EE 224 Signals and Systems I

Lab 08: Digital Image Processing: Filtering Images using Convolution

Pre-Lab: Read this handout before going to your assigned lab section. Download the Matlab file containing the images from course webpage.

Verification: Show the TA your written definitions of the key terms at the beginning of the lab. The noise reduction and edge filtering must be completed during your assigned lab time and demonstrated to your TA for verification.

Lab Report: Your report should include the results from Sections 2.1,2.2, and 2.3 with images and explanations. You need to label the axes of your plots and include a title for every plot. In order to keep track of plots, include your plot in-lined within your report. In your report, describe what you did and what your output sounded like. Include your Matlab code in an appendix.

1. Introduction

Image signals can be filtered in a manner similar to one-dimensional signals. Images vary in intensity across the image just as an audio signal varies in time. In images we use the term "spatial frequency". A high spatial frequency means that the image is changing quickly in intensity (e.g. a striped shirt). An object with a low spatial frequency only changes slightly over many pixels. A low pass filter will blur the image by removing the high frequency details. A high pass filter will enhance the edges and other areas that change quickly in intensity.

One way of filtering is to first filter the rows with a one dimensional filter and then filter the columns with a one dimensional filter. Two dimensional filters can also be used that use a neighborhood of pixels to filter the signal. Filtering can help remove noise from an image by averaging out the effects of random noise fluctuations. High pass filters help detect edges and sudden spatial changes in an image.

2. Image Filtering

This lab will start by filtering a single row of an image using the *conv.m* function in Matlab. The next step will be to extend this procedure to the entire image. Read in the file of image data for this lab and display the image in matrix *cicada* in grayscale. This can be done by loading the jpg images and converting them to grayscale:

```
cicada=imread('cicada.jpg');%read in image
gray_cicada = rgb2gray(cicada);  %convert to grayscale image
gray_cicada = double(gray_cicada);% convert image to a type suitable
for filtering
```

You may also find the images in the .MAT file in `ip2_images.mat` which is available on the course website. Load the file using the command:

```
> load ip2_images.mat
```

Extract the 50 th row of the image and plot the intensity values. Filter this row using an FIR averaging filter that averages seven samples. Plot the result on the same plot. Is the filtered output rougher or smoother than the input?

```
>row = cicada(50,:); %extracts 50th row
>ave_filt=ones(1,7)/7; % calculate filter coefficients
>filt_row = conv(row,ave_filt); %filter 50th row
% plot the unfiltered and filtered signals
>subplot(211);plot(row);xaxis('row_element');yaxis('intensity');
>subplot(212);plot(filt_row);xaxis('row_element');yaxis('filtered
intensity');
```

Extract the 100 th column of the image and plot the intensity values. Filter this column using an first difference filter (*y(n)=x(n)-x(n-1)*). Plot the result on the same plot. Is the filtered output rougher or smoother than the input?

2.1 Image Filtering Using Repeated One Dimensional Filters

Using a similar filtering technique as in the previous section, we can write an m-file that filters all the rows of the image. This creates a new image that is filtered in the horizontal direction. Display this image next to the original image. Remember to round the image pixels so they are integers.

```
>filt=ones(1,7)/7;
>y_row=yourroutine(image,filt);
>y_row=round(y_row);
```

The *conv.m* routine lengthens the row by the length of the filter minus 1 elements. To get an image of the original size 256X256 back, extract the center of the filtered row (Note this is easiest if filters have an odd length):

```
%find filter half-length
>length_filt=length(filt);half_len=floor(length_filt/2);
% extract the 256 pixels in the center of the rows
>y_row=y_row(:,half_len+1:256+half_len);
```

What does the filtered image look like? Is it blurred or sharpened? Give a qualitative description of what you see.

The image in the previous part was only filtered in one direction. Filtering the columns requires another *for* loop that filters all 256 columns of the image. Filter the image y_row to finish the filtering process. Repeated applications of the *conv* function are very inefficient, so this may take a long time. Display the result of the row-only filtering with the results from filtering both columns and rows. Compare your results with the row only filtering.

Repeated applications of the *conv* function are very inefficient, so the operations above may take awhile. Matlab has built-in function *conv2* that does either row or column filtering with a single call. The optional argument 'same' gives a resulting image that is the same size as the original image. If the filter vector is a row vector then the rows are

filtered. IF the filter vector is a column then the columns are filtered. Verify that using *conv2* gives you the same results as the previous section.

```
>filt_h=ones(1,7)/7;
>y_row=conv2(image,filt_h,'same');
>filt_v=ones(7,1)/7;
>y_filt=conv2(y_row,filt_v,'same);
```

Filter the image with the following filters.

- 3 point averaging filter
- 7 point averaging filter
- [1 2 4 2 1]/10

Filter both the columns and rows of the image. Comment on the effect of each filter. Pay special attention to image regions with a lot of detail and the edges. Which filters are low-pass filters and which are high-pass filters? Describe the general effect of high pass and low pass filters on an image. Display all four images on a single plot and turn in with your lab. (be sure to label your pictures)

2.2 Noise Reduction

Noise often has high spatial frequency that it looks like salt and pepper in the picture. Most objects in the image have lower spatial frequencies so we can often improve the image by low pass filtering with an averaging filter. This technique works well if the noise does not vary that much from the image. Speckle or impulsive noise is very extreme noise that looks like white or black spots on an image. This sections looks at how averaging filters affect the image.

1. Display the image '`ghopper_g`' in grayscale. The image has been corrupted by additive Gaussian Noise. You will use an averaging filter to remove the noise in this image.

2. Filter the image `ghopper_g` with a 3 point averaging filter and a 7 point averaging filter using *conv2*. Display the results along with the original image using the command shown below: (this displays the three images and a black rectangle). Discuss the results. Assess the trade-off between noise reduction and image resolution (how clear objects are in the image).

```
>imagesc([ghopper_g, ghopper_g _3; ghopper_g 7,zeros(256,256)]);
```

3. Display the image '`ghopper_sp`' in grayscale. The image has been corrupted by additive salt and pepper noise. You will use an averaging filter to try to remove the noise in this image.

4. Filter the image `ghopper_ sp` with a 3 point averaging filter and a 7 point averaging filter using *conv2*. Display the results along with the original image using the command from part 2.

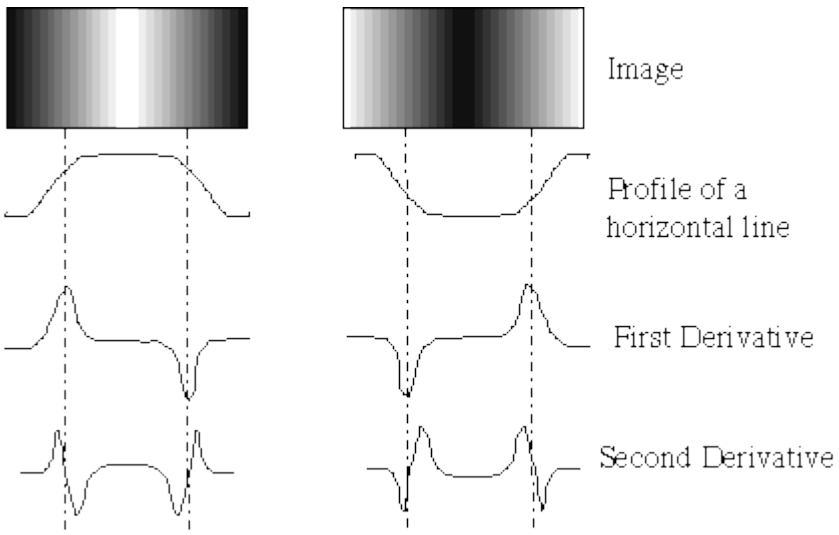
5. Compare the results for filtering different types of noise. Does the filtering work better for one type of noise? If so, why?

2.3 Edge Detection

Image analysis deals with techniques for extracting information from images. The first step is generally to segment an image. Segmentation divides an image into its constituent parts or objects. For example in a military air-to-ground target acquisition application one may want to identify tanks on a road. The first step is to segment the road from the rest of the image and then to segment objects on the road for possible identification. Segmentation algorithms are usually based on one of two properties of gray-level values: discontinuity and similarity. For discontinuity, the approach is to partition an image based on abrupt changes in gray level. Objects of interest are isolated points, lines, and edges.

Definition of Edges

An edge is the boundary between two regions with relatively distinct gray-level properties. The idea underlying most edge-detection techniques is the computation of a relative derivative operator. The figure below illustrates this concept. The picture on the right shows an image of a light stripe on a dark background, the gray-level profile of a horizontal scan line, and the first and second derivatives of the profile. The first derivative is positive when the image changes from dark to light and zero where the image is constant. The second derivative is positive for the part of the transition associated with the dark side of the edge and negative for the transition associated with the light side of the edge. Thus the magnitude of the first derivative can be used to detect the presence of an edge in the image and the sign of the second derivative can be used to detect whether a pixel lies on the light or dark side of the edge.



Spatial Masks

We will use two dimensional spatial masks to find lines of different directions in an image. The mask is similar to the filters used in part 2.2 of this lab except that it operates

in two dimensions. The mask is centered upon each pixel in the image and the following weighted calculation is performed (*x* is the original image pixels, *y* is the output):

$$
\begin{aligned}
y(i,j) &= \sum_{k=-1}^{1}\sum_{l=-1}^{1} w_{k,l}x(i-k,j-l) \\
&= w_{1,1}x(i-1,j-1) + w_{1,0}x(i-1,j) + w_{1,-1}x(i-1,j+1) \\
&\quad + w_{0,1}x(i,j-1) + w_{0,0}x(i,j) + w_{0,-1}x(i,j+1) \\
&\quad + w_{-1,1}x(i+1,j-1) + w_{-1,0}x(i+1,j) + w_{-1,-1}x(i+1,j+1)
\end{aligned}
$$

Mask:

$$
\begin{bmatrix}
w_{1,1} & w_{1,0} & w_{1,-1} \\
w_{0,1} & w_{0,0} & w_{0,-1} \\
w_{-1,1} & w_{-1,0} & w_{-1,-1}
\end{bmatrix}
$$

The masks shown below respond more strongly to lines of different directions in an image. The first mask would respond very strongly to horizontal lines with a thickness of one pixel. The maximum response would occur when line passed through the middle row of the mask. The direction of a mask can be established by noting that the preferred direction is weighted with a larger coefficient than the other possible directions.

(a)			(b)			(c)			(d)		
-1	-1	-1	-1	-1	2	-1	2	-1	2	-1	-1
2	2	2	-1	2	-1	-1	2	-1	-1	2	-1
-1	-1	-1	2	-1	-1	-1	2	-1	-1	-1	2

Exercises Directional Edge Detection

Load the image *ghopper*.
Filter the image using filters (a), (c),) above. Describe your results for each case. Which edges were highlighted in each case? The matlab commands for the first case are given below.

>wa=[-1,-1,-1,2,2,2;-1,-1,-1];

```
>y_a=conv2(ghopper,wa);
>y_a=round(y_a);
>imagesc(y_a);colormap('gray(256)');
```

Display your results by plotting the two output images together. Comment on your results.

Extra Credit

For extra credit, try your edge finder on the noisy images from part 2.2. Compare your results to the results from the edge detection section.

Instructor Verification Sheet

Student Name:

Date:

2.1 _____ ___

2.2 _____ ___

2.3 _____ _____

2.3 Extra Credit _____ ____