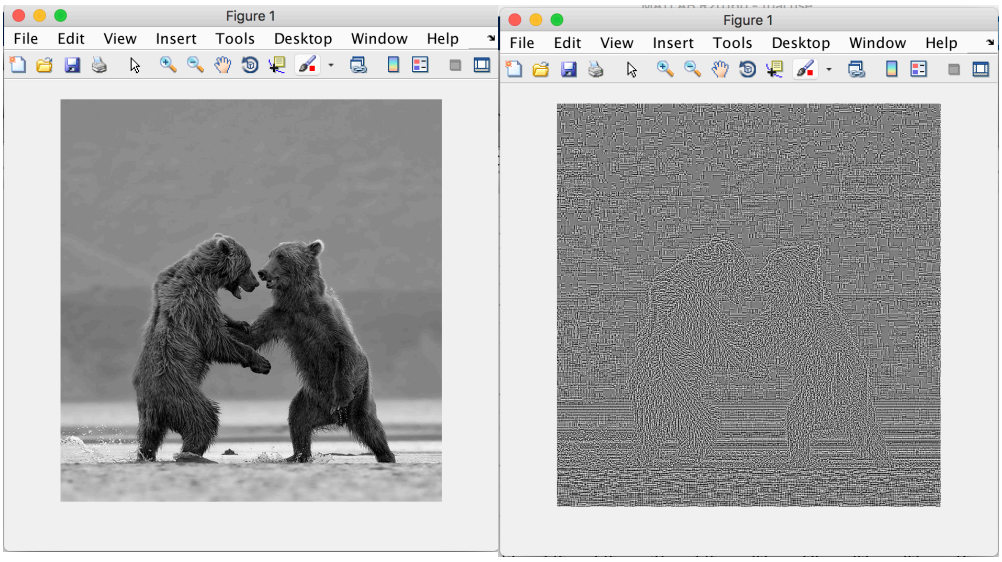
The two histograms are different because of the histogram equalization. Histogram equalization increases the global contrast of the image by more evenly distribute the pixel values across the whole intensity level range. Since the total number of pixels do not change in the image, many pixels will be grouped into with other pixels, or being discretized, to make the histogram as flat as possible. In Matlab, histeq() defaults to equalize the histogram into 64 discrete gray levels and so the uniform histogram has 64 unique intensity levels that have pixels.

e) Apply a local enhancement approach on this image and show your results. Before you start, consider how your image might call for a particular window size. For fun, you might want to try a few different window sizes.

The size of my image is 768x729. Since both rows and columns are divisible by 3, I choose to use a 3x3 window to start with and try a few more window sizes that are multiples of size 3.

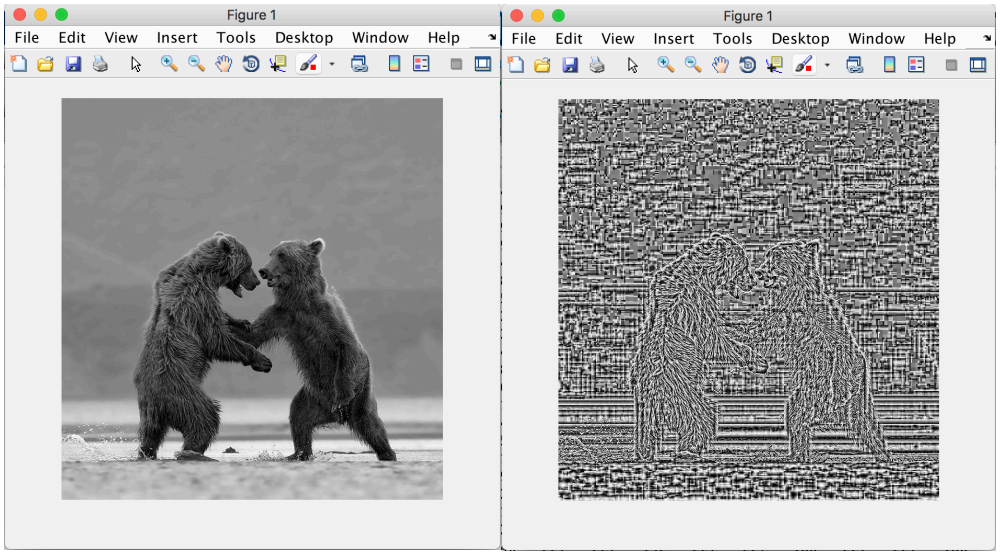
3x3 window size

```
>> m1 = imread('bear.jpg');
>> f = @ histeq;
>> m2 = rgb2gray(m1);
>> m3 = blkproc(m2, [3 3], f);
>> imshow(m3);
```



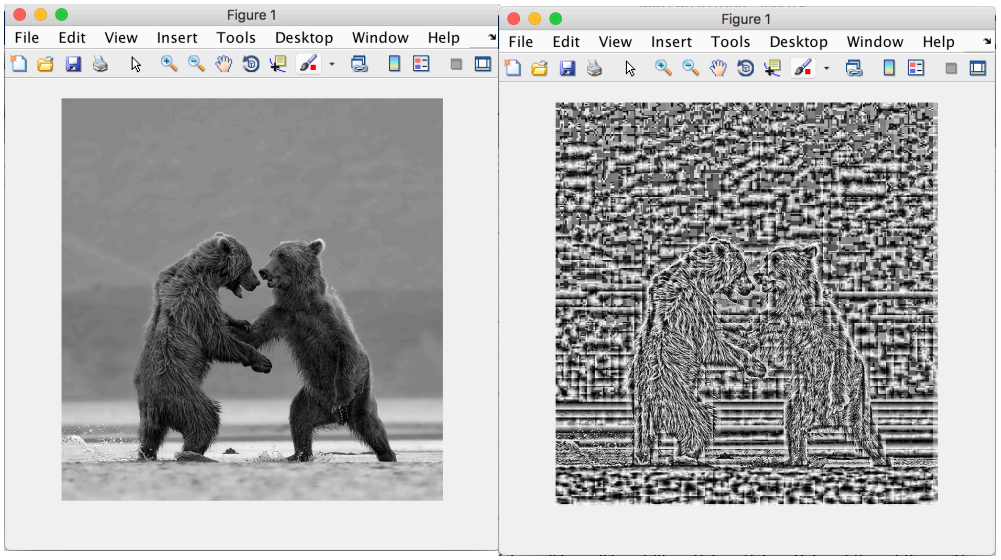
9x9 window size

```
>> m4 = blkproc(m2, [9 9], f);
>> imshow(m4);
```



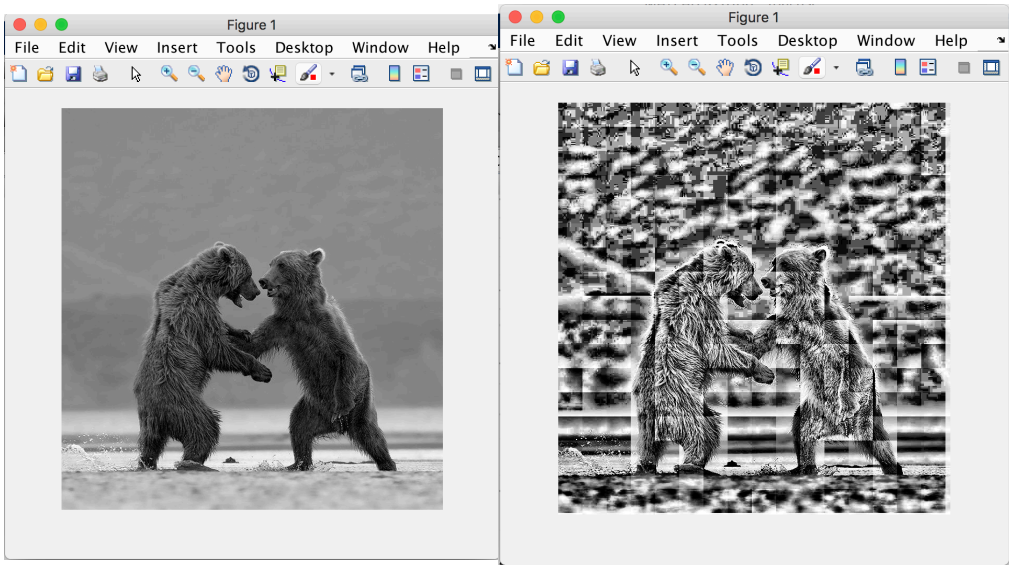
15x15 window size

```
>> m5 = blkproc(m2, [15 15], f);
>> imshow(m5);
```



45x45 window size

```
>> m6 = blkproc(m2, [45 45], f);
>> imshow(m6);
```

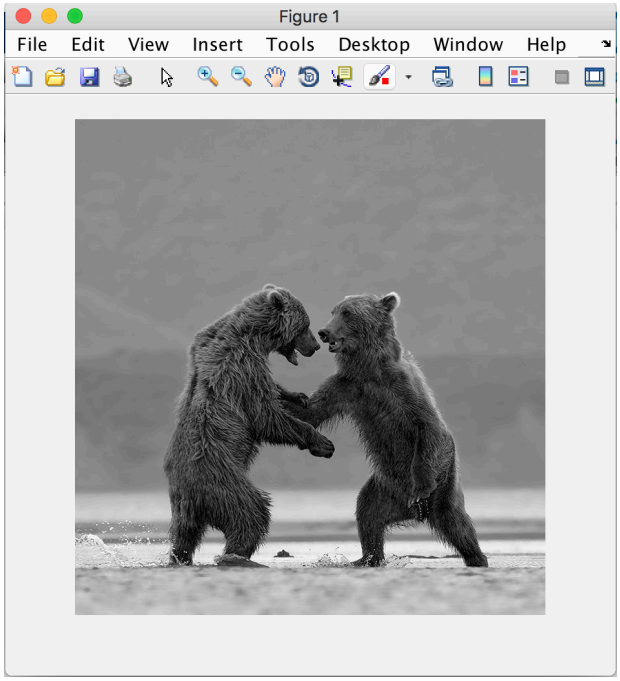


Histogram-based segmentation

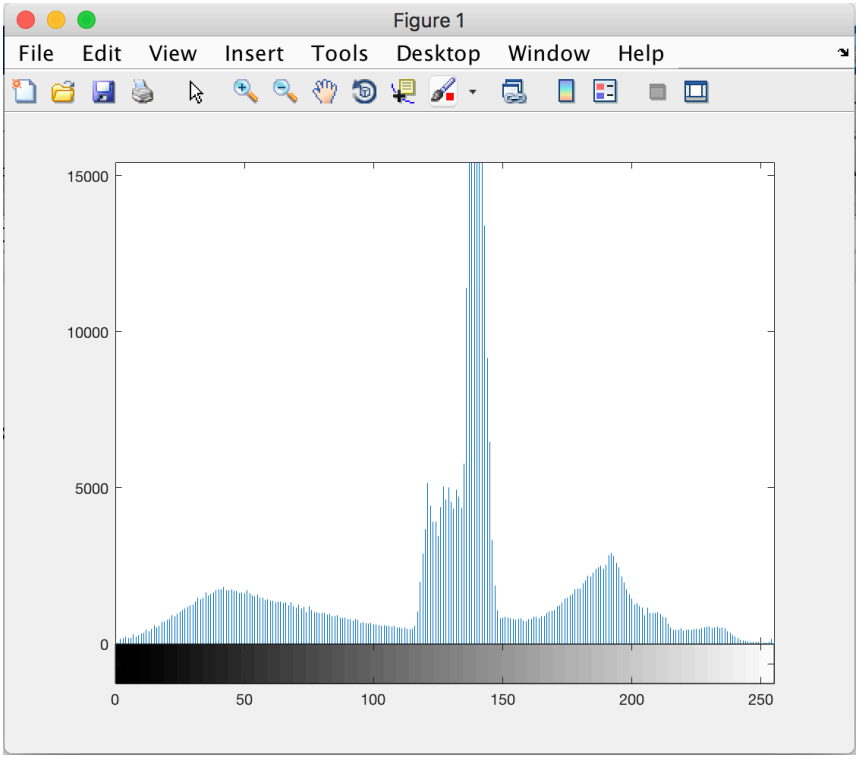
Implement histogram based segmentation on your image as follows:

a) Show your image.

```
>> m = imread(bear.jpg');
>> m2 = rgb2gray(m);
>> imshow(m2);
```



b) Display the histogram and identify the peaks of your histogram with the "objects" that they correspond to.




```
>> imhist(m2);
```

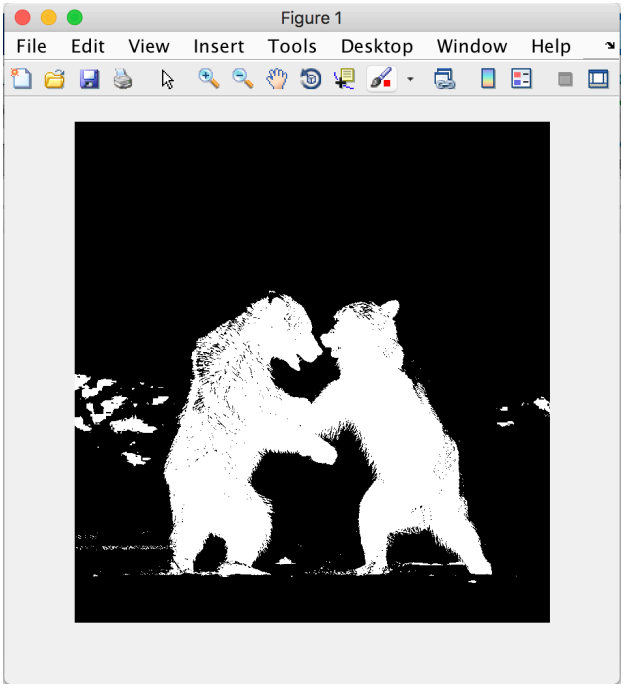
As shown on the histogram, there are three peaks. The first peak, between 0 and 100, indicate the bears in the picture. The second peak, between 100 and 150, indicates the background mountain and the third peak, between 150 and 210, indicates the river and the ground.

c) Specify the ranges that you will use to identify the binary objects.

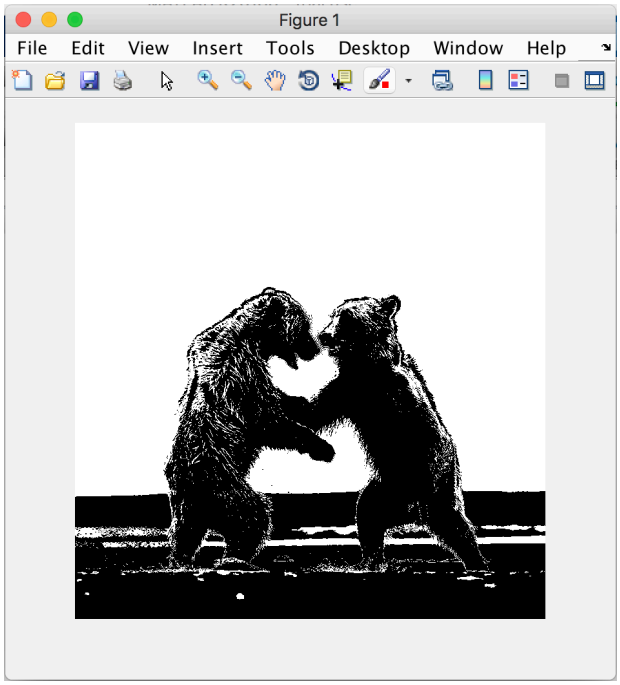
According to the histogram, I would choose to use the range between 0 and 100 to identify the bears. Also using the range between 100 and 150 can identify the background and the range between 150 to 255 can identify the river and the ground.

d) Show the identified objects as binary images for each range. (Remember to scale the images for display so that objects can be seen.)

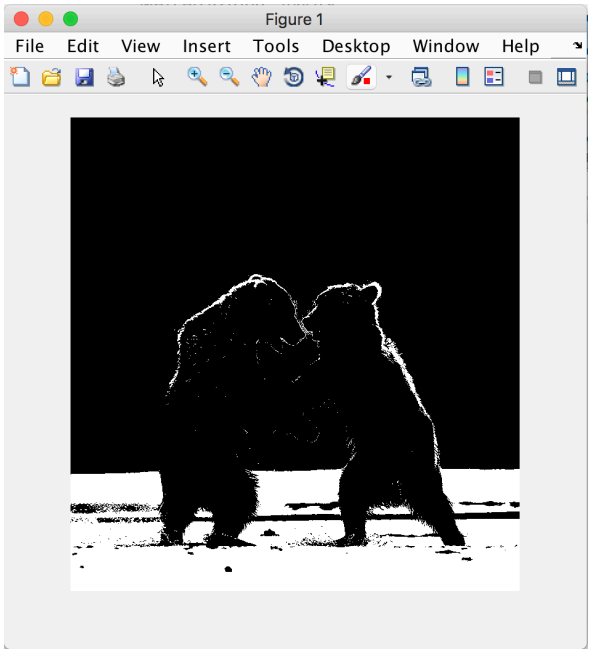
```
>> m3 = zeros(size(m2));
>> m3(m2<100) = 1;
>> imshow(m3)
```




```
>> m4 = zeros(size(m2));
>> m4(m2>=100 & m2<150) = 1;
>> imshow(m4);
```

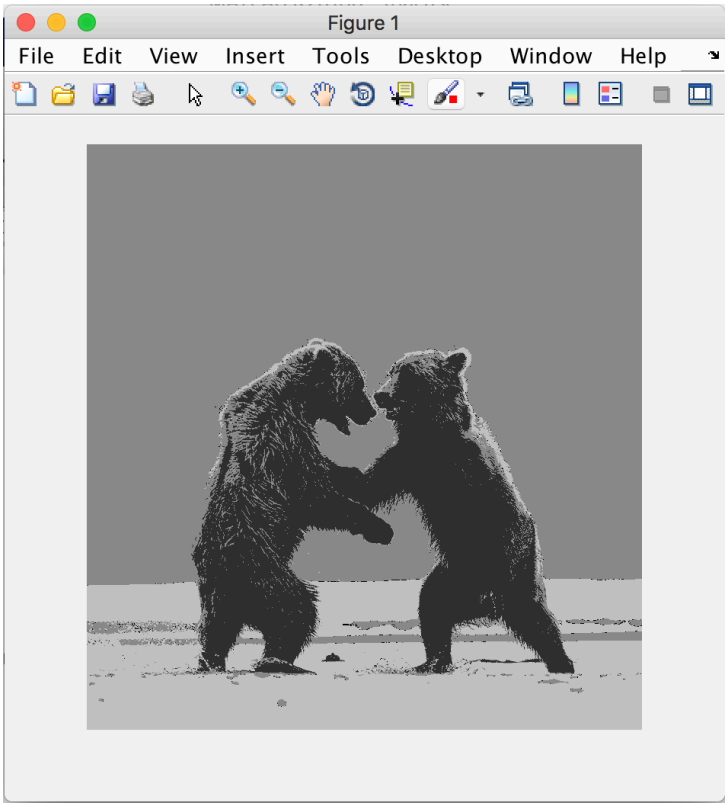


```
>> m5 = zeros(size(m2));
>> m5(m2>150 & m2<=255) = 1;
>> imshow(m5);
```



e) Finally construct the histogram-based segmented image, by combining the binary images.

```
>> mean1 = sum(sum(m3.*m2))/sum(sum(m3));
>> mean2 = sum(sum(m4.*m2))/sum(sum(m4));
>> mean3 = sum(sum(m5.*m2))/sum(sum(m5));
>> C = mean1*m3 + mean2*m4 + mean3*m5;
>> imshow(C);
```



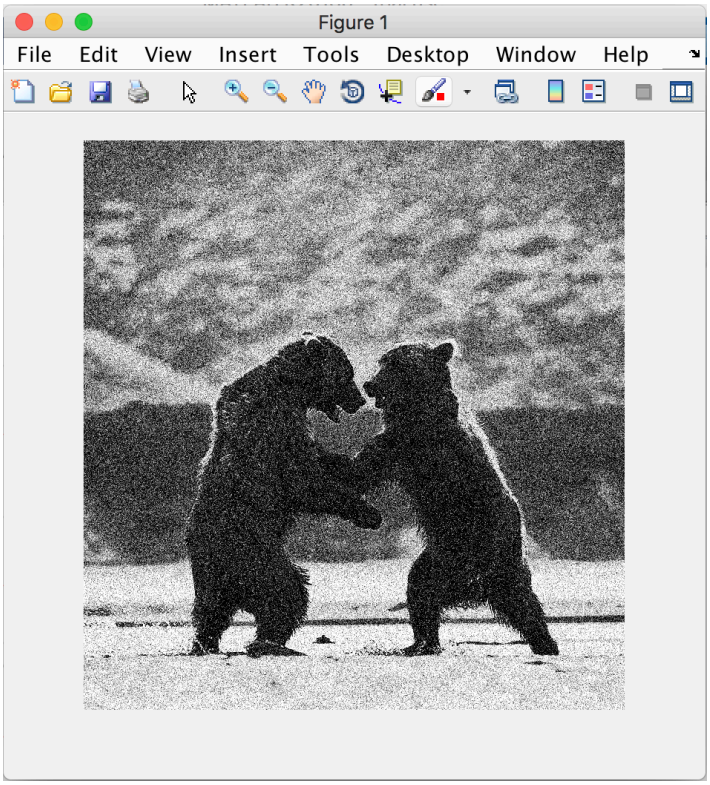
Histogram equalization, interpolation and noise reduction

a) **On the histogram equalized image from the first problem, use the imnoise function to generate two noise corrupted images as follows:**

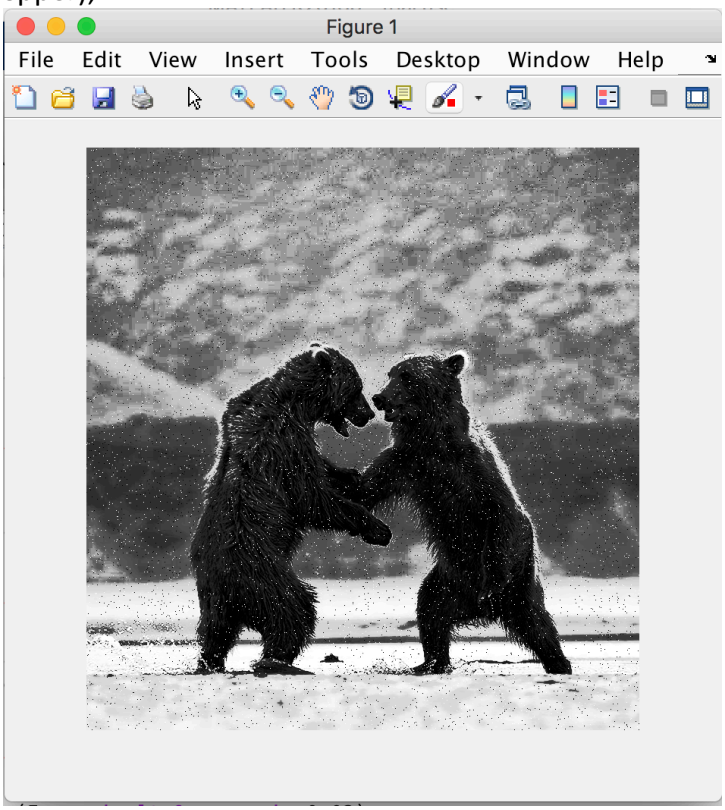
noise_gaussian =imnoise(Image,'gaussian',0,0.05);

noise_saltAndpepper=imnoise(Image,'salt & pepper', 0.02);

>> imshow(noise_gaussian);

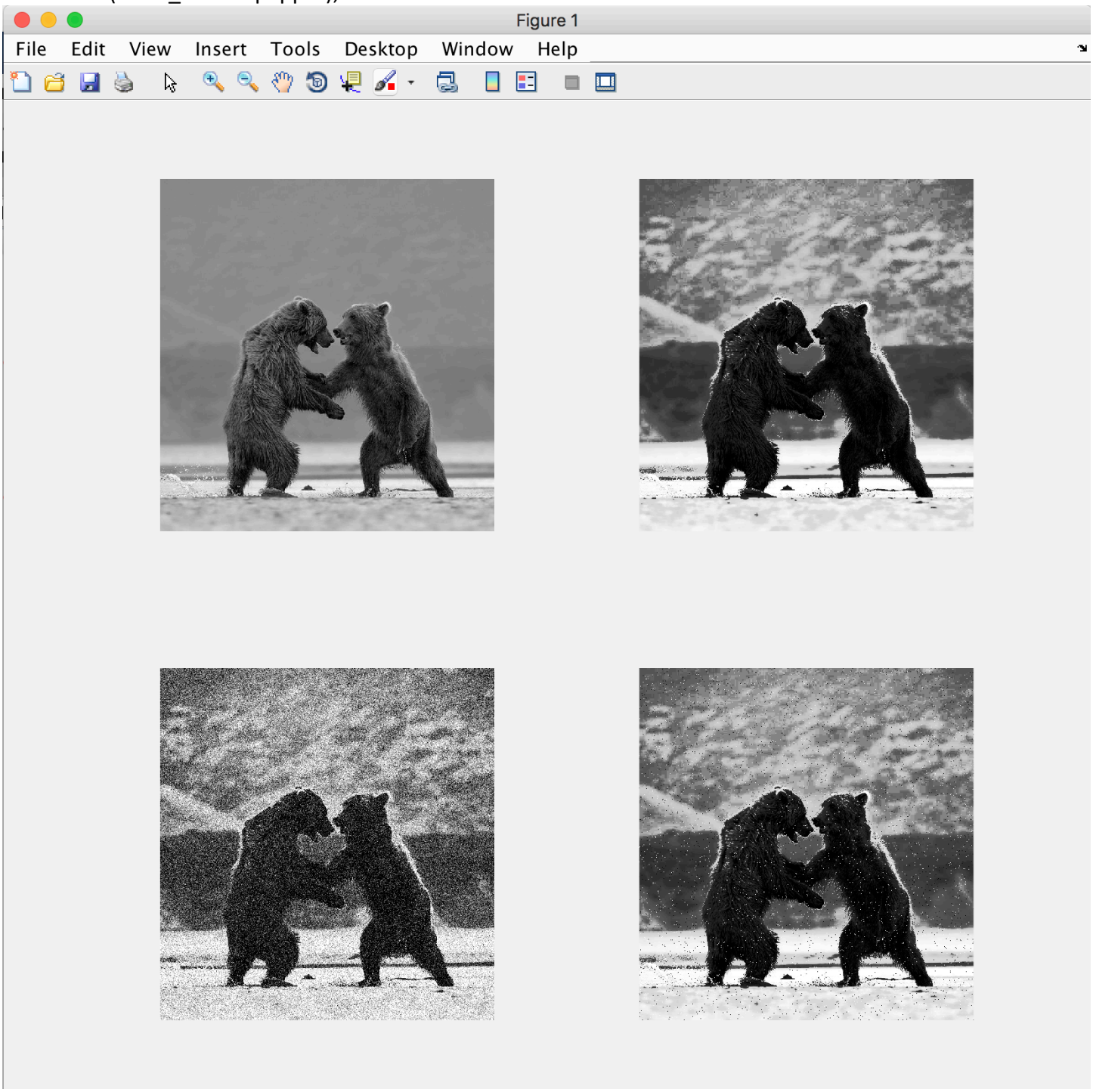


>> imshow(noise_ saltAndpepper);



b) Use subplot to display the original image, the histogram enhanced image, and the two noise corrupted images.

```
>> subplot(2, 2, 1);
>> imshow(m2);
>> subplot(2, 2, 2);
>> imshow(Image);
>> subplot(2, 2, 3);
>> imshow(noise_gaussian);
>> subplot(2, 2, 4);
>> imshow(noise_saltAndpepper);
```



c) Use the function fspecial to design averaging filters of size (3x3), (5,5), and (7x7). Use subplot to display the noise_saltAndpepper image and the three averaged filtered results. Do the same for the noise_gaussian image.

Here is my program,

```
function y = average( inimg )
h1 = fspecial('average', [3 3]);
h2 = fspecial('average', [5 5]);
h3 = fspecial('average', [7 7]);

f1 = imfilter(inimg, h1);
f2 = imfilter(inimg, h2);
f3 = imfilter(inimg, h3);

subplot(2, 2, 1);
imshow(inimg);
title('original');
subplot(2, 2, 2);
imshow(f1);
title('3x3 window');
subplot(2, 2, 3);
imshow(f2);
title('5x5 window');
subplot(2, 2, 4);
imshow(f3);
title('7x7 window');

end
```

```
>> average(noise_saltAndpepper);
```

```
>> average(noise_gaussian);
```

d) Use the medfilt2 function to perform median filtering on the noise_saltAndpepper image. Design the median filters to work with window sizes of (3x3), (5x5), and (7x7). Use your filters on the noise_gaussian image also and display as in part c).

Here is my program,

```
function y = median( inimg )
f1 = medfilt2 (inimg, [3 3]);
f2 = medfilt2 (inimg, [5 5]);
f3 = medfilt2 (inimg, [7 7]);

subplot(2, 2, 1);
imshow(inimg);
title('original');
subplot(2, 2, 2);
imshow(f1);
title('3x3 window');
subplot(2, 2, 3);
imshow(f2);
title('5x5 window');
subplot(2, 2, 4);
imshow(f3);
title('7x7 window');

end
```

```
>> median(noise_saltAndpepper);
```

```
>> median(noise_gaussian);
```