Ray Li
CSC 481
Dr. Furst
9/29/16

3.4 Propose a set of intensity-slicing transformations capable of producing all the individual bit planes of an 8-bit monochrome image. (For example, a transformation function with the property T(r) = 0 for r in the range [0, 127], and T(r) = 255 for r in the range [128, 255] produces an image of the 8th bit plane in an 8-bit image.)

Monochrome image means an image that has pixel values of either 0 or 255. For the 8th bit plane, we have two intervals which each has 128 values. For 7th bit plane, we have four intervals which each has 64 (2^6) values and for the 6th bit plane, we have eight intervals which each has 32 (2^5) values and so forth. Then for the 1st bit plane, we have 256 intervals which each has 1 (2^0) value. To see how the transformation function works on different intervals, we use the 7th bit plane as an example as shown below,

$$T(r) = \begin{cases} 0 & ,r \in [0,63] \; and \; [128,191] \\ 255 & ,r \in [64,127] \; and \; [192,256] \end{cases}$$

or for the 6th bit plane,

$$T(r) = \begin{cases} 0 & ,r \in [0,31] \; and \; [64,95] and \; [128,159] and \; [192,223] \\ 255 & ,r \in [32,63] \; and \; [96,127] \; and \; [160,191] and \; [224,255] \end{cases}$$

We can see that odd order intervals always get transformed into value 255 and even order intervals always get transformed into value 0. Therefore, we can write out this function, using MOD operation, as following,

$$s = \; floor(\frac{r}{2^{l-1}} \; MOD \; 2) \times 255$$

where s is the transformed pixel value, r is the input pixel value.

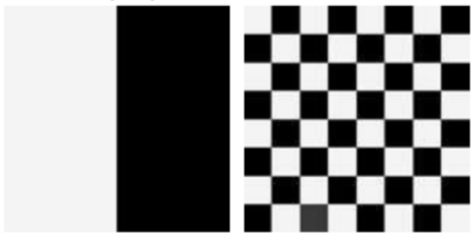
3.5 (a) What effect would setting to zero the lower-order bit planes have on the histogram of an image in general?

Setting to zero the lower-order bit planes would decrease the number of pixel values meaning since we zero out certain lower-bit plane pixel values, for example, we only keep 0, 8, 16, etc., we set 1 through 7 and 9 through 15 to zero. On the histogram, since we zero out certain pixel values, we will see wider gaps among histogram bars, which are sparser, and many bars will increase because the number of total pixel does not change. This means the loss of some high frequency details.

(b) What would be the effect on the histogram if we set to zero the higher-order bit planes instead?

By removing the high order bit planes, we will see narrower distribution and higher bars on the histogram. For example, if we set 7, 8, 15, 16, etc. to zero, the count for lower bit plane pixel values will increase and the image will become much darker and some low frequency components will be lost.

3.14 The images shown on the next page are quite different, but their histograms are the same. Suppose that each image is blurred with a 3 * 3 averaging mask.



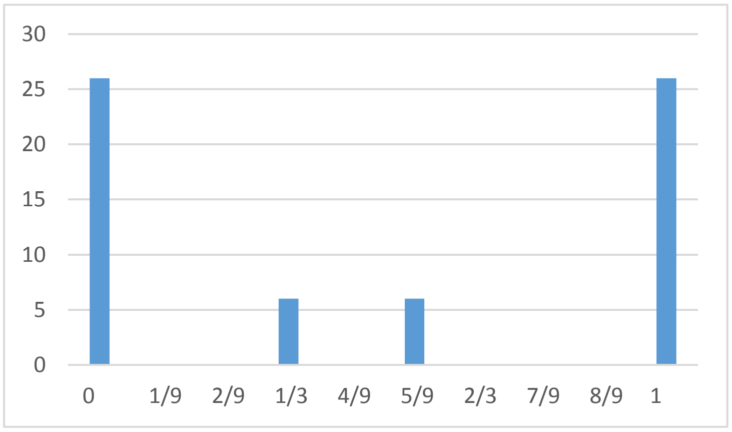
(a) Would the histograms of the blurred images still be equal? Explain.

The histograms will not be equal. On the left, when we slide the 3x3 averaging mask across the image, the pixel value will not change until it gets to the area where it separates the two colors, black and white. Therefore, the new pixel value that is going to be created through averaging only comes from averaging two columns of pixels in the middle of the image. In the second image, the new pixel value that is going to be created can come from the any 3x3 pixel matrices that have both white and black pixels, which are basically everywhere in the second image. Therefore, the count of that new pixel value is much higher than the one in the image on the left.

(b) If your answer is no, sketch the two histograms.

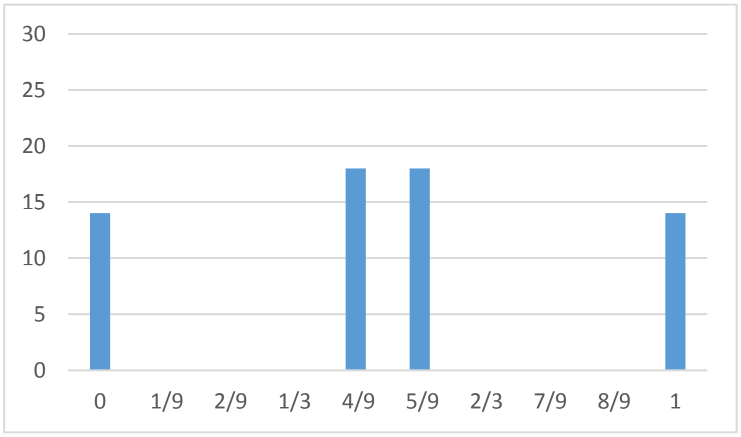
Assuming that we do not average the border pixel values (leave them untouched) and both images are size of 8x8. For the image on the left, after applying the averaging mask, we will have the following histogram

Value	Count
0	26
1/9	0
2/9	0
1/3	6
4/9	0
5/9	6
2/3	0
7/9	0
8/9	0
1	26



When we do the same to the image on the right with the same assumption, we will have the following result,

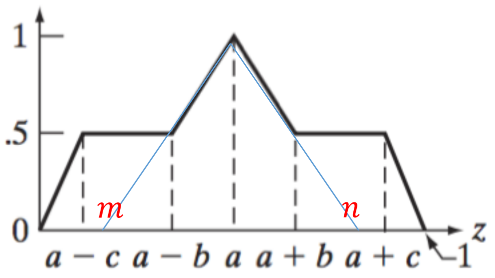
Value	Count
0	14
1/9	0
2/9	0
1/3	0
4/9	18
5/9	18
2/3	0
7/9	0
8/9	0
1	14



3.17 Discuss the limiting effect of repeatedly applying a 3 * 3 lowpass spatial filter to a digital image. You may ignore border effects.

Lowpass spatial filters are also referred as averaging filter, which removes some of the higher frequency features. If a 3x3 lowpass spatial filter is repeatedly applying to a digital image, assuming the boarder pixels remain untouched, it will zero out most of other non-zero pixels in the image because all the other pixels are repeatedly being averaged out with its eight neighbors. Therefore, the limiting effect of repeatedly applying a 3x3 lowpass spatial filter to a digital image is that it will eventually set pixel values to zero or nearly zero.

3.32 Use the fuzzy set definitions in Section 3.8.2 and the basic membership functions in Fig. 3.46 to form the membership functions shown below.



★ (a)

In this graph, we can interpret it as the union of two trapezoids and a triangle. We extend the lines for the triangular part so it will allow us to calculate the slope of those two lines. As shown above, we added to intersection points with the horizontal axes, marked as m and n. Therefore, we can form the membership function as:

$$\mu(\mathcal{Z}) = \begin{cases} 0.5-(a-c-z)/(a-c) & 0 \le z \le a-c \\ 1-(a-z)/(a-m) & a-b < z \le a \\ 1-(z-a)/(n-a) & a < z \le a+b \\ 0.5-[z-(a+c)]/[1-(a+c)] & a+c < z \le 1 \\ 0.5 & a-c \le z < a-b \text{ and } a+b \le z < a+c \end{cases}$$

Programming Problems

Problem 1: *Zooming/Oversampling* and *Shrinking/Subsampling Images* by Pixel Replication (15 points)

Write your own function capable of shrinking and zooming an image by pixel replication and decimation. Assume that the desired zoom/shrink factors will be the inputs for your function and will have integer values: a negative input means shrink and a positive input means expand. Do not use MatLab built in functions for decimation and replication, although you can use other MatLab functions.

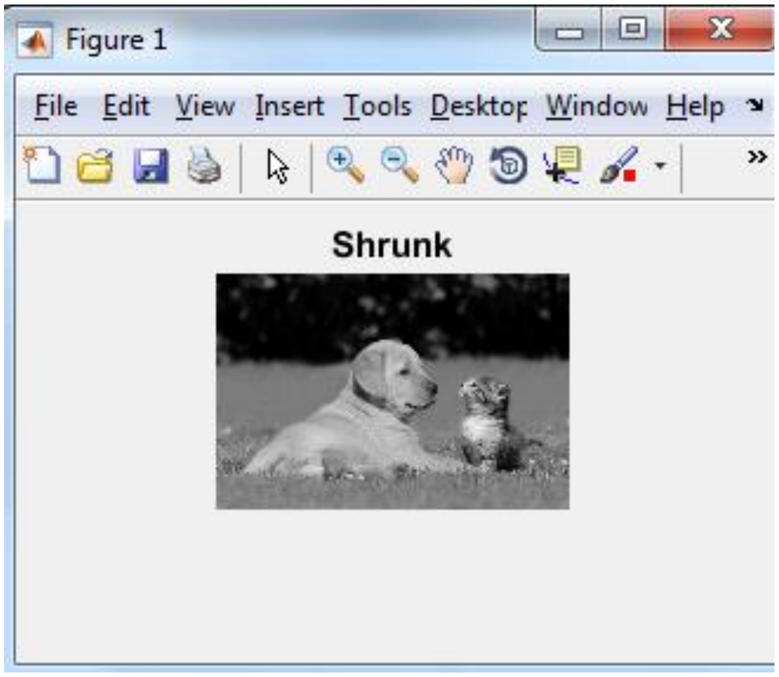
(a) Use your program to shrink an image by a factor of 4 in each dimension. Show the shrunk image.

Here is my program:

```
function [ B ] = questionOne( A, alpha )
[m, n] = size(A); %assuming gray scale
if alpha < 0 %shrinking
    f = abs(alpha);
    B = A([1:f:end], [1:f:end]);
imshow(B);
title('Shrunk');
elseif alpha > 0 %zooming
    exp = [];
    for i = 1:n
        exp=cat(2, exp, ones(1, alpha)*i);
    end
    B = A(:, exp);
    exp = [];
    for i = 1:m
        exp=cat(2, exp, ones(1, alpha)*i);
    end
    B = B(exp, :);
imshow(B);
title('Zoomed');
end
end
```

Here is the input picture and output picture that has been shrunk by a factor of 4

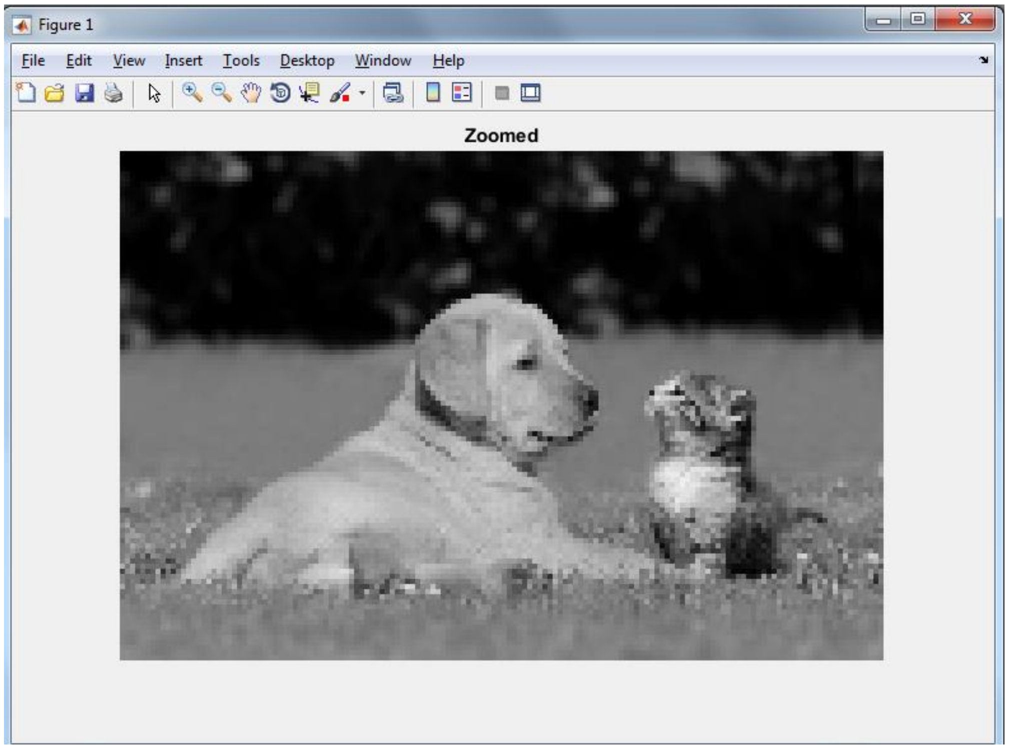
>> out = questionOne(B, -4);



*I resized it manually because it is not showing up well in the document since it has a size of 100x150 which is really small.

(b) Use your program to zoom the image back to its original size. Show the zoomed image and explain how and why the original image and the shrunk/zoomed images are different.

```
>> back = questionOne(out, 4);
```

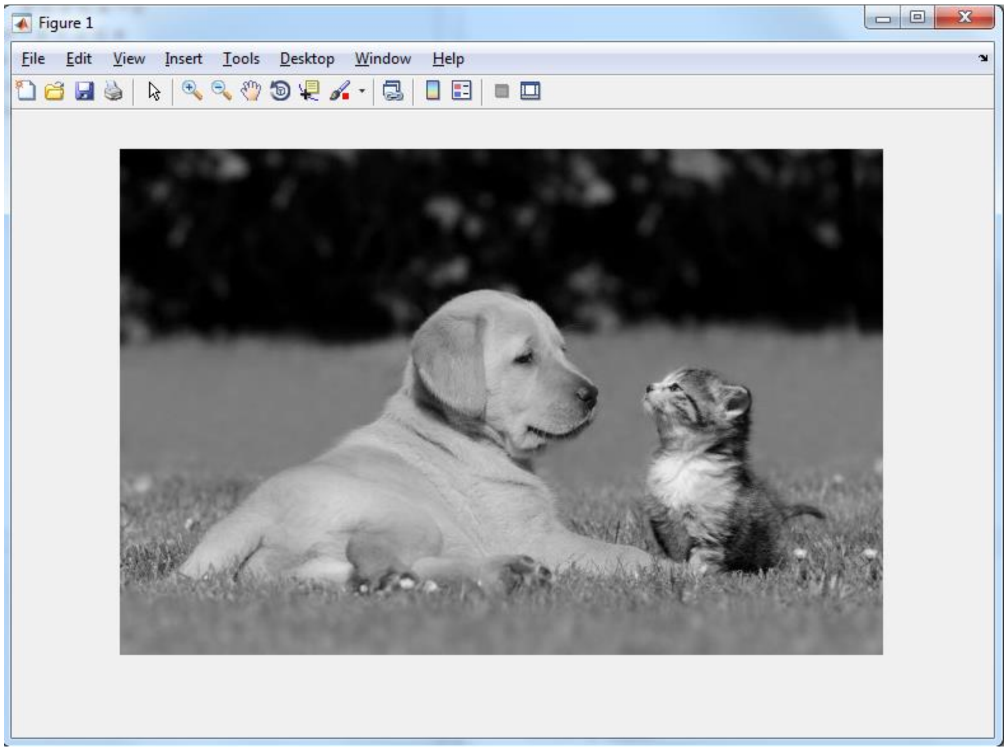


When the image was shrunk, we used the pixel decimation, which basically delete a certain number of rows and columns. Those pixel information was lost when the pixel decimation operation was done. Therefore, when we zoomed the image back, even though we use the pixel replication, we were replicating the left-over pixels after pixel decimation, so the shrunk-and-zoomed-back image is different from the original image.

Problem 2: Basic Grey Level Transformations (5 points)

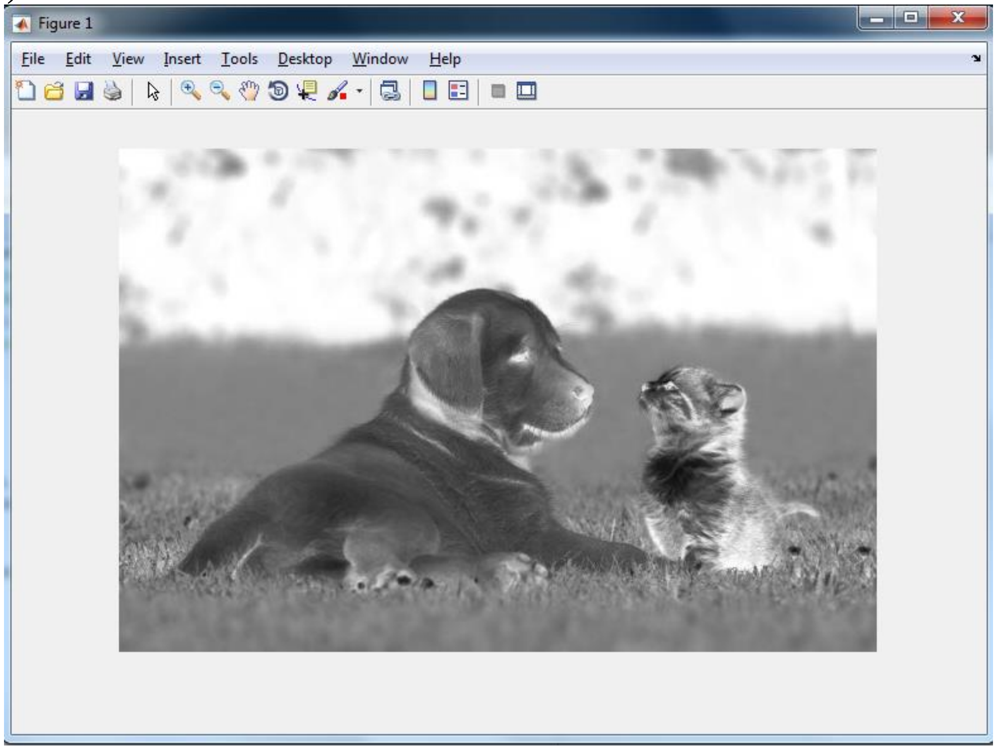
(a) Read and display an image.

```
>> A = imread('DC.jpg');
>> B = rgb2gray(A);
>> imshow(B)
```



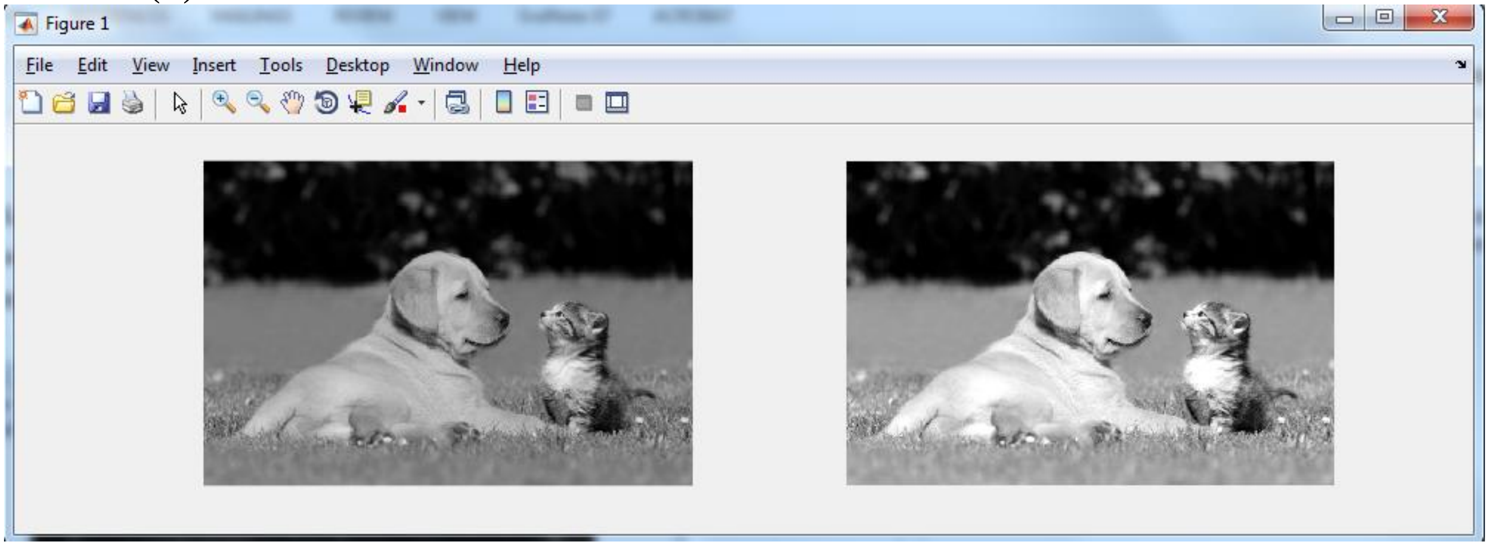
(b) Calculate the negative of the image and display it.

```
>> C = imadjust(B, [0 1],[1 0]);
>> imshow(C)
```



(c) Perform contrast stretching using the contrast stretching technique in Section 3.2.4. You can use the imadajust.m function to perform the image transformation.

```
>> D = imadjust(B, stretchlim(B),[]);
>> subplot(1, 2, 1)
>> imshow(B)
>> subplot(1, 2, 2)
>> imshow(D)
```

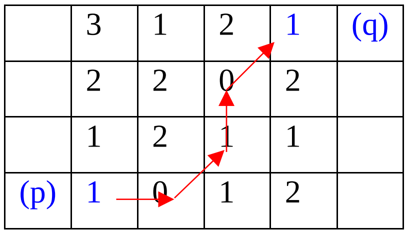


As shown above, the contract stretched picture on the right is brighter and shows more details of the objects in the image.

Work by hand:

Some basic relationships between pixels (5 points) Note: You do not need to use Matlab code to solve this problem.

Consider the image segment shown below (the values in blue represent the p and q pixels):



	3	1	2	1	(q)
	2	2	0	2	
	1	2	1	1	
(p)	1	0	1	2	

(a) Let V= {0,1} and compute the lengths of the shortest 4-, 8- paths between p and q. If a particular path does not exist between these two points, explain why.

4-path does not exist between p and q because none of the four neighbors of (q) have a value of either 0 or 1.

The shortest 8-path from p to q is shown in the picture marked in red and the length is 4.

(b) Calculate the D_4 distance (city-block distance) and the D_8 distance (chessboard distance) between pixels p and q. Do these two distances depend on which path you choose between p and q? Explain your answer.

D_3 distance is between p(0,0) and q(3, 3), so it can be calculated as

$$D_4(p,q) = |3-0| + |3-0| = 6$$

D_8 distance is between p(0,0) and q(3, 3), so it can be calculated as

$$D_8(p,q) = \max(|3-0|, |3-0|) = 3$$

These two distances do not depend on which path I choose because in part (a), we consider the pixel values of the neighbors of both (q) and (p), or the adjacency of (q) and (p). However, in this question, we only consider the coordinates of these pixels, which are used to calculate the distances.