EN1610 Image Understanding

Lab # 2: Filtering and Convolution: Neighborhood Operations, Deblurring

The goal of this second lab is to

- Understand the concept of convolution
- Implement various filters, and understand their effect on the image
- Understand various type of image noise
- Techniques to restore image, that have been severely degraded

All the images you will need for the lab can be downloaded off the class website, `http://vision.lems.brown.edu/engn161/fall2011/Labs`.

2D convolution

1. Implement 2D convolution As you have learned in class, the filters you have seen whether to smooth or enhance, are convolved with the image. The equation 1 below represents the discrete 2D convolution formula you will implement for the lab.

$$y[m,n] = x[m,n] * h[m,n] = \sum_{j=-\omega}^{\omega} \sum_{i=-\omega}^{\omega} x[i,j]h[m-i,n-j] \tag{1}$$

Write a matlab function to implement 2D convolution. The function, will take in a filter, and the image, and return the output of the filter.

```
function [output]=my_conv(image,filter)
```

- You will need to do a raster scan over the input image, and for each pixel, apply the filter, this will require four loops (Hint: Think about what direction to do the raster scan, over columns or rows)
- In cases where your filter falls outside the region (borders of the image) , pad with a single strip of the border
- Set the output to an empty matrix, the same size as the input image, (The output of full convolution is the sum of the length of the two signals minus one, but in this case we will restrict it to the same size as the input image)

It is very **important** you have working, code as you will use this for future problems. You can compare your results to matlab. They should be equal.

```
image=magic(10);
```

```
filter=ones(3,3);
matlab_output=imfilter(image,filter,'replicate');
my_output=my_conv(image,filter);
```

You can try other , other examples, to make sure your code behaves exactly as matlab does.

Problem 2. Experiment with these filters Using the code you have written implement these filters on the test images. **Remember the output of all these filters, can be compared with matlab, so if you are unsure, of your implementation, compare the result with** `output=imfilter(image,filter,'replicate');`

- Smoothing Filters: *a*) Box Filter *b*) Weighted Box Filter *c*) Horizontal Filter *d*) Vertical Filter *e*) Circular-Shape Filter *f*) 5 by 5 Box Filter *g*) 5 x 5 circular shape filter , Try this filters out on the images below in Figure 1

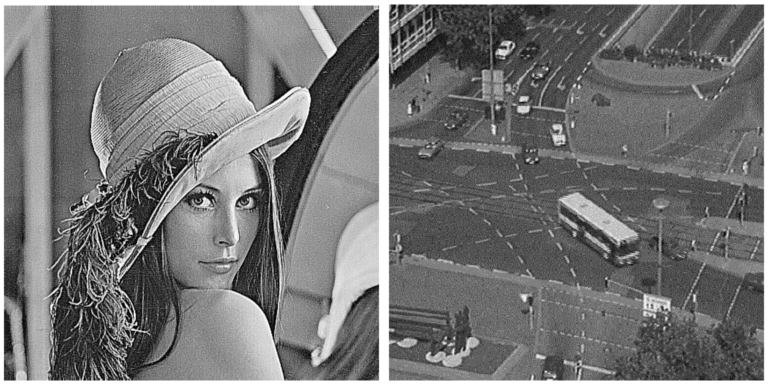

Figure 1: Images to Test on a) Very Sharp Image b) Degraded by noise

$$\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} a) \quad \frac{1}{16}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix} b) \quad \frac{1}{3}\begin{bmatrix}0&0&0\\1&1&1\\0&0&0\end{bmatrix} c) \quad \frac{1}{3}\begin{bmatrix}0&1&0\\0&1&0\\0&1&0\end{bmatrix} d)$$

$$\frac{1}{5}\begin{bmatrix}0&1&0\\1&1&1\\0&1&0\end{bmatrix} e) \quad \frac{1}{25}\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix} f) \quad \frac{1}{25}\begin{bmatrix}0&1&1&1&0\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\0&1&1&1&0\end{bmatrix} g)$$

- Enhancment Filters: *a*) 4 by 4 Laplacian *b*) 8 by 8 Laplacian, Use the images in Figure 2 to test out your filters

$$\begin{bmatrix}0&1&0\\1&-4&1\\0&1&0\end{bmatrix} a) \quad \begin{bmatrix}1&1&1\\1&-8&1\\1&1&1\end{bmatrix} b)$$

- Edge Filters: *a*) Prewitt X *b*) Prewitt Y *c*) Sobel X *d*) Sobel Y , Use the images in Figure 2

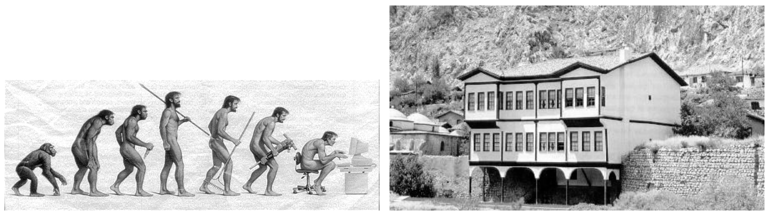

Figure 2: Images to Test on for enhancement and edge

$$\begin{bmatrix}1 & 0 & -1\\1 & 0 & -1\\1 & 0 & -1\end{bmatrix} a) \quad \begin{bmatrix}1 & 1 & 1\\0 & 0 & 0\\-1 & -1 & -1\end{bmatrix} b) \quad \begin{bmatrix}1 & 0 & -1\\2 & 0 & -2\\1 & 0 & -1\end{bmatrix} c) \quad \begin{bmatrix}1 & 2 & 1\\0 & 0 & 0\\-1 & -2 & -1\end{bmatrix} d)$$

When writing your report please discuss the following

- Smoothing Filters: Give some general comments on each of the filters, and what you observe. Which filter performs the best on each image? What is the difference between the box filter, and the weighted box filter?
- Enhancement Filters: Give some general comments on what this filter does to the image. Which filter does the best on each image?
- Edge Filters: Which filters performs the best on each image? What effect does weighting the center pixel have on the performance? What difference do you notice between the enhancement filters and edge filters?

Problem 3. Gaussian Smoothing The Gaussian is a very special function, and we will look at how to define kernels, using the Gaussian. You will have to sample the Gaussian. In this case we will use a zero mean symmetric Gaussian, so the formula, Equation 2, you will be using

$$f(x,y) = e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)} \tag{2}$$

where σ is the standard deviation.

Implement the function

```
function [filter]=gauss_kernel(filter_size,sigma);
```

- You will have to sample the Gaussian on a grid size equivalent to your window , for example, if your window size is 3, you will evaluate your Gaussian at x points -1,0,1 and y points -1, 0, 1, You can use the `meshgrid` function for help with this
- Remember to normalize your filter by the sum of all the elements in the filter
- You can check your answer, with matlab, `H = fspecial('gaussian',HSIZE,SIGMA)`, your filter should be exactly the same as matlab

- Report your results with these combinations, on the images in Figure 1
    1. Window Size of 3, $\sigma = 0.5,1,2$
    2. Window Size of 5, $\sigma = 0.5,1,2$
    3. Window Size of 7, $\sigma = 0.5,1,2$
- Discuss what is the effect of increasing window size? What is the effect of increasing sigma? How does the Gaussian compare to the other smoothing filters you tried?

Problem 4. Non-Linear Filters Again using the same code you developed for 2D convolution , you will implement some non-linear filters. Here we are dealing with order-statistic filters. Unlike , the smoothing filters you have seen before, now we will be looking at non-linear filters. In each region, you will return either the median, max, or min, rather than summing the values within the window of the filter. Experiment with the image in Figure 3

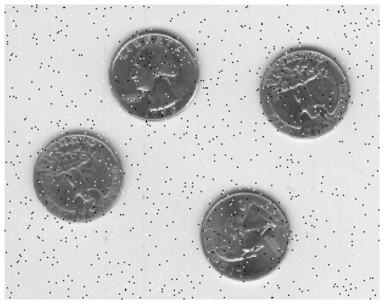

Figure 3: Noisy Coins

- Min Filtering - Use a 3x3,5x5, and 7x7 window, and the pixel value is equal to the minimum value in the window
- Max Filtering - Use a 3x3,5x5, and 7x7 window, and the pixel value is equal to the maximum value in the window
- Median Filtering - Use a 3x3,5x5, and 7x7 window, and the pixel value is equal to the median value in the image

Report the best setting for each type of filter. Out of the three types of filter which performed the best.

Problem 5. Salt and Pepper vs White Noise Two of the images, Figure 4, you have used so far were degraded by noise. But each has a different type of noise, the traffic image is degraded by a diffused more continuous distribution type **white noise**, and the coin image is contaminated by **salt and pepper noise**.

- Go back and run Gaussian Smoothing (pick your own window size, and sigma), and pick one of the box filters and run it on the noisy coin image

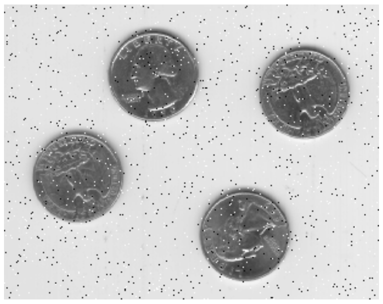
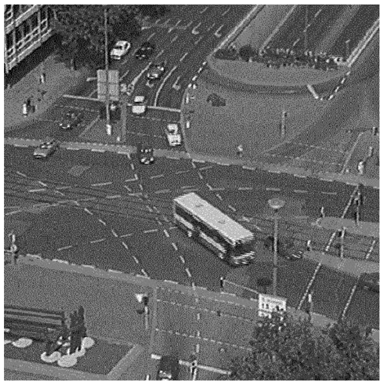

Figure 4: Images Displaying Different Type of Noise

- Go back and run the median, max, and min (pick the best window size) filter on the noisy traffic image
- Which type of filter is best for white noise, and why? Which type of filter is best for salt and pepper noise, and why?

Problem 6. Separable Convolution 2D Certain filters have the special property that they are separable. What this means, is that matrix (M by N) defining the filter, can be decomposed into (Mx1)and (1xN) matrices where the outer product returns the original. A 3 by 3 averaging filter, Equation 3, is a separable filter. Gaussian smoothing is another well-known separable matrix.

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix} = \begin{bmatrix}\frac{1}{3}\\\frac{1}{3}\\\frac{1}{3}\end{bmatrix}\begin{bmatrix}\frac{1}{3} & \frac{1}{3} & \frac{1}{3}\end{bmatrix} \tag{3}$$

As a result, in order to reduce the computation, we perform 1D convolution twice instead of 2D convolution; convolve with the input and M1 kernel in vertical direction, then convolve again horizontal direction with the result from the previous convolution and 1N kernel. The first vertical 1D convolution requires M times of multiplications and the horizontal convolution needs N times of multiplications, altogether, M+N products.

However, the separable 2D convolution requires additional storage (buffer) to keep intermediate computations. That is, if you do vertical 1D convolution first, you must preserve the results in a temporary buffer in order to use them for horizontal convolution subsequently.

Notice that convolution is associative; the result is same, even if the order of convolution is changed. So, you may convolve horizontal direction first then vertical direction later.

Implement the function

```
function [output]=conv_separable(image,filter)
```

The convolution of an NN image with an MM filter kernel requires a time proportional to N2M2. In other words, each pixel in the output image depends on all the pixels in the filter kernel. In comparison, convolution by separability only requires a time proportional to N2M. For filter kernels that are hundreds of pixels wide, this technique will reduce the execution time by a factor of hundreds.Use the matlab `tic,toc` command and compare the times.

```
tic
function [output]=conv_separable(image,filter)
toc
```

- Which filters in Problem 2 are separable? Describe a way to test whether a filter is separable?
- Compare the times for these filters again with your separable and brute force approach to convolution, use the image of the women (Lenna) in Figure 1

1. 5 by 5 Box Filter

$$\frac{1}{25}\begin{bmatrix}1 & 1 & 1 & 1 & 1\\1 & 1 & 1 & 1 & 1\\1 & 1 & 1 & 1 & 1\\1 & 1 & 1 & 1 & 1\\1 & 1 & 1 & 1 & 1\end{bmatrix} \tag{4}$$

2. 5 x 5 circular shape filter

$$\frac{1}{25}\begin{bmatrix}0 & 1 & 1 & 1 & 0\\1 & 1 & 1 & 1 & 1\\1 & 1 & 1 & 1 & 1\\1 & 1 & 1 & 1 & 1\\0 & 1 & 1 & 1 & 0\end{bmatrix} \tag{5}$$

3. 7 x 7 Gaussian with sigma of 0.5
4. 17 x 17 Gaussian with sigma of 0.5
5. 35 x 35 Gaussian with sigma of 2

Image Restoration

A model for image degradation can be seen in Equation 6

$$g = h \otimes f + n \tag{6}$$

where f is the original undistorted image, g is the distorted noisy image, h is the PSF of the system, $\otimes$ is the convolution operator, and n is the corrupting noise.

One method to recover this, is by using the Lucy-Richardson algorithm. This iterative algorithm can be succinctly expressed as 7

$$\hat{f}_{k+1} = \hat{f}_k \left(h * \frac{g}{h \otimes \hat{f}_k} \right) \tag{7}$$

where $\hat{f}_k$ is the estimate of f after k iterations, $*$ is the correlation operator. The image $h \otimes \hat{f}_k$ is referred to as the reblurred image.

Problem 7. Deblurring- Lucy Richardson Implement this algorithm

```
function [restored_image]=lucy_richardson(degraded_image,psf);
```

Your results might vary, but an example of what you should get is in the Figure below 5

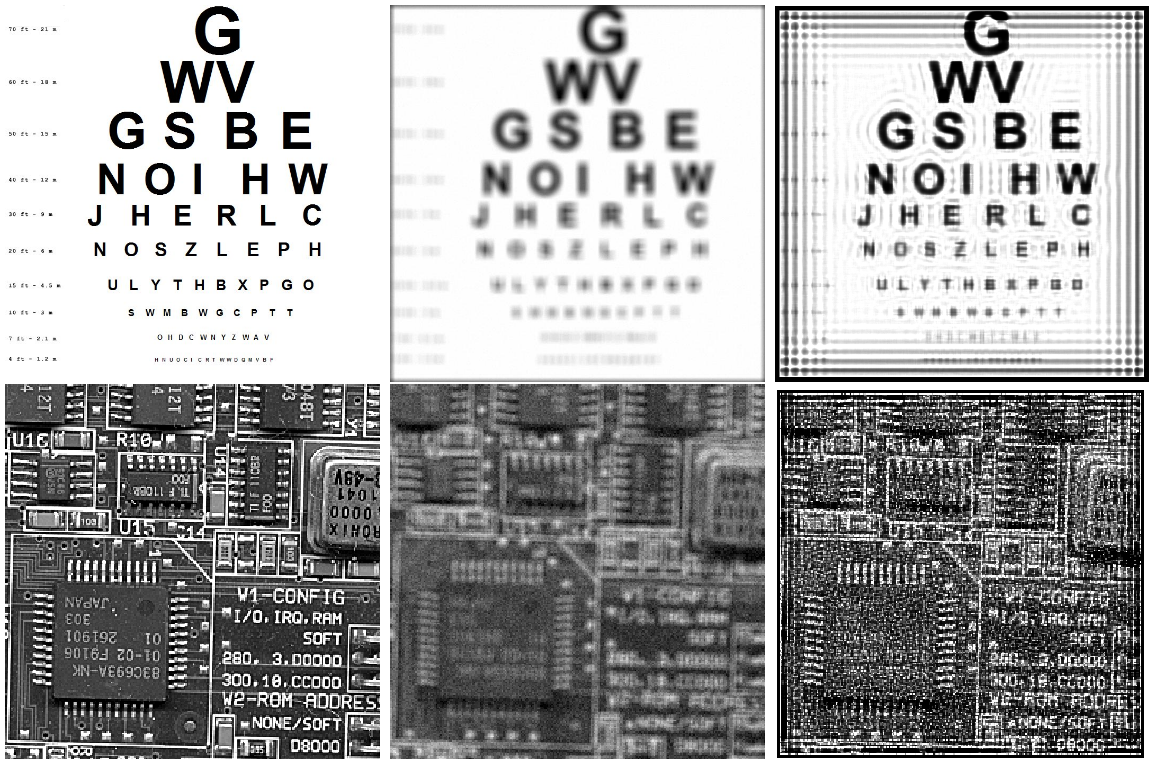

Figure 5: a) Original Image b) Degraded Image c) Restored Image Using Lucy Richardson

a)
- For the first iteration $\hat{f}_1$ just set it to the degraded image g, or you can try your own initialization
- For the convolution operation in Equation 7 you can use matlabs conv2
- For the correlation operation you can use matlab's `filter2` function
- You can download the psf (point spread function) off the class website, use the `load(<MAT_FIlE>)` to load the .mat file
- Try various number of iterations, in general it will get better the longer you go

b) Compare your results, against matlab's lucy richardson implementation, `help deconvlucy`, Show images from both your implementation and matlab's

c) Now try restoring the two degraded images, using the median filter you wrote earlier, try various, window size. Compare those results to the lucy richardson algorithm. Also try any of the sharpening filters you implemented, and see what you get.

Challenge Looking at the restored images, you see that they display some ringing. What causes this? What can be done to deal with this? Submit your results with your solution.