EEE 178: Homework 3

Curtis Muntz

February 20, 2014

- Problem 1
  - See attached MATLAB code and figures.
  - Part 6

    After comparing the differences between the filtered image using the spacial convolution method and the filtered image using the frequency convolution method, it seems as if the spacial filtering was inferior to the frequency filtering. Mathematically, they should produce the exact same effect and the two should not differ. An examination as to why they differ makes me place blame on one thing: mask sizing. In the spacial domain, it is computationally expensive to use a large Gaussian mask, but in the frequency domain, it isn't just easier, it's necessary. Because we use smaller masks in the time domain, it hurts our ability to effectively filter out the noise.
- Problem 2
  - See attached MATLAB code and figures.
  - Part 3

    As we take less and less of the DCT data, since we're still taking data from the 'important' region of the DCT, there still exists a distribution. That is — the data still varies in magnitude in an orderly fashion. The top-left most part of the DCT image having the highest magnitude, and conversely the bottom-right most part of the image containing low magnitude values.
  - Part 4

    As we take less and less data from the DCT, the image quality suffers, but the 'gist' of the image is still there. It is very interesting to see that even using half of the image's data, we can nearly construct the same image, albeit not perfectly.
  - Part 6

    The bottom right of the DCT image was taken and scaled lower and lower, and an interesting phenomena was noted. First of all, the data seems to have less variance. that is, while outliers do exist, most of the data corresponds to high magnitude values. Secondly, the data is arranged more randomly. while the magnitude of the data is relatively the same everywhere in this region, the differences are much more sporadic and random, less organized than the data in the 'good DCT region'.
  - Part 7

    The inverse of these DCT regions produce very unusable images. You can still make out some parts of the original image, in my case you can still see some of the rocket exhaust, but for the most part the image doesn't contain enough information to be recognizable by a human as the original image.
  - Part 8

    Because mathematically the DCT is invertible, inverting the whole DCT of the original image should reproduce the original image. However, because we are taking only portions of the mapping, we are losing some data, and therefore some of the image. When taking the lower right most

portion of the DCT, we lose most of the image, but also in the opposite case, we still lose some of the image. This can be seen as the general image degregation between the original image and the iDCT image. While the discarded information obviously wasn't as 'important' to the image, it still makes a difference between a perfectly reconstructed image and the lossy 'approximation'.

```
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')
```

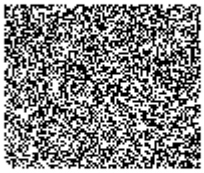
```
Warning: Displaying real part of complex input.
Warning: Displaying real part of complex input.
Warning: Displaying real part of complex input.
Warning: Displaying real part of complex input.
Warning: Displaying real part of complex input.
Warning: Displaying real part of complex input.
Warning: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

original image

noizy image

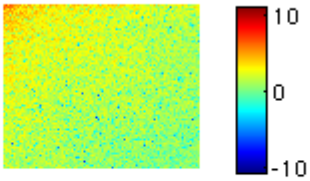
convolved (filtered) image

fouriered image

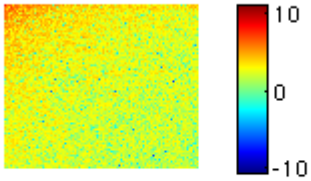


freq filtered image

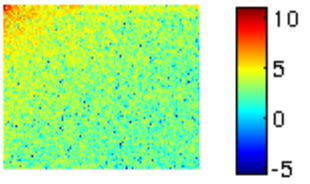
dct of original image



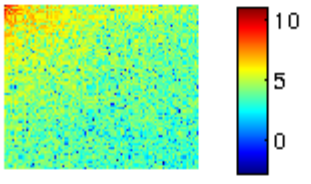
half of dct info



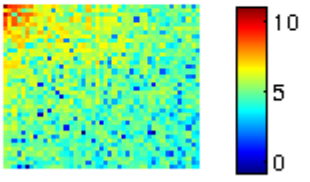
25% of dct info



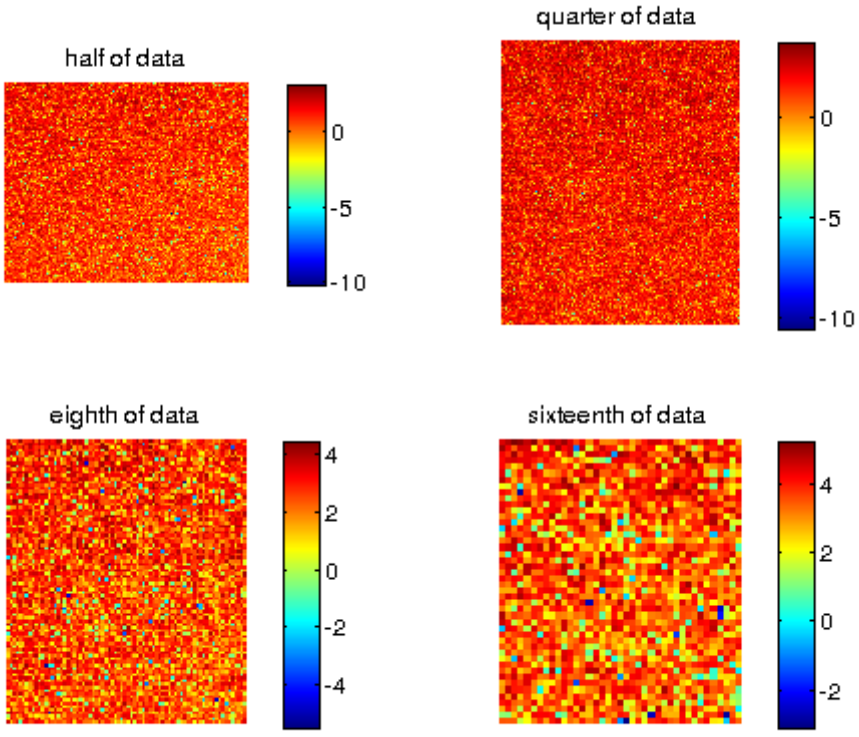
12.5% of dct info

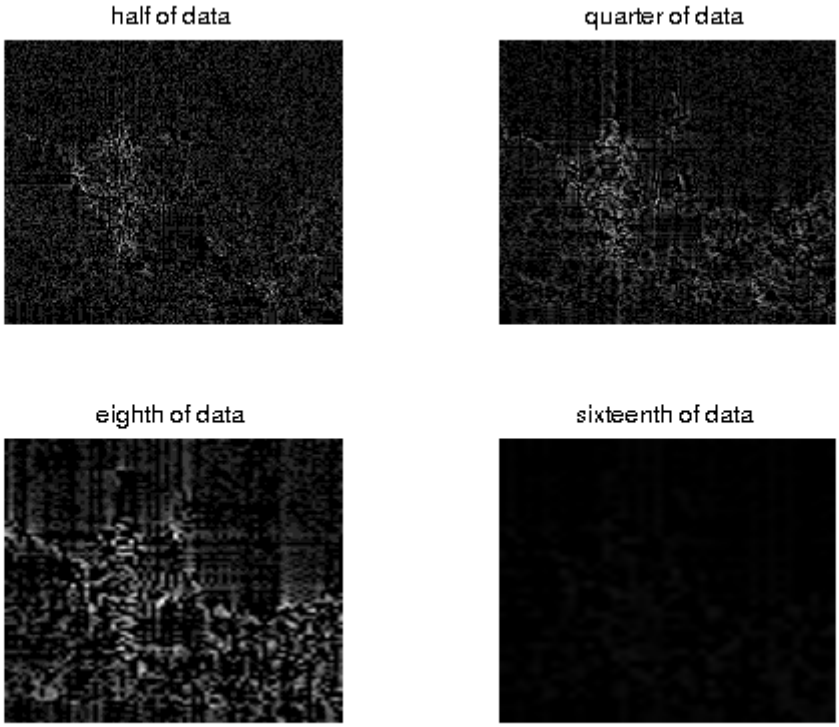


6.25% of dct info




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

```
function [croppedIMG] = cropper(I, fraction, section)
 [M,N]=size(I);
    %currently only works for cropping images from the top left to a
    %certain percentage of the original size.
 croppedIMG=zeros(ceil((M*fraction)),ceil((N*fraction)));
 croppedIMG = im2double(croppedIMG);

 if (strcmp(section, 'normal') == 1)
  rowStart = 1;
  colStart = 1;
  rowEnd=ceil((M*fraction));
  colEnd=ceil((N*fraction));
%  elseif(strcmp(section, 'bad') == 1 )
%   rowStart=ceil((M*fraction));
%   colStart=ceil((N*fraction));
%   rowEnd=M;
%   colEnd=N;
 end

 for i=rowStart:rowEnd
  for j=colStart:colEnd
   croppedIMG(i,j)=I(i,j);
  end
 end
end
```

```
        Error using cropper (line 2)
        Not enough input arguments.
```

Published with MATLAB® R2013b