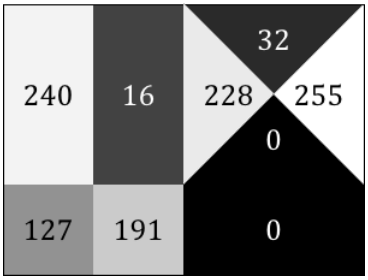
EECE-5626 (IP&PR) : Homework-2 Solutions

Table of Contents

Problem-1: DIP4E Problem 3.7 (Page 240)

Obtain the unnormalized and the normalized histograms of the following 8-bit, $M \times N$ image. Give your histogram either in a table or a graph, labeling clearly the value and location of each histogram component in terms of M and N. Doublecheck your answer by making sure that the histogram components add to the correct value.



Solution: Total number of pixels in the $\text{image} = MN$. The division in the vertical axis (M) are \$M/3\$. The divisions in the horizontal axis (N) are $N/4$. Therefore, the number of pixels in the smallest square is $(M/3)(N/4) = MN/12$. So, for example, the number of pixels having value 127 is $MN/12$, the number of pixels having value 16 is $MN/6$, and the number of pixels having value 0 is $MN/4$. For the unnormalized histogram, the number of counts in each bin are equal to the values just computed. For the normalized histogram we divide each component by MN. Computation of the other components of the histogram follows the same line of reasoning. The histogram components are listed in the following table.

Intensity value Value	Number of Pixels	Unnormalized Hist Components (Bins)	Normalized Hist Components (Bins)
0	$MN/4$	$MN/4$	1/4
16	$MN/6$	$MN/6$	1/6
32	$MN/12$	$MN/12$	1/12
127	$MN/12$	$MN/12$	1/12
191	$MN/12$	$MN/12$	1/12
228	$MN/12$	$MN/12$	1/12
240	$MN/6$	$MN/6$	1/6
255	$MN/12$	$MN/12$	1/12

Problem-2: DIP4E Problem 3.9 (Page 240)

Suppose that the digital image is subject to histogram equalization. Show that a second pass of histogram equalization (on the histogram-equalized image) will produce exactly the same result as the first pass.

Solution: Let MN be the total number of pixels and let n_{r_ℓ} be the number of pixels in the input image with intensity value r_ℓ. Then the histogram equalization transformation is

$$s_k = T(r_k) = \sum_{\ell=0}^{k} \frac{n_{r_\ell}}{MN} = \frac{1}{MN} \sum_{\ell=0}^{k} n_{r_\ell}.$$

Now, every pixel (and no other) with value r_k is mapped to value s_k. Hence, we must have $n_{s_k} = n_{r_k}$. A second pass of histogram equalization would produce values according to the transformation

$$v_k = T(s_k) = \frac{1}{MN} \sum_{\ell=0}^{k} n_{s_\ell}.$$

But $n_{s_\ell} = n_{r_\ell}$, so

$$v_k = T(s_k) = \frac{1}{MN} \sum_{\ell=0}^{k} n_{r_\ell} = s_k$$

which proves that the second pass of histogram equalization would yield the same result as the first pass.

Problem-3: DIP4E Problem 3.13 (Page 240)

Assume continuous intensity values, and suppose that the intensity values of an image have the PDF $p_r(r) = 2r/(L-1)^2$ for $0 \leq r \leq L-1$, and $p_r(r) = 0$ for other values of r.

(a) Histogram Equalization

Find the transformation function that will map the input intensity values, r, into values, s, of a histogram equalized image.

Solution: The: histogram equalization transformation for the interval $[0, L-1]$:

$$s = T(r) = (L-1)\int_0^r p_r(w)\mathrm{d}w = \frac{2}{L-1}\int_0^r w\,\mathrm{d}w = \frac{r^2}{L-1}.$$

By definition, the transformation is 0 for values outside the range $[0, L-1]$. Squaring the values of the input intensities and dividing them by $(L-1)$ will produce an image whose intensities, s, have a uniform PDF because this is a histogram-equalization transformation.

(b) Histogram Specification

Find the transformation function that (when applied to the histogram-equalized intensities, s) will produce an image whose intensity PDF is $p_r(z) = 3z^2/(L-1)^3$ for $0 \leq z \leq L-1$ and $p_z(z) = 0$ for other values of z.

Solution: We are interested in an image with a specified histogram, so we obtain next

$$G(z) = (L-1)\int_0^z p_z(w)\mathrm{d}w = \frac{3}{(L-1)^2}\int_0^r w^2\,\mathrm{d}w = \frac{z^3}{(L-1)^2}, \quad z \in [0, L-1]$$

and $G(z) = 0$ elsewhere by definition. Finally, we require that $G(z) = s$, but $G(z) = z^3/(L-1)^2$, so $z^3/(L-1)^2 = s$ and we have

$$z = G^{-1}(s) = \left[(L-1)^2 s\right]^{1/3}.$$

Thus, if we multiply every histogram equalized pixel by $(L-1)^2$ and raise the product to the power $1/3$, the result will be an image whose intensities, z, have the PDF $p_z(z) = 3z^2/(L-1)^3$ in the interval $[0, L-1]$ as desired.

(c) Overall Transformation

Express the transformation function from (b) directly in terms of r, the intensities of the input image.

Solution: Because $s = T(r) = r^2/(L-1)$ we can genente the z values directly from the intensities. r, of the input image:

$$z = G^{-1}\left[T(s)\right] = \left[(L-1)^2 s\right]^{1/3} = \left[(L-1)^2\frac{r^2}{(L-1)}\right]^{1/3} = \left[(L-1)r^2\right]^{1/3}.$$

Thus, squaring the value of each pixel in the original image, multiplying the result by $(L-1)$, and raising

the product to the power $1/3$ will yield In image whose intensity levels, z, will have me specified PDF.

Problem-4: DIP4E Problem 3.36 (Page 243)

An image is filtered with three Gaussian lowpass kernels of sizes 9×9, 13×13, and 25×25, and standard deviations 1.5, 2, and 4, respectively. A composite filter, w, is formed as the convolution of these three filters.

(a) Convolution of Gaussian filters

Is the resulting filter Gaussian? Explain.

Solution: Note that the size of each Gaussian filter array is $6\sigma \times 6\sigma$ or larger, where σ is the standard deviation, so that each array represents an almost perfect Gaussian function. Under this condition, the resulting filter obtained by convolving three Gaussian arrays is also Gaussian. This result stems from two well-known properties: 1. The Fourier transform of a Gaussian function is also a Gaussian function with reciprocal variance and 2. The product of two Gaussian functions is a Gaussian function. Now if we convolve two Gaussian functions, then in the frequency domain we have product of two Gaussian transform functions. But this product is also Gaussian and hence, its inverse is also Gaussian. This result is also summarized in Table 3.6 of DIP4E textbook and can be extended to more than two Gaussian arrays.

(b) Standard Deviation of the Composite Filter

What is the standard deviation of the composite filter w?

Solution: Using the extension of the results from Table 3.6 for the convolution of two Gaussians, the standard deviation of the resulting filter is

$$\sigma_w = \sqrt{\sigma_1^2 + \sigma_2^2 + \sigma_3^2} = \sqrt{1.5^2 + 2^2 + 4^2} = 4.717.$$

```
clc; clear; close all;
sigmaw = sqrt(1.5^2+2^2+4^2)
```

```
sigmaw = 4.7170
```

(c) SIze of the Composite Filter

What should be the ***appropriate*** size of the composite filter w?

Solution: When $M \times M$ size filter is convolved with $N \times N$ filter, the resulting composite filter has the size of $(M + N - 1) \times (M + N - 1)$. Using this result for two convolutions, the size of the composite filter, when obtained via convolutions, is

$$\big((9 + 13 - 1) + 25 - 1\big) \times \big((9 + 13 - 1) + 25 - 1\big) = 45 \times 45.$$

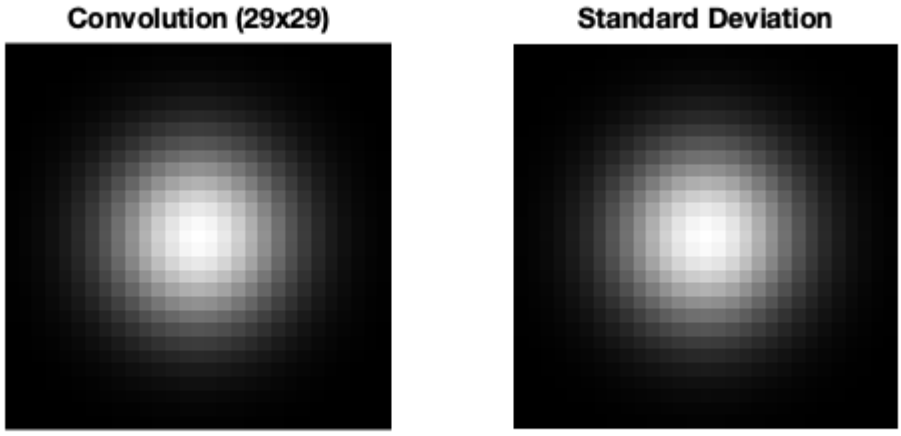
However, this size is too large based on the variance calculation in part (b). Since $\sigma_w = 4.717$, the minimum size needed is $6\sigma_w \times 6\sigma_w = 29 \times 29$. Thus, there is a significant reduction in computational complexity while filtering an image using 29×29 size Gaussian filter. The following calculations justify this argument.

```
h1 = fspecial('gaussian',9,1.5);
h2 = fspecial('gaussian',13,2);
h3 = fspecial('gaussian',25,4);
w1 = conv2(conv2(h1,h2),h3); % Composite filter using convolution
w1 = w1(9:37,9:37); % Middle 29x29 area
w2 = fspecial('gaussian',29,sigmaw); % Composite filter using standard deviation
maxDifference = max(abs(w1(:)-w2(:)));
fprintf('Maximum Difference = %g',maxDifference);
```

```
Maximum Difference = 2.02757e-05
```

The difference between the two equal-size filters is very small. It is not zero because of the normalization done in **w1** over its original larger 45×45 size while a different normalization is done in **w2** over its smaller 29×29 size. The following display shows the visual equivalence.

```
figure('units','inches',"Position",[0,0,8,4]);
subplot(1,2,1); imshow(w1,[]); title('Convolution (29x29)','FontSize',14);
subplot(1,2,2); imshow(w2,[]); title('Standard Deviation','FontSize',14);
```



Clearly, 29×29 size of the composite filter w is appropriate.

Problem-5: DIP4E Project 3.7 (Page 247)

Lowpass filtering

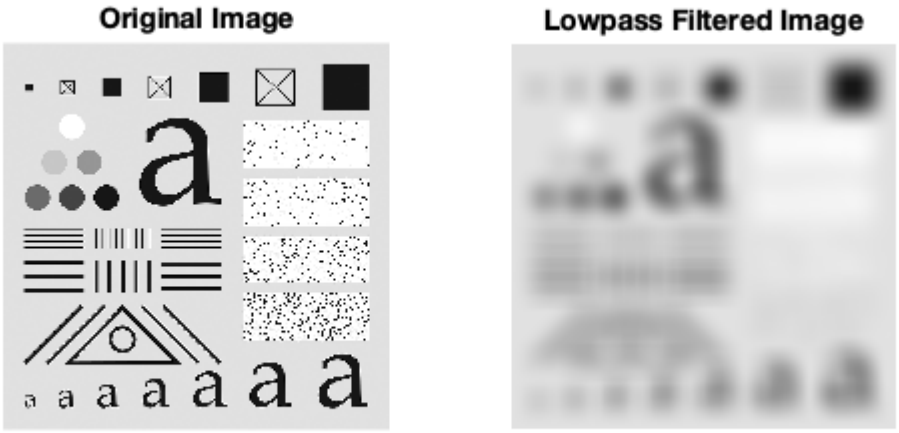
(a) Blurring of `testpattern1024` image.

Read the image **`testpattern1024.tif`** and lowpass filter it using a Gaussian kernel large enough to blurr the image so that the large letter "a" is barely readable, and the other letters are not.

Solution: MATLAB script:

```
clc; close all; clear;
```

```
f = imread('../artfiles/testpattern1024.tif');
w = gaussKernel4e(101,30); % after several trials (this is not a unique kernel)
g = twodConv4e(f,w);
figure('units','inches',"Position",[0,0,8,4]);
subplot(1,2,1); imshow(f); title('Original Image','FontSize',14);
subplot(1,2,2); imshow(g); title('Lowpass Filtered Image','FontSize',14);
```

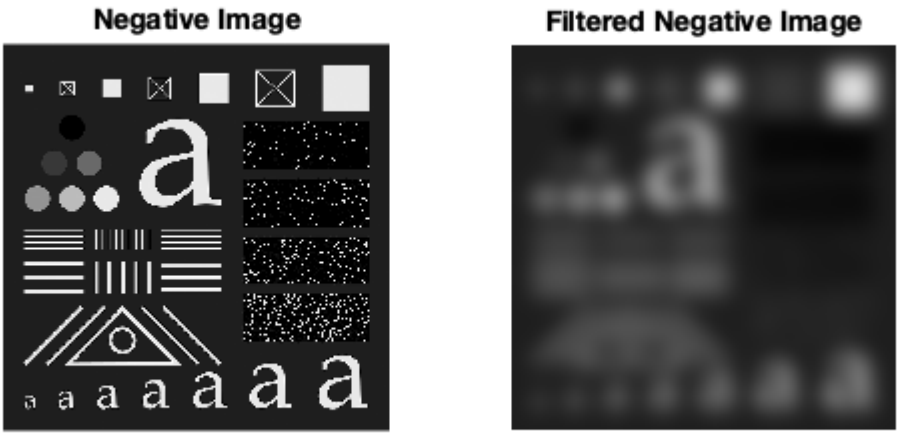


(b) Thresholding of `testpattern1024.tif` image.

Read the image **`testpattern1024.tif`**. Lowpass filter it using a Gaussian kernel of your specification so that, when thresholded, the filtered image contaions only part of the large square on the top, right.

Solution: MATLAB script:

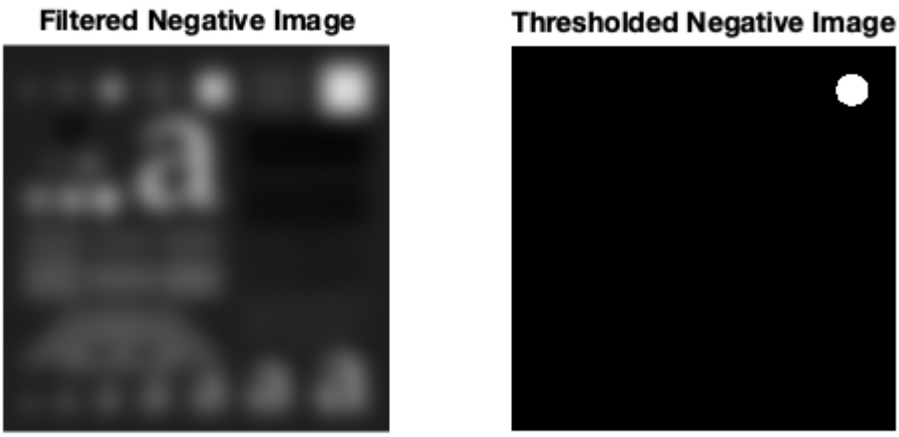
```
% We will work in the negative image, as suggested in the problem statement
f = intXform4e(f,'negative');
figure('units','inches',"Position",[0,0,8,4]);
subplot(1,2,1); imshow(f); title('Negative Image','FontSize',14);
% After several trials:
w = gaussKernel4e(201,30);
g = twodConv4e(f,w); % Filter the image
% Display the results and its maximum value
subplot(1,2,2); imshow(g); title('Filtered Negative Image','FontSize',14);
```




```
fprintf('Maximum value of g = %g',max(g(:)));
```

```
Maximum value of g = 0.848601
```

```
% Thereshold the image at a high value, about 80%, determined by
% experimentation
gT = g > 0.8*max(g(:));
figure('units','inches',"Position",[0,0,8,4]);
subplot(1,2,1); imshow(g); title('Filtered Negative Image','FontSize',14);
subplot(1,2,2); imshow(gT); title('Thresholded Negative Image','FontSize',14);
```



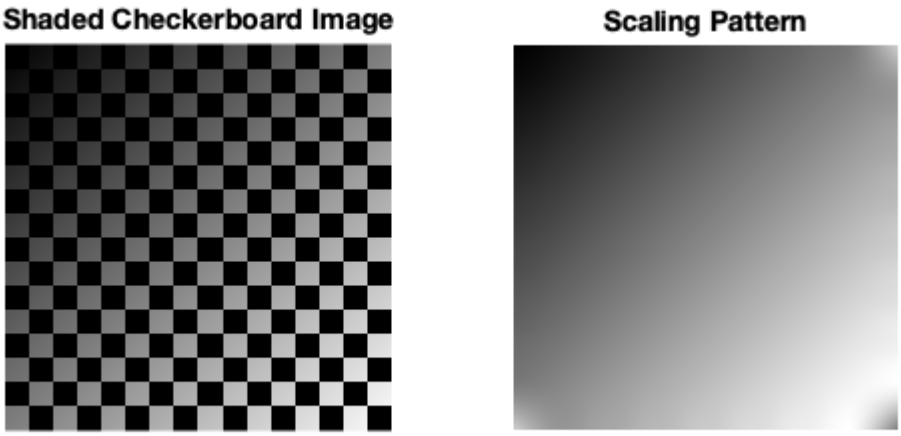
As expected, part of the object of interest was extracted, and nothing else is present in the image.

(c) Reproduce Example 3.18.

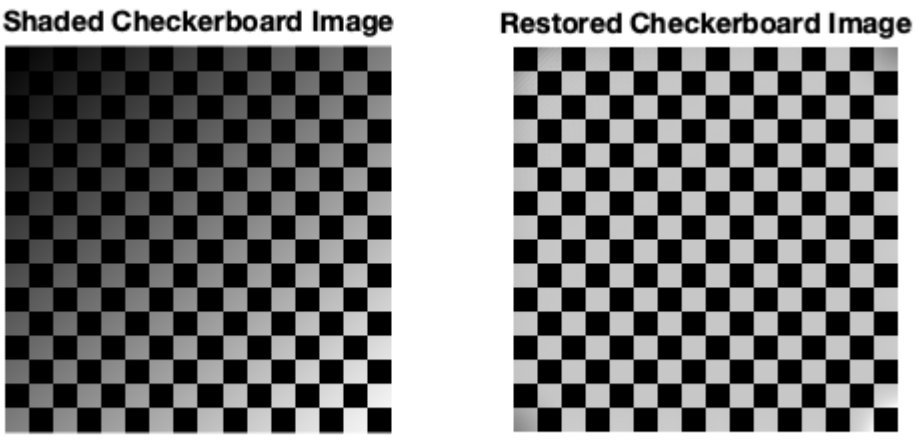
Read the image checkerboard1024-shaded.tif and reproduce the results in Example 3.18, keeping in mind that the above image is of size 1024×1024 pixels, so that the checkerboard squares are 64×64.

Solution: MATLAB script:

```
f = imread('../artfiles/checkerboard1024-shaded.tif');
f = intScaling4e(f); % Floating point in range [0,1]
figure('units','inches',"Position",[0,0,8,4]);
subplot(1,2,1); imshow(f); title('Shaded Checkerboard Image','FontSize',14);
% Define the kernel experimentally suitable for obtaining the shading
% pattern
w = gaussKernel4e(257,64);
% Filter f using w to obtain the shading pattern
g = twodConv4e(f,w);
% scale the shading pattern to the full [0,1] range for display:
subplot(1,2,2); imshow(intScaling4e(g,'full'));
title('Scaling Pattern','FontSize',14);
```




```
% Perform division to correct for the shading pattern:
fout = f./g;
% Scale the result to the full [0,1] range:
fout = intScaling4e(fout,'full');
figure('units','inches',"Position",[0,0,8,4]);
subplot(1,2,1); imshow(f); title('Shaded Checkerboard Image','FontSize',14);
subplot(1,2,2); imshow(fout); title('Restored Checkerboard Image','FontSize',14);
```



Thus, the results of Example 3.18 are reproduced.

Problem-6: DIP4E Project 3.10 (Page 248)

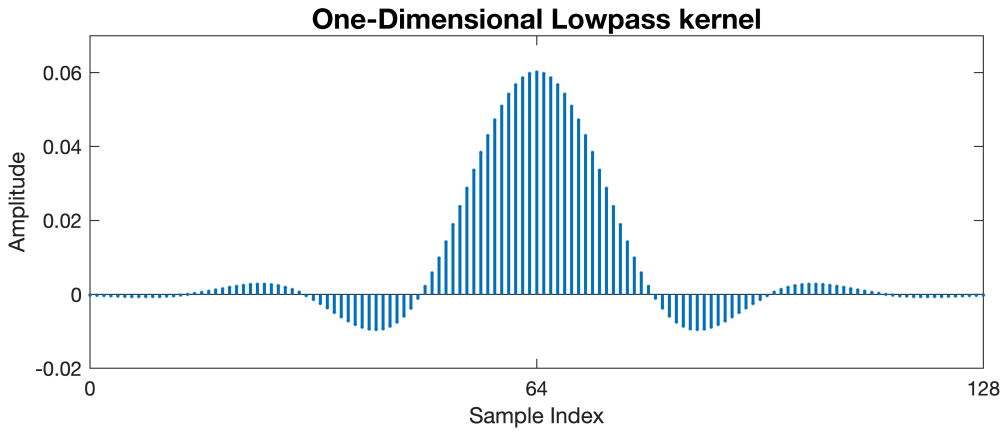
Using kernels generated by filter design software.

(a) 1-D Lowpass Filter

File **`lpkernel1D.txt`** is a comma-delimited text file containing a 129-coefficient, one-dimensional filter kernel similar to the kernel shown in Fig. 3.60(a). This kernel was generated using MATLAB's Signal processing Toolbox. Read this file and plot it.

Solution: MATLAB script.

```
clc; close all; clear;
lowpass = dlmread('../artfiles/lpkernel1D.txt'); % ASCII delimited file read
figure('Units',"inches","Position",[0,0,8,3]);
stem((0:128),lowpass,'filled','LineWidth',1.5,'MarkerSize',1); axis([0,128,-0.02,0.07]);
xlabel('Sample Index'); ylabel('Amplitude'); set(gca,'xtick',[0,64,128]);
title('One-Dimensional Lowpass kernel','FontSize',14);
```



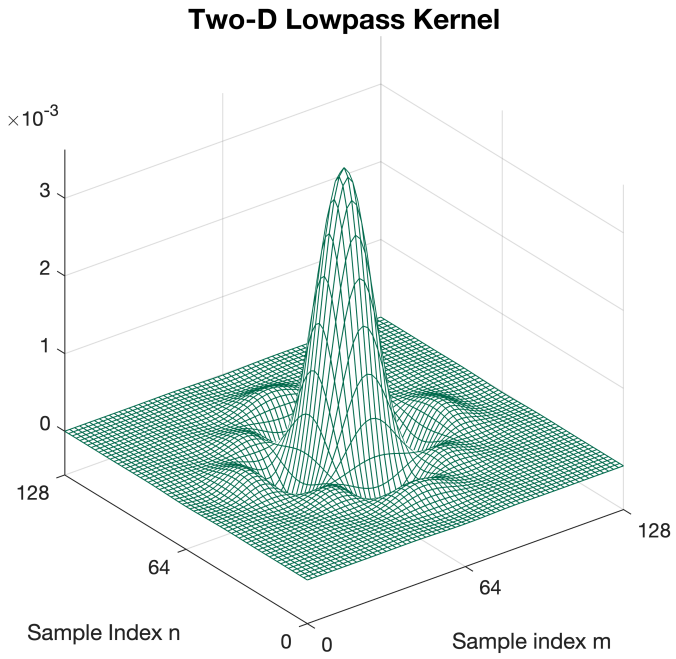
As explained in the text, this filter is an approximation to an ideal lowpass filter, so we should expect some ***ringing*** in the filtered image.

(b) 2-D Lowpass Filter

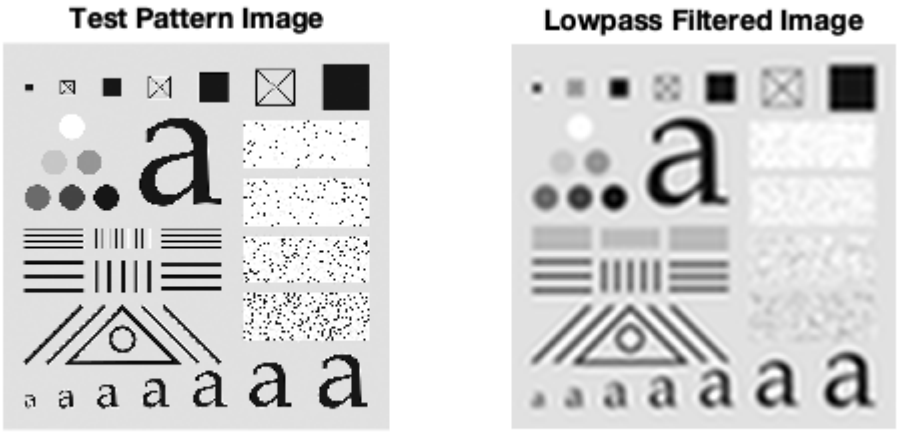
Construct a 2-D lowpass kernel from the data in (a) and filter the image **`testpattern1024.tif`** with it. Display your result.

Solution: MATLAB script.

```
f = imread('../artfiles/testpattern1024.tif');
lowpass2D = (lowpass')*lowpass;
figure('Units',"inches","Position",[0,0,5,5]);
lp2d = mesh(0:2:128,0:2:128,lowpass2D(1:2:end,1:2:end)); % every other sample is plotted
xlim([0,128]); ylim([0,128]);
lp2d.EdgeColor = [0,106,78]/255; axis tight;
xlabel('Sample index m'); ylabel('Sample Index n');
title('Two-D Lowpass Kernel','FontSize',14);
set(gca,'xtick',[0,64,128],'ytick',[0,64,128]);
```




```
glp = twodConv4e(f,lowpass2D); % Filter the image
figure('units','inches',"Position",[0,0,8,4]);
subplot(1,2,1); imshow(f); title('Test Pattern Image','FontSize',14);
subplot(1,2,2); imshow(glp); title('Lowpass Filtered Image','FontSize',14);
```



The ringing, mentioned in (a), is visible as halos around the sharp intensity transitions in the image. The halos are most visible around the characters at the bottom of the image.

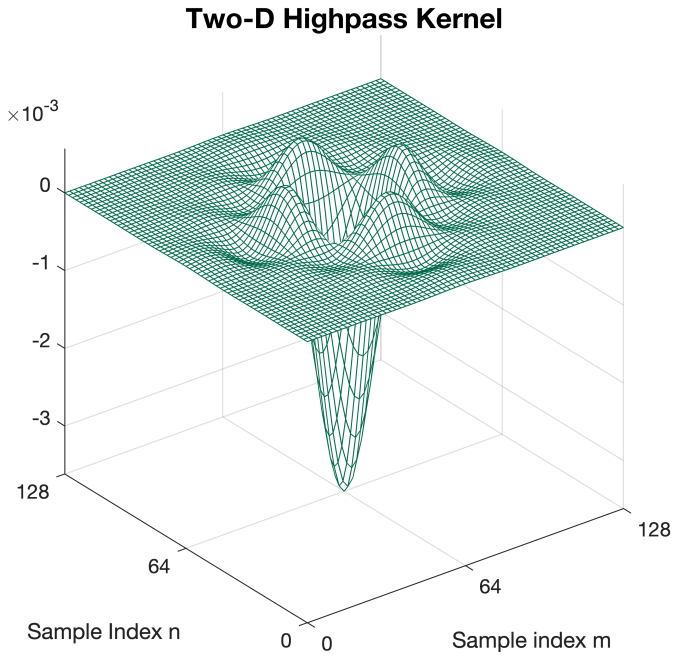
(c) Highpass filtering of `testpattern1024.tif` image.

Create a 2-D highpass filter kernel from the powpass kernel in (b) and filter the image **`testpattern1024.tif`** with it. Display your results.

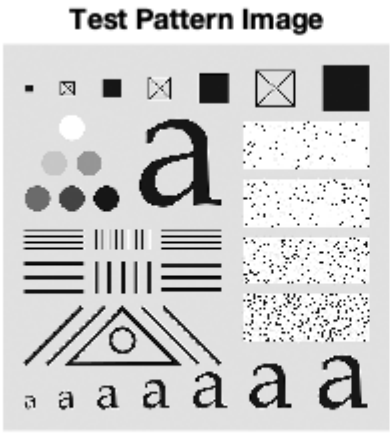
Solution: The highpass fiiter is formed from the lowpass fiiter by subtracting the lowpass fiiter from a 2-D unit impulse of the same dimensions.

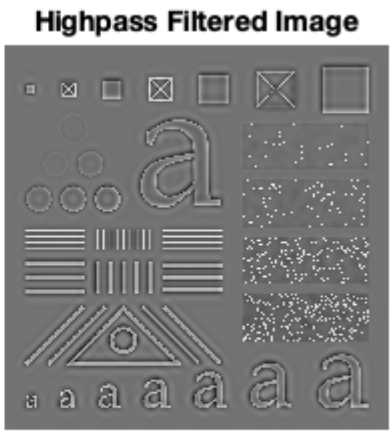
MATLAB script:

```
% Construct the impulse by generating an array of zeros first.
[M, N] = size(lowpass2D);
impulse = zeros(M,N);
% Insert a 1 in the center.
impulse(round(M/2) + 1,round(N/2) + 1) = 1;
% Construct the filter.
highpass2D = impulse - lowpass2D;
figure('Units',"inches","Position",[0,0,5,5]);
hp2d = mesh(0:2:128,0:2:128,highpass2D(1:2:end,1:2:end)); % every other sample is plotted
xlim([0,128]); ylim([0,128]);
hp2d.EdgeColor = [0,106,78]/255; axis tight;
xlabel('Sample index m'); ylabel('Sample Index n');
title('Two-D Highpass Kernel','FontSize',14);
set(gca,'xtick',[0,64,128],'ytick',[0,64,128]);
```




```
% Filter the image.
ghp = twodConv4e(f,highpass2D);
figure('units','inches',"Position",[0,0,8,4]);
subplot(1,2,1); imshow(f); title('Test Pattern Image','FontSize',14);
subplot(1,2,2); imshow(intXform4e(ghp,'negative'),[]); % Inverted image for better display
title('Highpass Filtered Image','FontSize',14);
```





As expected, the highpass result emphasizes the edges in the image.

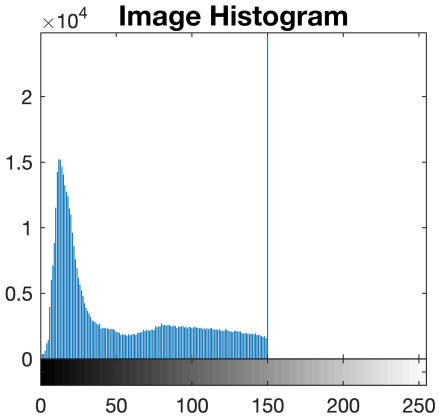
Problem-7: DIPUM3E 3.1 (Page 191)

Read the image **`spillway.tif`** and enhance it to bring out details in the coastal road that are barely visible in the original image. Try enhancement techniques based on the following intensity transformations.

(a) Log

Solution: To determine parameters of the log transformation, we have to study histogram of the given image.

```
clc; close all; clear;
f = imread('../artfiles/spillway.tif');
figure('units','inches',"Position",[0,0,8,3]);
subplot(1,2,1); imshow(f); title('Spillway Image','FontSize',14);
subplot(1,2,2); imhist(f); title('Image Histogram','FontSize',14);
```



There are two characteristics of this histogram that are of interest when analized in conjunction with the image display of **`f:`**

1. We note that the histogram has two dominant. wide-spaced modes, which accounts for the high contrast of the image.
2. The second mode is a single intensity level.

Because it is the highest level. it is easy to deterrline that its value is 150 (**`value = max(f(:))`**). The number of pixels with this high value is close to 25% of the total number of image pixels

```
idx = find(f == 150);
ratio = numel(idx)/numel(f); disp(ratio);
```

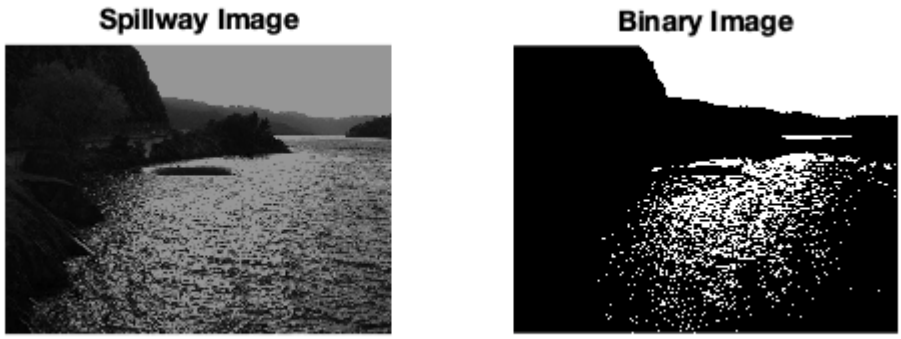
```
    0.2275
```

We display which pixels in the image have this value by creating a binary image in which all pixels whose value is not 150 are black and all pixels with value 150 are white:

```
g = zeros(size(f)); g(idx) = 255;
```

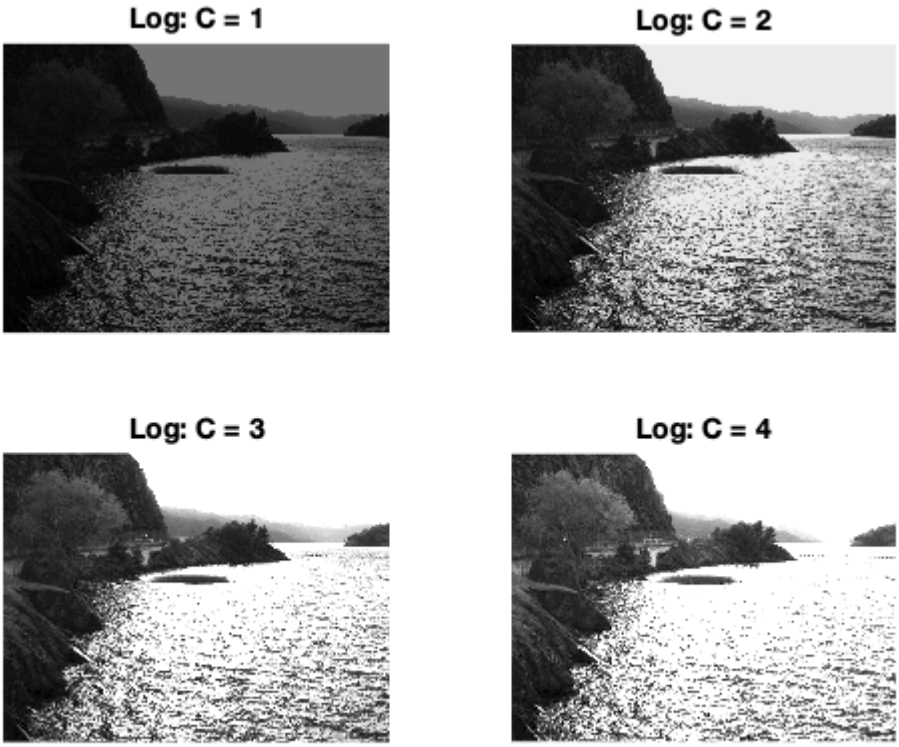
As the following image shows, a great deal of the water and all the sky pixels have value 150. These are the pixels that are white in the image.

```
figure('units','inches',"Position",[0,0,8,3]);
subplot(1,2,1); imshow(f); title('Spillway Image','FontSize',14);
subplot(1,2,2); imshow(g); title('Binary Image','FontSize',14);
```



In order to see details in the dark area, we have to expand the gray scale. The log transformation is the correct one to do this. All we have to do is to determine which value of the multiplying constant to use. In the following display we show images with four multiplying constants: **`[1,2,3,4]`**.

```
ga1 = intensityTransformations(f,'log',1);
ga2 = intensityTransformations(f,'log',2);
ga3 = intensityTransformations(f,'log',3);
ga4 = intensityTransformations(f,'log',4);
figure('units','inches',"Position",[0,0,8,6]);
subplot(2,2,1); imshow(ga1); title('Log: C = 1','FontSize',14);
subplot(2,2,2); imshow(ga2); title('Log: C = 2','FontSize',14);
subplot(2,2,3); imshow(ga3); title('Log: C = 3','FontSize',14);
subplot(2,2,4); imshow(ga4); title('Log: C = 4','FontSize',14);
```

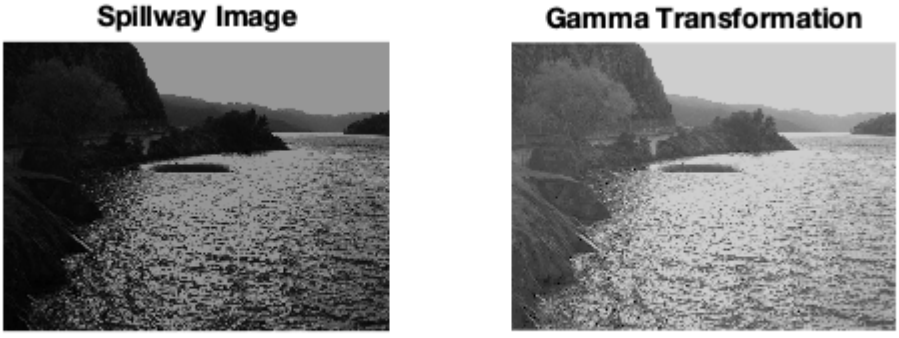


The log transformation with **`C = 3`** shows details without saturating other areas as the image with **`C = 4`** shows.

(b) Gamma

Solution: Based on the results in part (a), we know that the transformation curve should expand the lower end of the intensity scale. This implies a gamma value less than 1. The **`gamma`** transformation affords more control than the **`log`** transformation. After ex[erimentation, a gamma value of 0.4 did a good job.

```
gb = intensityTransformations(f,'gamma',0.4);
figure('units','inches',"Position",[0,0,8,3]);
subplot(1,2,1); imshow(f); title('Spillway Image','FontSize',14);
subplot(1,2,2); imshow(gb); title('Gamma Transformation','FontSize',14);
```

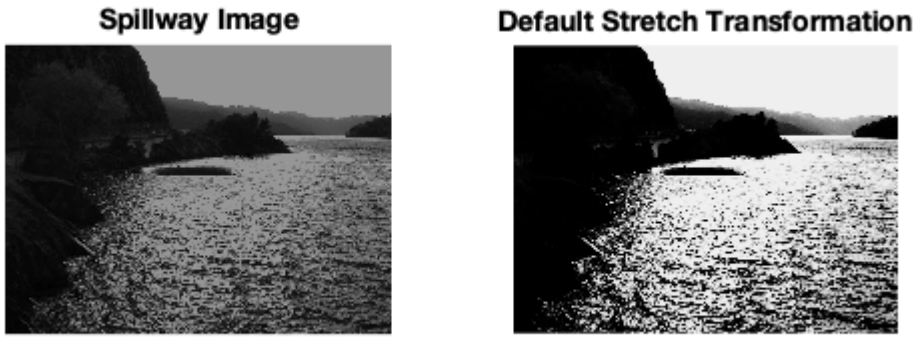


Note that the enhanced image has a broad spectrum of gray levels while showing more details than the **log** transformation image in (a) with **C = 3**.

(c) Stretch

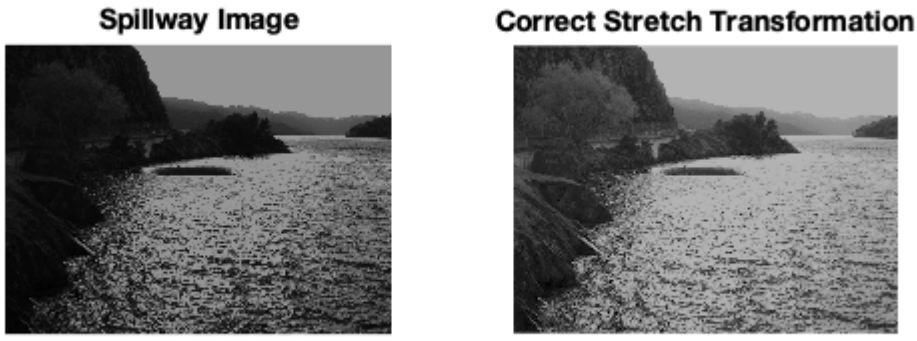
Solution: First, we will use the function **intensityTransformations** with default values for option '**stretch**'.

```
gc1 = intensityTransformations(f,'stretch');
figure('units','inches',"Position",[0,0,8,3]);
subplot(1,2,1); imshow(f); title('Spillway Image','FontSize',14);
subplot(1,2,2); imshow(gc1); title('Default Stretch Transformation','FontSize',14);
```



Clearly, default parameters did not do a good job. Using lower values fpr **E** and **k** (called **m** in the function) yielded results comparable to those in part (b).

```
gc2 = intensityTransformations(f,'stretch',0.25,1.0);
figure('units','inches',"Position",[0,0,8,3]);
subplot(1,2,1); imshow(f); title('Spillway Image','FontSize',14);
subplot(1,2,2); imshow(gc2); title('Correct Stretch Transformation','FontSize',14);
```

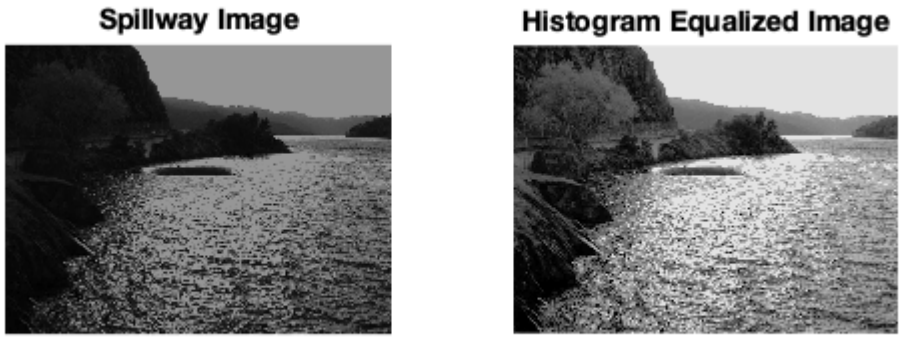


(d) Your specified transformation function.

Solution: The transformation function of choice is histogram equalization due to its automatic determination of the intensity transformation function based on the image data.

```
gd = histeq(f);
```

```
figure('units','inches',"Position",[0,0,8,3]);
subplot(1,2,1); imshow(f); title('Spillway Image','FontSize',14);
subplot(1,2,2); imshow(gd); title('Histogram Equalized Image','FontSize',14);
```



Problem-8 DIPUM3E 3.5 (Page 193)

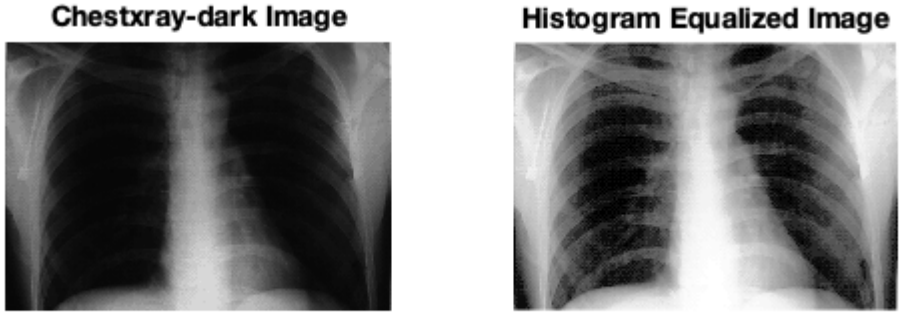
Read image **`chestxray-dark.tif`** and note that the ribs and the other bone structures are not clearly visible.

(a) Histogram Equalization

Enhance the image using histogram equalization function **`histeq`**.

Solution: MATLAB script:

```
clc; close all; clear;
f = imread('../artfiles/chestXray-dark.tif');
ghe = histeq(f);
figure('units','inches',"Position",[0,0,8,3]);
subplot(1,2,1); imshow(f); title('Chestxray-dark Image','FontSize',14);
subplot(1,2,2); imshow(ghe,[]); title('Histogram Equalized Image','FontSize',14);
```



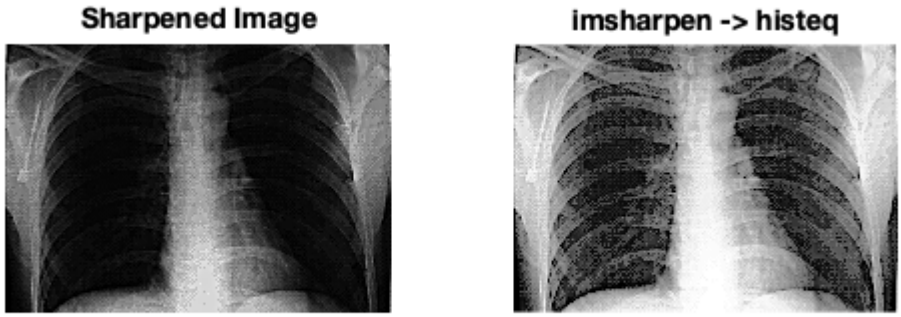
Comment: The resulting image shows the bone structures more clearly, but the image is not sharper than the original image.

(b) Sharpening followed by Enhancement

Enhance the image by using functions **`imsharpen`** followed by **`histeq`**. The objective is to produce an image in which all the ribs and the other bone structures are sharp and visible, and the gray level tonality is also enhanced.

Solution: MATLAB script

```
% Parameters are determined experimentally
gsh = imsharpen(f,'radius',5,'amount',2);
gsheq = histeq(gsh);
figure('units','inches',"Position",[0,0,8,3]);
subplot(1,2,1); imshow(gsh); title('Sharpened Image','FontSize',14);
subplot(1,2,2); imshow(gsheq,[]); title('imsharpen -> histeq','FontSize',14);
```



Comment: The result is an image in which the bone structures are more visible and sharper than in (a) and the gray level tonality is about the same as with just histogram equaliization.

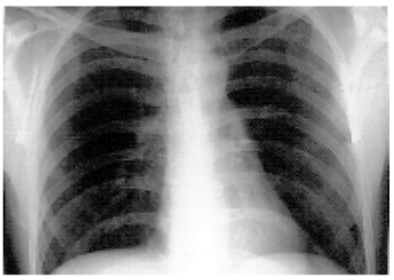
(c) Enhancement followed by Sharpening

Reverse the processing order in (b) and discuss the reason(s) for any differencea in the results in (b) and (c).

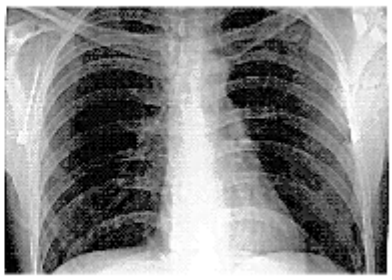
Solution: MATLAB script

```
% Parameters are determined experimentally
geqsh = imsharpen(ghe,'radius',5,'amount',2);
figure('units','inches',"Position",[0,0,8,3]);
subplot(1,2,1); imshow(ghe); title('Histogram Equalized Image','FontSize',14);
subplot(1,2,2); imshow(geqsh,[]); title('histeq -> insharpen','FontSize',14);
```

Histogram Equalized Image



histeq -> insharpen



Discussion: In general, if sharpening is done with a traditiona1 highpass filter, then sharpening followed by histogram equalization yields different results than if the order is reversed. The reason is that, while highpass filtering is a linear process, histogram equalization is not. Highpass filtering tones down overall image contrast, while histogram equalization enhances it. If histogram equalization is applied first, the contrast enhancement done by histogram equalization is "undone" by the toning down of the highpass filter. Here, we are using function **`imsharpen`**, which is a formulation of unsharp masking. As discussed in section 3.4 of DIPUM3E, unsharp masking adds the original back to the sharpened image, thus restoring the contrast tonality lost by highpass filtering. Thus, in this project, while the results in (b) and (c) are slightly different, they are not as dramatically different as would be the case if we had sharpened the image with just highpass filtering.