Homework #6
Due M 4/08

You must turn in your code as well as output files. Please generate a report that contains the code and ouput in a single readable format.

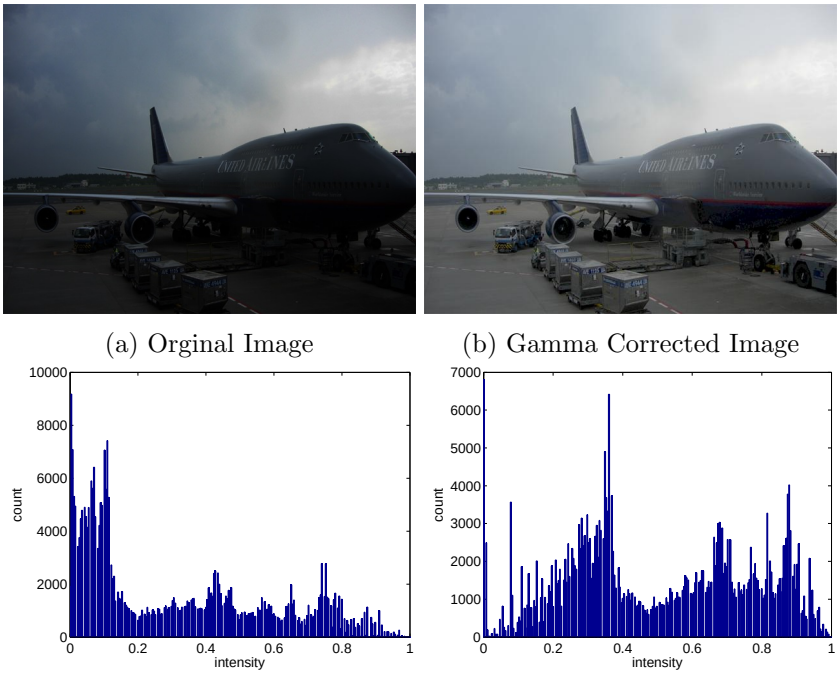
Visit the book website to download companion software, including all the example problems.
`http://www.wiley.com/WileyCDA/WileyTitle/productCd-1118414322.html`

Test Images
Download the sample images from the class website
`http://www.ee.unlv.edu/~b1morris/ee482images/hw06`

1. (KLT 11.4)

   Solution

   The original image and the gamma corrected version is shown in Fig. 1. Notice the dark pixels on the left side of the histogram have been shifted toward the right. The code uses Example 11.3 to setup the images.



(a) Orginal Image (b) Gamma Corrected Image

(c) Orginal Histogram (d) Gamma Corrected Histogram

Figure 1: KLT 11.4

```
%% convert to ycbcr for intensity
[M,N,D] =size(RGB);
ycbcr1 = double(rgb2ycbcr(RGB))/255;
ycbcr2 = double(rgb2ycbcr(RGBgamma))/255;
%display histogram
h1 = hist(reshape(ycbcr1(:,:,1), M*N,1), 256);
h2 = hist(reshape(ycbcr2(:,:,1), M*N,1), 256);
```

```
h=figure;
bar((0:255)/255, h1);
xlabel('intensity')
ylabel('count')
set(gca,'xlim',[0 1]);
h=figure;
bar((0:255)/255, h2);
xlabel('intensity')
ylabel('count')
set(gca,'xlim',[0 1]);
```

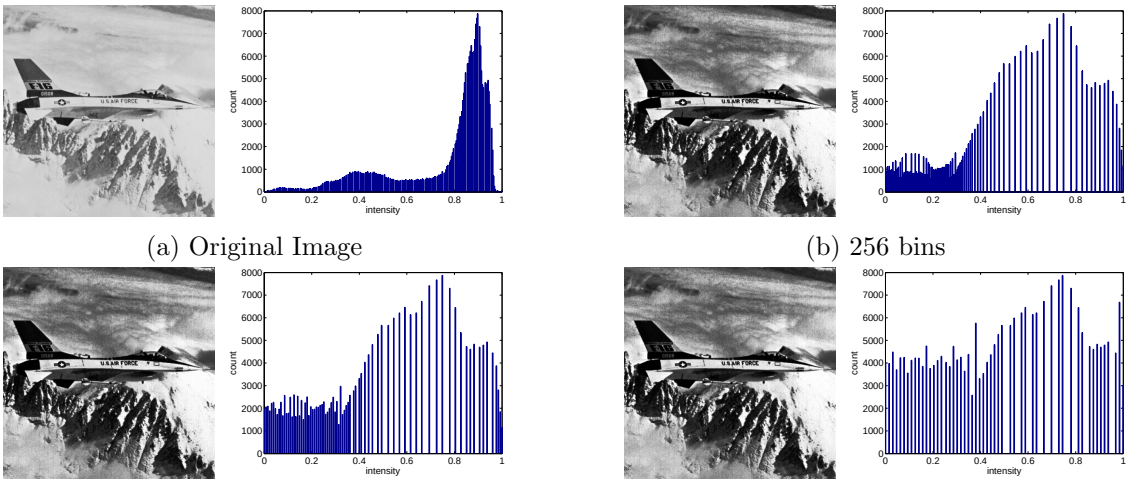
2. Histogram Equalization

   (a) Perform histogram equalization on the `jetplane.png` image using 256, 128, and 64 bins. Compare the original image and the histogram equalized images by plotting the corresponding histograms and images side-by-side in a 2×2 subplot matrix for each of the bin sizes.

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

Solution

(a) The images and associated histograms are shown in Fig. 2. Notice that as the number of bins used for equalization decreases, the number of unique values in the image decreases and spacing between values is greater. However, the output images are quite perceptually similar.



(a) Original Image (b) 256 bins

(c) 128 bins (d) 64 bins

Figure 2: Problem 2(a)

```
I = double(rgb2gray((imread('hw06\jetplane.png'))))/255;
horig = hist(I(:), 256);
h=figure;
bar((0:255)/255, horig); set(gca,'xlim',[0 1]);
xlabel('intensity'), ylabel('count');
```

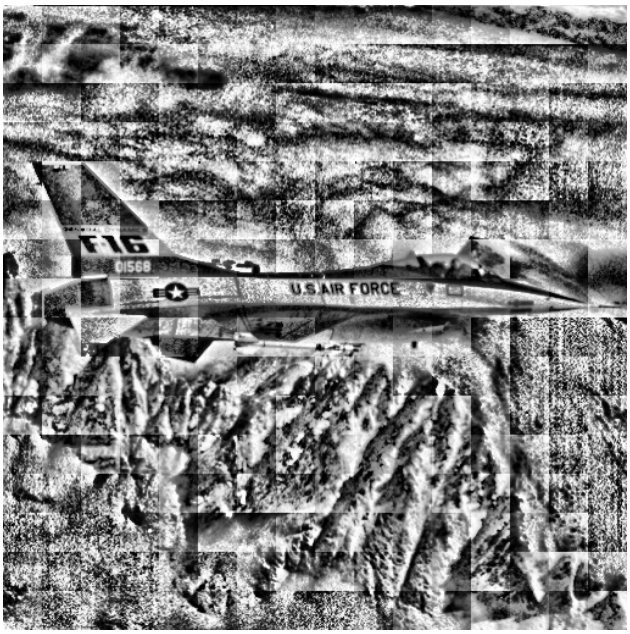


Figure 3: Problem 2(b)

```
bmSaveFigure(h, 'jetplane_hist');
for nbins = [256, 128, 64]
    Ieq = histeq(I, nbins);
    heq = hist(Ieq(:), 256);
    imwrite(Ieq, sprintf('jetplane_n%03d.jpg', nbins));
    h=figure;
    bar((0:255)/255, heq); set(gca,'xlim',[0 1]);
    xlabel('intensity'), ylabel('count');
    bmSaveFigure(h, sprintf('jetplane_hist_n%03d', nbins));
end
```

(b) Histogram equalization with block processing is shown in Fig. 3. Notice the block artifacts and the noise that occurs in the areas of cloud that were all white (lower right and left sides of the image).

```
fun = @(block_struct) histeq(block_struct.data, 256);
Iblock = blockproc(I, [32,32], fun);
```

3. Noise Filtering

(a) Consider image `DSCN0479-001.JPG` as a perfect image. Add white Gaussian noise with variance 0.005. Smooth with a 3×3 and 7×7 box filter and a median filter. Compute the mean squared error (MSE)

$$MSE = \frac{1}{MN} \sum_m \sum_n (I_1(m,n) - I_2(m,n))^2$$

and the peak signal-to-noise ratio (PSNR)

$$PSNR = 20 \times \log_{10}(255/\sqrt{MSE})$$

for the noise reduced images. Which filter has the best results based on the error measures? How do the results compare visually?

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

(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? How much time does each type of filter require (use `tick.m` and `toc.m`)?

Solution

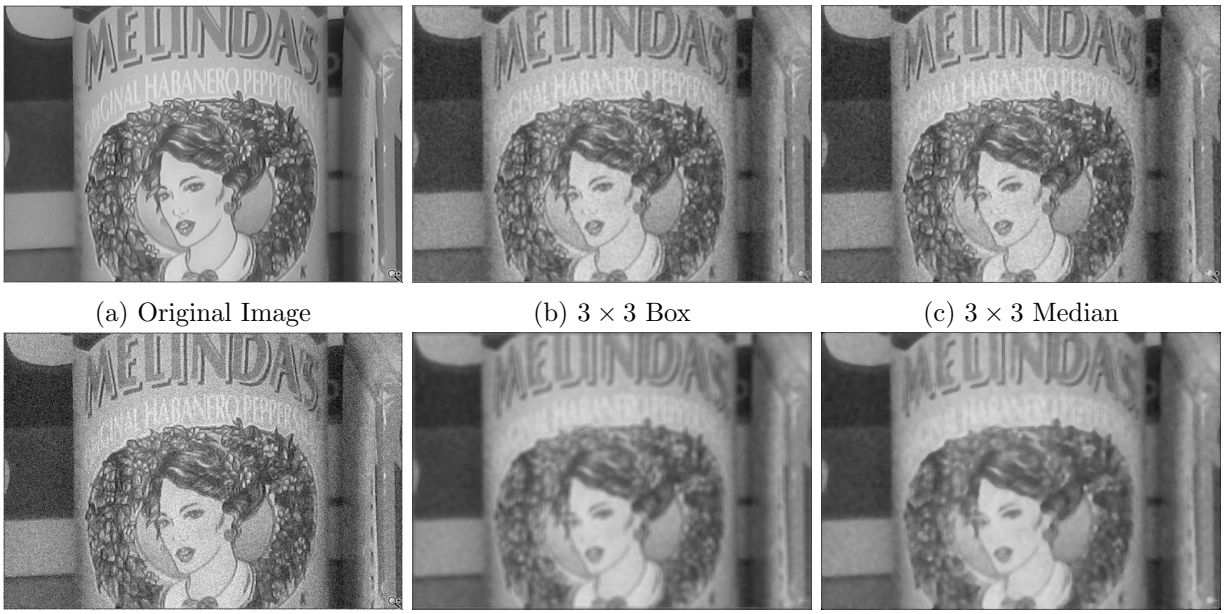
(a) The output images after filtering the noisy image using different filters is shown in Fig. 4. Notice the larger filters produce more blurring, but visually there is little apparent diffrence between median and box (average) filters. The median tends to produce a more "mosaic"-like image. The associated MSE and PSNR are provided in Table 1. Notice the 3×3 box filter has the lowest MSE and highest PSNR indicating the best Gaussian noise removal performance. The code framework for all parts of the problem are similar.

```
I = imread('hw06\DSCN0479-001.JPG');
I = rgb2gray(I);
[M,N] = size(I);
imwrite(I, '3_clean.jpg');
I2 = imnoise(I, 'gaussian', 0, 0.005);
imwrite(I2, '3_gauss.jpg')

emetric = zeros(2,5);
%mse
emetric(1,1) = (1/(M*N))*sum((I(:)-I2(:)).^2);
%psnr
emetric(2,1) = 20*log10(255 / sqrt(emetric(1,1)));

%3x3 box
Ifilt = imfilter(I2, 1/9*ones(3));
%mse
emetric(1,2) = (1/(M*N))*sum((I(:)-Ifilt(:)).^2);
%psnr
emetric(2,2) = 20*log10(255 / sqrt(emetric(1,2)));
imwrite(Ifilt, '3_box3x3.jpg')

%7x7 box
Ifilt = imfilter(I2, 1/49*ones(7));
%mse
emetric(1,3) = (1/(M*N))*sum((I(:)-Ifilt(:)).^2);
%psnr
emetric(2,3) = 20*log10(255 / sqrt(emetric(1,3)));
imwrite(Ifilt, '3_box7x7.jpg')

%3x3 median
Ifilt = medfilt2(I2, [3 3]);
%mse
emetric(1,4) = (1/(M*N))*sum((I(:)-Ifilt(:)).^2);
```

Table 1: Problem 3a - Gaussian Noise

Metric	Noise	3×3 Box	7×7 Box	3×3 Median	7×7 Median
MSE	71.69	32.02	41.72	37.49	39.10
PNSR	29.58	33.08	31.92	32.39	32.21

```
%psnr
emetric(2,4) = 20*log10(255 / sqrt(emetric(1,4)));
imwrite(Ifilt, '3_med3x3.jpg')

%7x7 median
Ifilt = medfilt2(I2, [7 7]);
%mse
emetric(1,5) = (1/(M*N))*sum((I(:)-Ifilt(:)).^2);
%psnr
emetric(2,5) = 20*log10(255 / sqrt(emetric(1,5)));
imwrite(Ifilt, '3_med7x7.jpg')
```



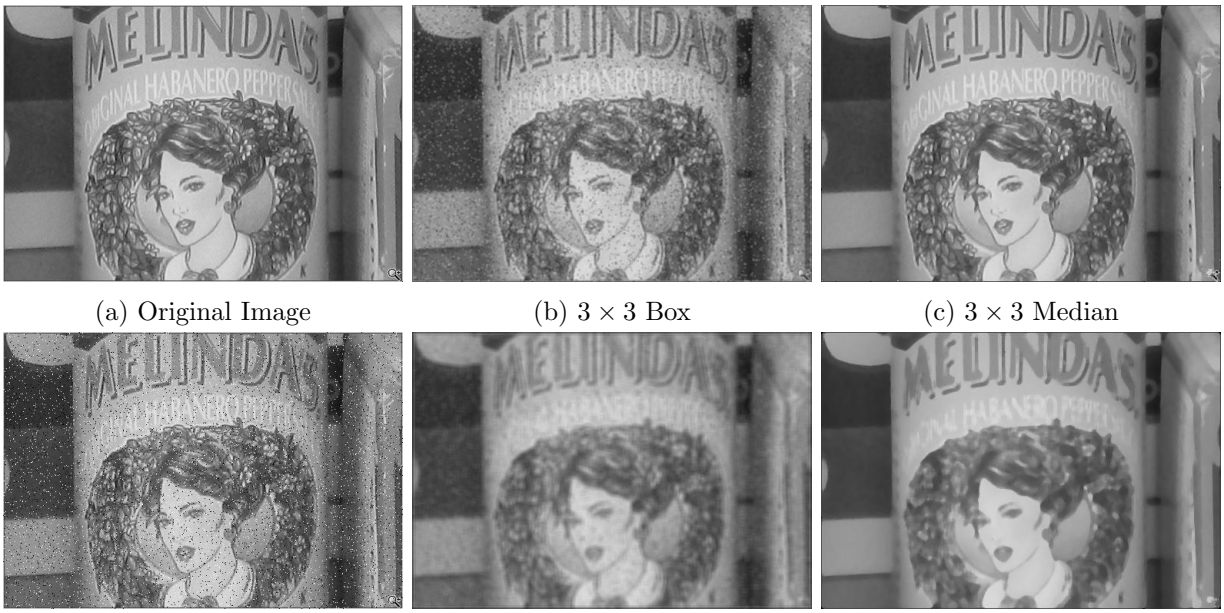
(a) Original Image (b) 3×3 Box (c) 3×3 Median

(d) Noise Image (e) 7×7 Box (f) 7×7 Median

Figure 4: Problem 3(a) - Gaussian Noise

(b) The repeated experiment with salt and pepper noise results are Fig. 5 and Table 2. Notice the 3×3 median filter has the lowest MSE and highest PSNR indicating the best salt and pepper noise removal performance (clearly the case visually). The raw salt and pepper image has the lowest MSE but doesn't look as good as the filtered versions indicating that error metrics do not always match visual quality. In addition, these examples show how the size of the filter needs to be matched to the noise characteristics for best performance. E.g. larger filters do not always mean "better" performance because of blurring.

(c) The results for an image with real noise from high ISO is presented in Fig. 6 and Table



(a) Original Image (b) 3×3 Box (c) 3×3 Median

(d) Noise Image (e) 7×7 Box (f) 7×7 Median

Figure 5: Problem 3(b) - Salt and Pepper Noise

Table 2: Problem 3b - Salt and Pepper Noise

Metric	Noise	3×3 Box	7×7 Box	3×3 Median	7×7 Median
MSE	6.25	42.92	47.38	12.09	31.05
PNSR	40.18	31.80	31.37	37.31	33.21

6. Visually the 3×3 filters are the best performing but it seems as if the median filter has the lowest MSE. However, these numbers really aren't meaningful because we do not have a "clean" image without noise to compare with only the noisy image.

4. Spatial Domain Filtering

The following question operates on the `city.jpg` image.

(a) Perform image smoothing using a 7×7 averaging filter and a Gaussian filter with $\sigma = 0.5$ and 3. Compare the outputs.

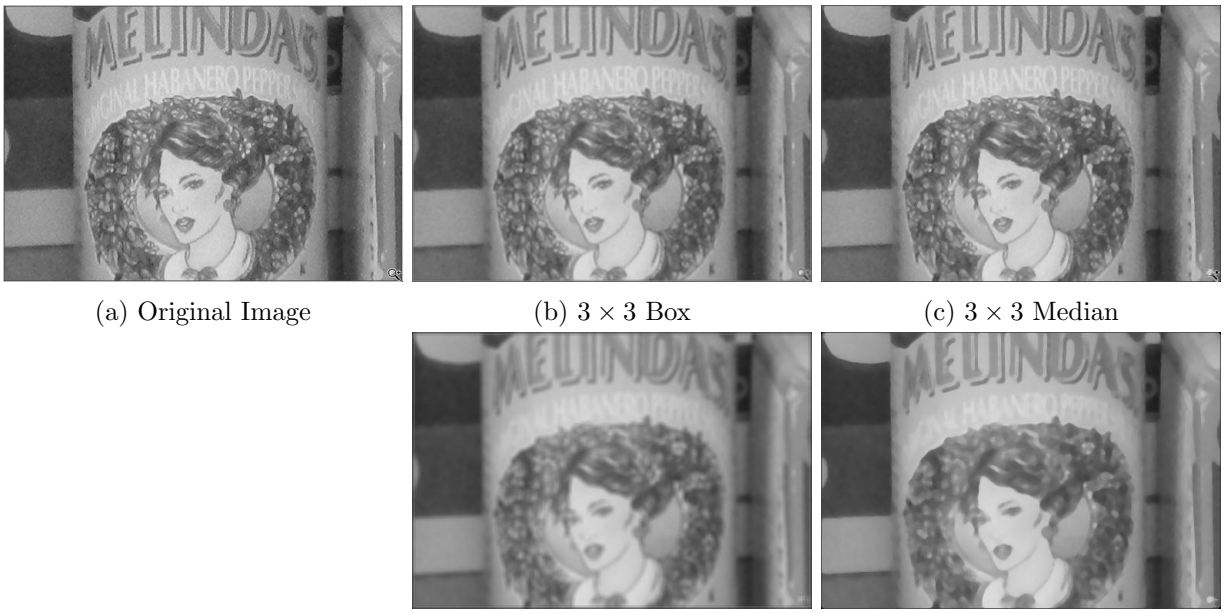
(b) Perform edge enhancement using the Sobel operator (Matlab's default parameters). Repeat using the Laplacian and Laplacian of Gaussian operators. Compare the outputs. Be sure to read the help info for `fspecial.m`.

Solution

(a) The filtered images are shown in Fig. 7. The averaging filter has significant blurring.

Table 3: Problem 3c - Real Noise

Metric	3×3 Box	7×7 Box	3×3 Median	7×7 Median
MSE	12.00	31.71	7.89	22.85
PNSR	37.34	33.12	39.16	34.54



(a) Original Image (b) 3×3 Box (c) 3×3 Median

(d) 7×7 Box (e) 7×7 Median

Figure 6: Problem 3(c) - Real Noise

The larger $\sigma = 3.0$ has similar smoothing to the averaging filter while the smaller $\sigma = 0.5$ maintains sharpness. The Gaussian filter is able to trade off sharpness without changing the kernel size.

```
I = imread('hw06/city.jpg');
H1 = fspecial('average', [7 7]);
H2 = fspecial('gaussian', [7 7], 0.5);
H3 = fspecial('gaussian', [7 7], 3.0);

I1 = imfilter(I, H1);
I2 = imfilter(I, H2);
I3 = imfilter(I, H3);
```

(b) The Sobel operator gives thick edges and the default parameters triggers more on horizontal edges. The Laplacian and the LoG have very similar performance on this image. However, the LoG image seems to have better edge distinction (darker color). The edge images have inverted color for printing purposes.

```
I = rgb2gray(imread('hw06/city.jpg'));
H1 = fspecial('sobel');
H2 = fspecial('laplacian');
H3 = fspecial('log');

I1 = imfilter(I, H1);
I2 = imfilter(I, H2);
I3 = imfilter(I, H3);
```

5. Frequency Domain Filtering

The following question operates on the `city.jpg` image.

(a) Original Image (b) Gaussian $\sigma = 0.5$

(c) Box (d) Gaussian $\sigma = 3.0$

Figure 7: Problem 4(a) - Smoothing

(a) Find the Fourier transform of the image. Be sure to center the frequencies using `fftshift.m`.

(b) Perform image smoothing in the frequency domain using the filters defined in the previous problem. Compare the output images from the two methods (spatial and frequency).

(c) Perform edge enhancement using the filters defined in the previous problem.

(d) Define a lowpass filter in the frequency domain with radius of 1/4 the height. Display the LP filter in the frequency domain and show the result of filtering the input image.

(e) Repeat with a rectangular filter with the same dimension as the ideal lowpass. Compare the results between the ideal filter and the rectangular approximation.

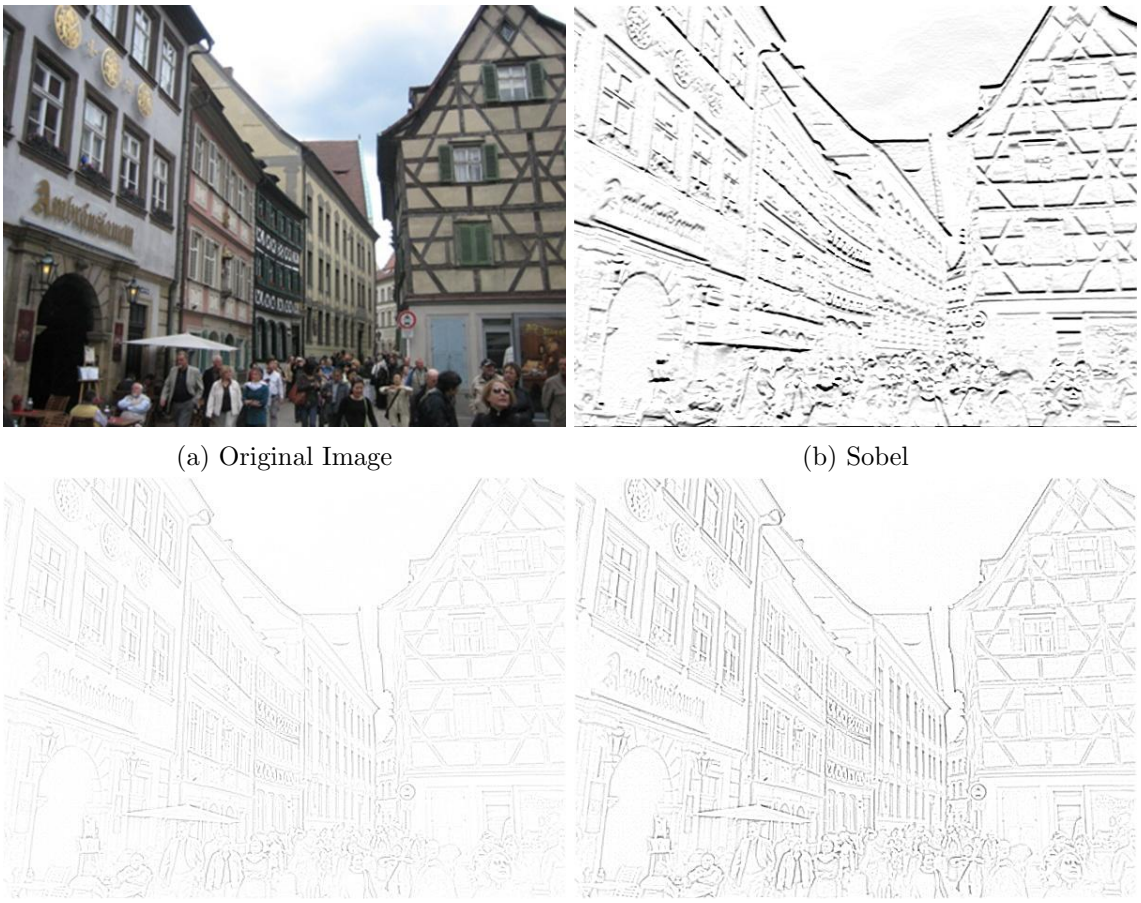
Solution

(a) The FFT in dB of the image is shown in Fig. 9. Using `fftshift.m` places the lowest frequency $(0, 0)$ in the center of the image. There are strong frequency components in the vertical and 45° directions.

(b) The results are visually the same as shown in Fig. 7.

(c) The results are visually the same as shown in Fig. 8.

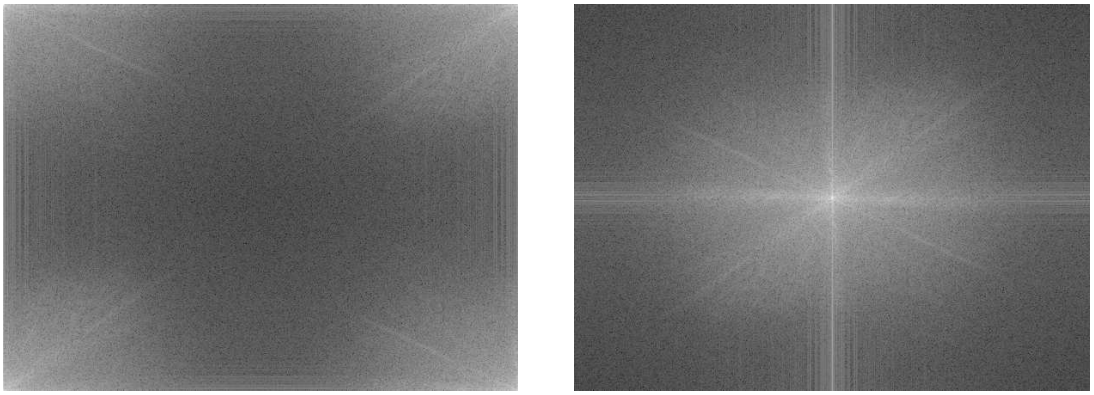
(d) The ideal LP filter is shown in Fig. 10.



(a) Original Image (b) Sobel

(c) Laplacian (d) Laplacian of Gaussian

Figure 8: Problem 4(b) - Edge Enhancement



(a) FFT (b) FFT Shift

Figure 9: Problem 5(a) - FFT [dB]

(e) The approximation is shown in Fig. 11. The resulting image looks almost the same as the circular but it much easier to create.

6. DCT

(a) Ideal LP Filter (b) Output

Figure 10: Problem 5(d) - Ideal LP Frequency Domain Filter

(a) LP Approximation Filter (b) Output

Figure 11: Problem 5(e) - Rectangular LP Approximation

The following question operates on the `jetplane.png` image.

(a) Write a function to compute the DCT coefficients for the image as described in Figure 11.12.

(b) Give the coefficient output for block 3523 (raster scan ordering). Show the output both as a raw matrix and as the zig-zagged ordered vector.

(c) Repeat for block 2637. Compare the outputs of the blocks and comment on results.

Solutions

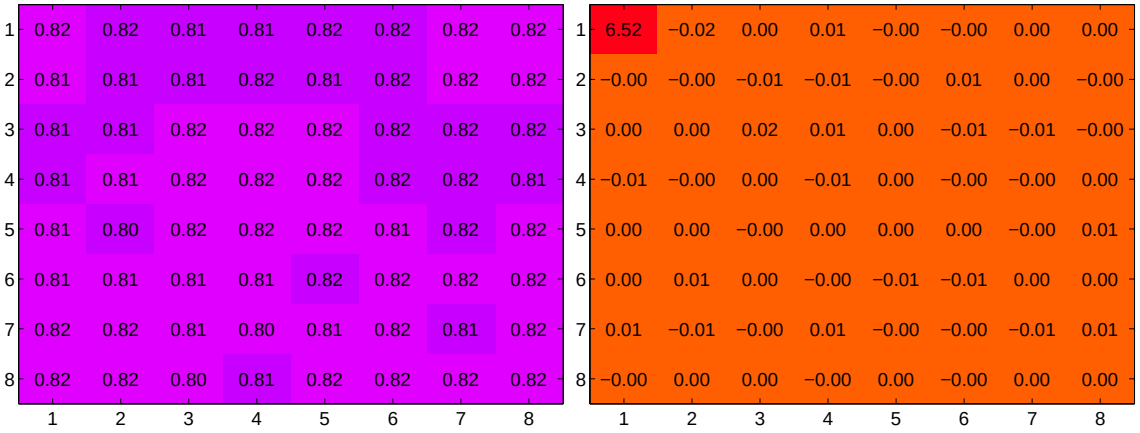
(a) Process each block of the image, compute the DCT coefficients and organize into column vectors using the zig-zag ordering.

```
I = double(rgb2gray((imread('hw06\jetplane.png'))))/255;

%annoymous function
fun = @(block_struct) zigzag8(dct2(block_struct.data));
f = @(I) blockproc(I, [8 8], fun);
```

```
6
7  %create list of dct vectors
8  C = f(I);
9  C2 = reshape(C', 8*8, []);
10
11 function [v] = zigzag8(b)
12 %zig-zag ordering of 8x8 DCT2 block
13
14 v=[     b(1,1) b(1,2) b(2,1) b(3,1) b(2,2) ...
15         b(1,3) b(1,4) b(2,3) b(3,2) b(4,1) ...
16         b(5,1) b(4,2) b(3,3) b(2,4) b(1,5) ...
17         b(1,6) b(2,5) b(3,4) b(4,3) b(5,2) ...
18         b(6,1) b(7,1) b(6,2) b(5,3) b(4,4) ...
19         b(3,5) b(2,6) b(1,7) b(1,8) b(2,7) ...
20         b(3,6) b(4,5) b(5,4) b(6,3) b(7,2) ...
21         b(8,1) b(8,2) b(7,3) b(6,4) b(5,5) ...
22         b(4,6) b(3,7) b(2,8) b(3,8) b(4,7) ...
23         b(5,6) b(6,5) b(7,4) b(8,3) b(8,4) ...
24         b(7,5) b(6,6) b(5,7) b(4,8) b(5,8) ...
25         b(6,7) b(7,6) b(8,5) b(8,6) b(7,7) ...
26         b(6,8) b(7,8) b(8,7) b(8,8)];
```

(b) The DCT coefficients are shown in Fig. 12.

(c) The DCT coefficients are shown in Fig. 12. It is clear that the DC coefficient is larger for the first block because this is a higher intensity block of all white. The second block in contrast is lower intensity and has texture. The texture results in more variation in the DCT coefficients, meaning more bits required for reproduction.



(a) 3523 Image Patch (b) 3523 DCT Block

(c) 2637 Image Patch (d) 2637 DCT Block

Figure 12: Problem 6(b)(c) - DCT