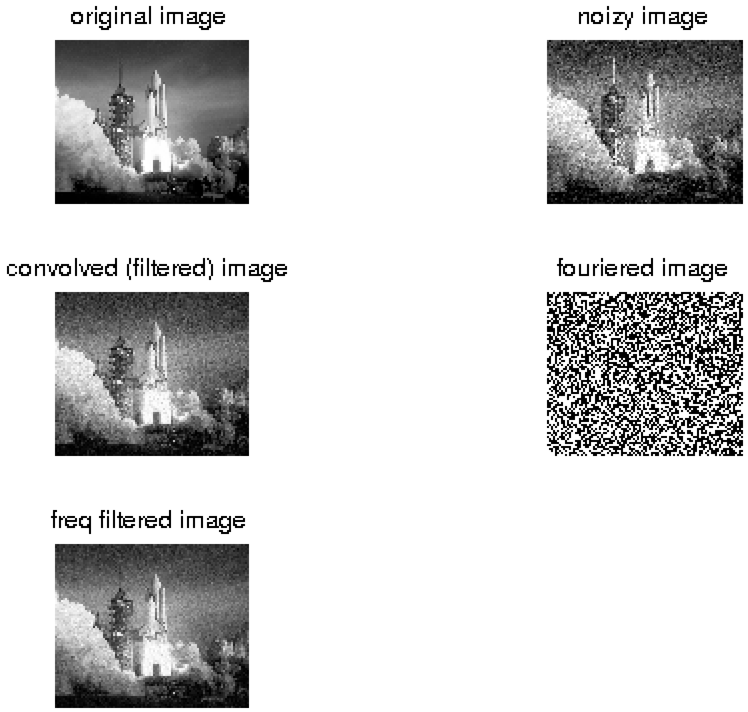
```
close all;
clear;
clc;

% Find the file (different path on my linux vs Win installs)
if (ispc == 0)
 filepath = '/home/me/Dropbox/';
 image = strcat(filepath, 'Space_Shuttle_Columbia_launching.jpg');
else
 filepath='D:\home\Documents\git\machine_vision\hw3\';
 image = strcat(filepath, 'Space_Shuttle_Columbia_launching.jpg');
end

%---------------------------------------------%
% problem 1                                   %
%---------------------------------------------%

%--------%
% part 1 %
%--------%
% read image and transform to gray level

I = imread(image);
I = rgb2gray(I);
I = imresize(I, [600,712]);

figure('name', 'Problem 1');
subplot(3,2,1);
imshow(I);
title('original image');

%--------%
% part 2 %
%--------%
% add gaussian noise

noisy = imnoise(I, 'gaussian');
subplot(3,2,2);
imshow(noisy);
title('noizy image');

%--------%
% part 3 %
%--------%
% create 5x5 gauss mask

mask = fspecial('gaussian', [5 5], 0.5);

%--------%
% part 4 %
%--------%
% filter in space domain
% this utilizes custom convolution function coded in hw2.

cd ../hw2/
space = convolution(noisy, mask);
cd ../hw3/
subplot(3,2,3);
imshow(space);
title('convolved (filtered) image')

%--------%
% part 5 %
%--------%
% fourier transform the image

fftd = fft2(noisy);
subplot(3,2,4);
imshow(fftd);
title('fouriered image');

%--------%
% part 6 %
%--------%
% filter in frequency domain

cd ../hw3/
% convolution in frequency is element by element matrix multiplication
% so mask dimensions have to equal image dimensions
Z = fspecial('gaussian', [600 712], 0.5);
Z = fft2(Z);
filtered = fftd.*Z;

subplot(3,2,5);
imageD = ifft2(filtered);
dmin = min(min(abs(imageD))); dmax = max(max(abs(imageD)));
imshow( ( ifftshift(imageD)), [dmin dmax]),
title('freq filtered image');



%---------------------------------------------%
% problem 2                                   %
%---------------------------------------------%

%--------%
% part 1 %
%--------%
% Apply the DCT on an image of your choice.

imDCT=im2double(I);
imDCT=dct2(I);
figure('name', 'Good DCT choices')
subplot(3,2,1);
imshow(log(abs(imDCT)), []), colormap(jet), colorbar;
title('dct of original image')
```

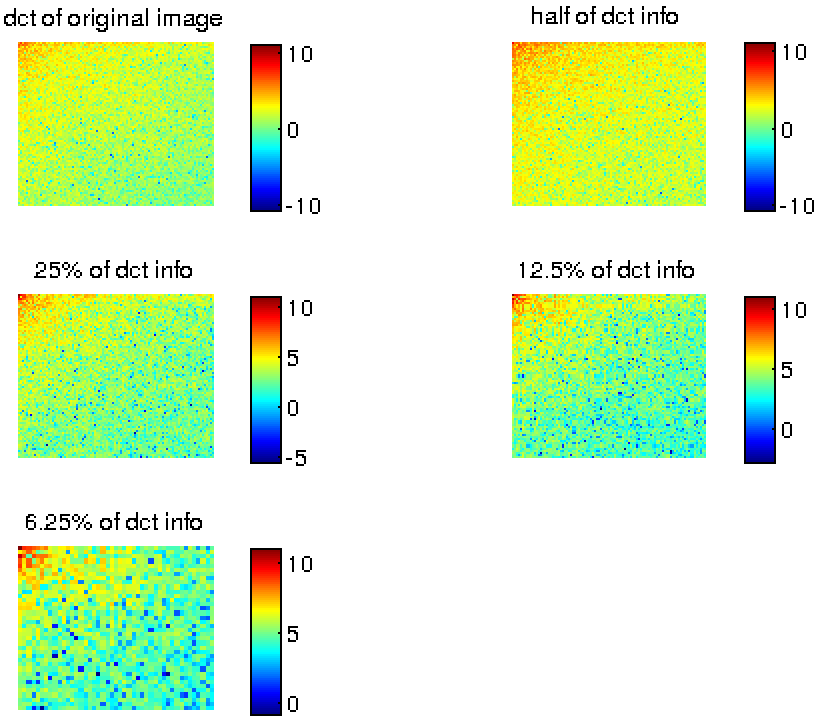
```
%--------%
% part 2 %
%--------%
% cut these images, store top left
% these crop the image using a custom cropping function cropper

imDCT2=cropper(imDCT, .5, 'normal');
imDCT4=cropper(imDCT, .25, 'normal');
imDCT8=cropper(imDCT, .125, 'normal');
imDCT16=cropper(imDCT, .0625, 'normal');


%--------%
% part 3 %
%--------%
% display the images

subplot(3,2,2);
imshow(log(abs(imDCT2)),[]), colormap(jet), colorbar;
title('half of dct info')

subplot(3,2,3);
imshow(log(abs(imDCT4)),[]), colormap(jet), colorbar;
title('25% of dct info')

subplot(3,2,4);
imshow(log(abs(imDCT8)),[]), colormap(jet), colorbar;
title('12.5% of dct info')

subplot(3,2,5)
imshow(log(abs(imDCT16)),[]), colormap(jet), colorbar;
title('6.25% of dct info')


%--------%
% part 4 %
%--------%
% invert the dct and display

figure('name', 'inverse DCTd image')
[M,N]=size(imDCT);
K = uint8(imresize(sqrt(idct2(imDCT2)), [M,N]));
subplot(2,2,1)
imshow(K, []);
title('half of data')

K = uint8(imresize(sqrt(idct2(imDCT4)), [M N]));
subplot(2,2,2)
imshow(K, []);
title('quarter of data')

K = uint8(imresize(sqrt(idct2(imDCT8)), [M N]));
subplot(2,2,3)
```

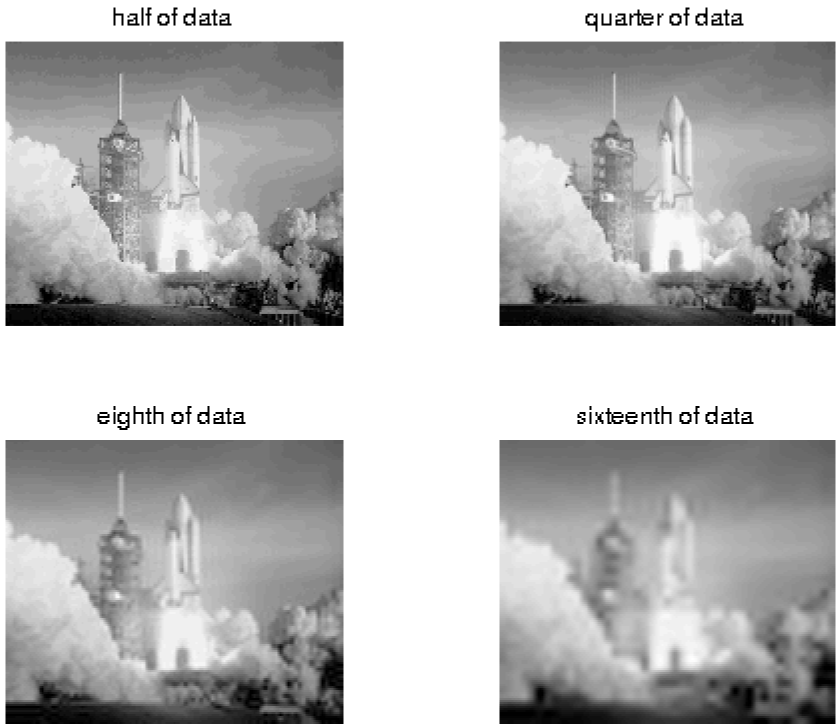
```
imshow(K, []);
title('eighth of data')

K = uint8(imresize(sqrt(idct2(imDCT16)), [M N]));
subplot(2,2,4)
imshow(K, []);
title('sixteenth of data')

%--------%
% part 5 %
%--------%
% cut these images, store bottom right

[x,y]=size(imDCT);
badDCT2  = imcrop(imDCT, [ceil(x/2), ceil(y/2), x/2, y/2]);
badDCT4  = imcrop(imDCT, [ceil(x/4), ceil(y/4), x/4, y/4]);
badDCT8  = imcrop(imDCT, [ceil(x/8), ceil(y/8), x/8, y/8]);
badDCT16 = imcrop(imDCT, [ceil(x/16), ceil(y/16), x/16, y/16]);


%--------%
% part 6 %
%--------%
% display dcts of bad stuffs

figure('name', 'bad dct data');
subplot(2,2,1)
imshow(log(abs(badDCT2)),[]), colormap(jet), colorbar;
title('half of data')


subplot(2,2,2)
imshow(log(abs(badDCT4)),[]), colormap(jet), colorbar;
title('quarter of data')

subplot(2,2,3)
imshow(log(abs(badDCT8)),[]), colormap(jet), colorbar;
title('eighth of data')

subplot(2,2,4)
imshow(log(abs(badDCT16)),[]), colormap(jet), colorbar;
title('sixteenth of data')



%--------%
% part 7 %
%--------%
% inverse bad dcts

figure('name', 'inverse DCTd image')
[M,N]=size(imDCT);
K = uint8(imresize(sqrt(idct2(badDCT2)), [M,N]));
subplot(2,2,1)
```

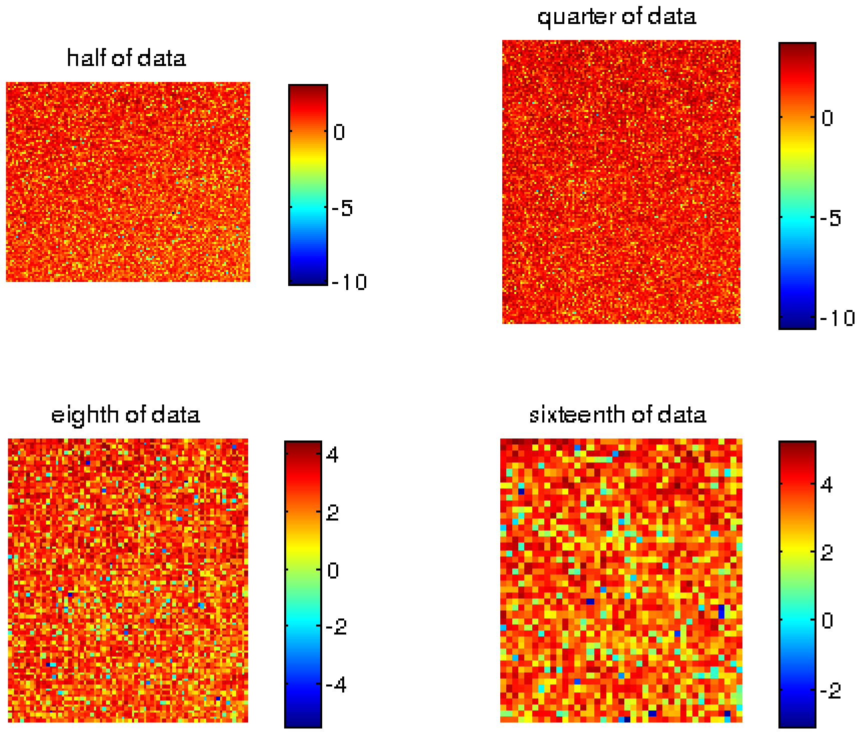
```
imshow(K, []);
title('half of data')

K = uint8(imresize(sqrt(idct2(badDCT4)), [M N]));
subplot(2,2,2)
imshow(K, []);
title('quarter of data')

K = uint8(imresize(sqrt(idct2(badDCT8)), [M N]));
subplot(2,2,3)
imshow(K, []);
title('eighth of data')

K = uint8(imresize(sqrt(idct2(badDCT16)), [M N]));
subplot(2,2,4)
imshow(K, [0 255]);
title('sixteenth of data')
```

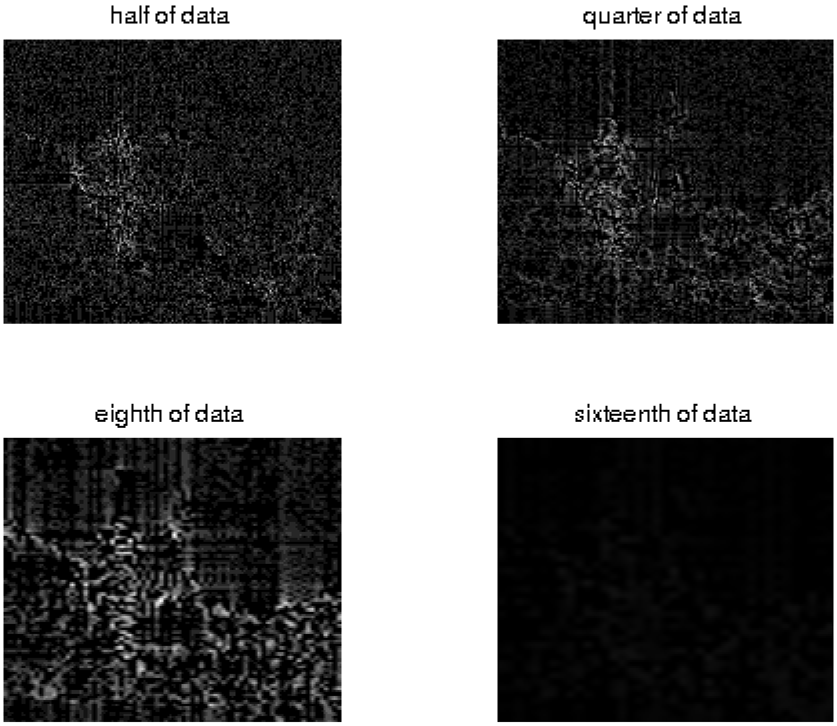
```
Warning: Displaying real part of complex input.
Warning: Displaying real part of complex input.
Warning: Displaying real part of complex input.
Warning: Displaying real part of complex input.
Warning: Displaying real part of complex input.
Warning: Displaying real part of complex input.
Warning: Displaying real part of complex input.
Warning: Displaying real part of complex input.
Warning: Displaying real part of complex input.
```


original image
noizy image
convolved (filtered) image
fouriered image
freq filtered image


dct of original image
10
0
-10
half of dct info
10
0
-10
25% of dct info
10
5
0
-5
12.5% of dct info
10
5
0
6.25% of dct info
10
5
0


half of data
quarter of data
eighth of data
sixteenth of data


half of data
0
-5
-10
quarter of data
0
-5
-10
eighth of data
4
2
0
-2
-4
sixteenth of data
4
2
0
-2



Published with MATLAB® R2013b