Image Processing Lab

Batch 9:

Ajay Viswanathan [09EC3509]

Sourin Sutradhar [09EC3514]

Experiment 1

Spatial Filtering

Problem Objective:

Write C/C++ modular function/subroutine to design spatial filters (mean, median, gradient, Laplacian, all Sobel kernals) on a stack of grayscale images (at least 10 images per stack)

(Use OpenCV (or) ImageJ for image reading, writing and GUI development only. Use the OpenCV tracker functionality.)

(a) Input: Path to the stack of images

(b) Output: Filtered stack of images to be shown beside input stack in the same pane of GUI

Brief Theory:

A spatial filter is an image operation where each pixel value I(u,v) is changed by a function of the intensities of pixels in a neighborhood of (u,v). Spatial filtering is a technique for modifying or enhancing an image. For example, you can filter an image to emphasize certain features or remove other features. Image processing operations implemented with filtering include smoothing, sharpening, and edge enhancement. Filtering is a neighborhood operation, in which the value of any given pixel in the output image is determined by applying some algorithm to the values of the pixels in the neighbourhood of the corresponding input pixel. A pixel's neighborhood is some set of pixels, defined by their locations relative to that pixel.

Mean Filtering

The output (response) of a smoothing, linear spatial filter is simply the average of the pixels contained in the neighborhood of the filter mask. These filters sometimes are called averaging filters. They also are referred to a lowpass filters. The idea behind smoothing filters is straightforward. By replacing the value of every pixel in an image by the average of the gray levels in the neighborhood defined by the filter mask, this process results in an image with reduced "sharp" transitions in gray levels. Because random noise typically consists of sharp transitions in gray levels, the most obvious application of smoothing is noise reduction. However, edges (which almost always are desirable features of an image) also are characterized by sharp transitions in gray levels, so averaging filters have the undesirable side effect that they blur edges.

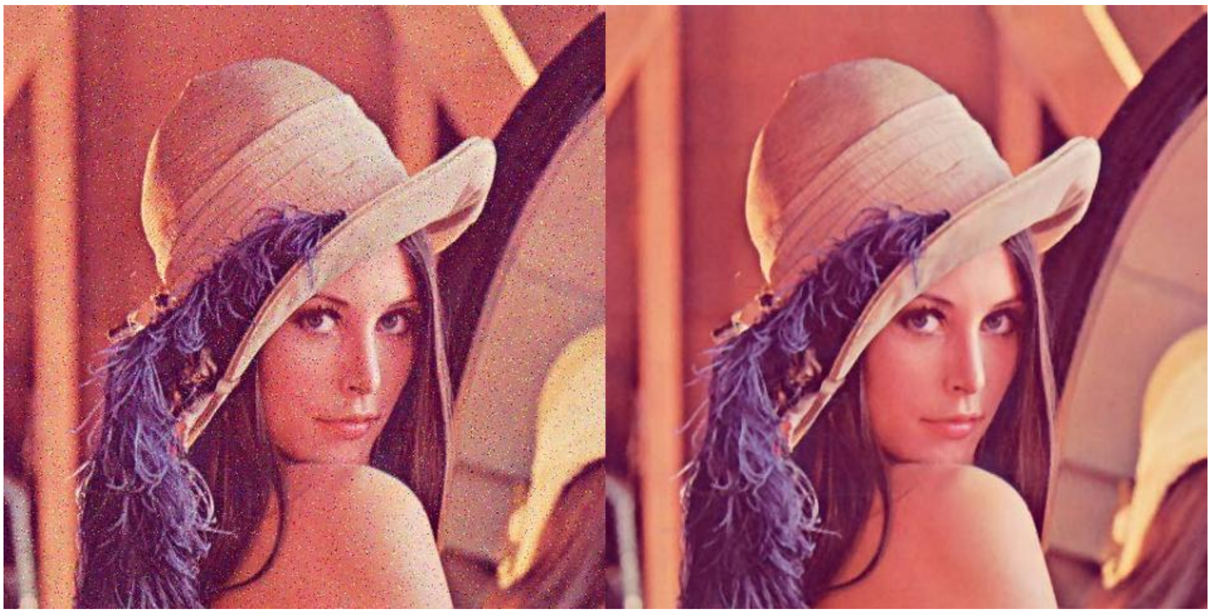
A major use of averaging filters is in the reduction of “irrelevant” detail in an image. By “irrelevant” we mean pixel regions that are small with respect to the size of the filter mask.

$$\frac{1}{9}\times\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$

$$R = \frac{1}{9}\sum_{i=1}^{9} z_i$$

Median Filtering

In signal processing, it is often desirable to be able to perform some kind of noise reduction on an image or signal. The median filter is a nonlinear digital filtering technique, often used to remove noise. Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection on an image). Median filtering is very widely used in digital image processing because, under certain conditions, it preserves edges while removing noise.



The main idea of the median filter is to run through the signal entry by entry, replacing each entry with the median of neighboring entries. The pattern of neighbors is called the "window", which slides, entry by entry, over the entire signal. For 1D signals, the most obvious window is just the first few preceding and following entries, whereas for 2D (or higher-dimensional) signals such as images, more complex window patterns are possible (such as "box" or "cross" patterns). Note that if the window has an odd number of entries, then the median is simple to define: it is just the middle value after all the entries in the window are sorted numerically. For an even number of entries, there is more than one possible median.

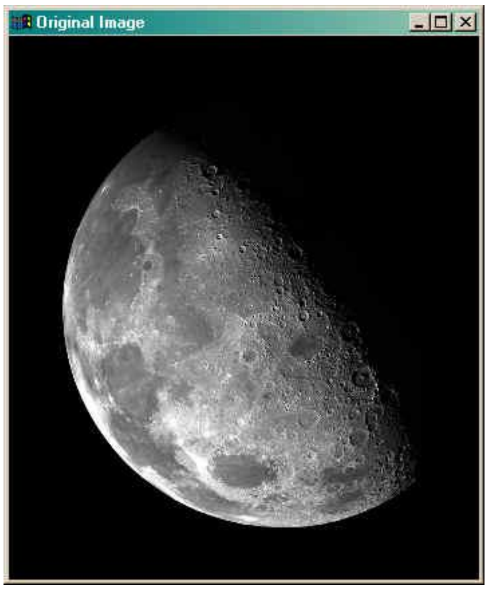
To demonstrate, using a window size of three with one entry immediately preceding and following each entry, a median filter will be applied to the following simple 1D signal:

x = [2 80 6 3]
So, the median filtered output signal y will be:
y[1] = Median[2 2 80] = 2
y[2] = Median[2 80 6] = Median[2 6 80] = 6
y[3] = Median[80 6 3] = Median[3 6 80] = 6
y[4] = Median[6 3 3] = Median[3 3 6] = 3
Hence, y = [2 6 6 3].

Laplacian Filtering

The Laplacian operator is an example of a second order or second derivative method of enhancement. It is particularly good at finding the fine detail in an image. Any feature with a sharp discontinuity (like noise, unfortunately) will be enhanced by a Laplacian operator. Thus, one application of a Laplacian operator is to restore fine detail to an image which has been smoothed to remove noise. (The median operator is often used to remove noise in an image.)

Consider the original 2D byte value image below.



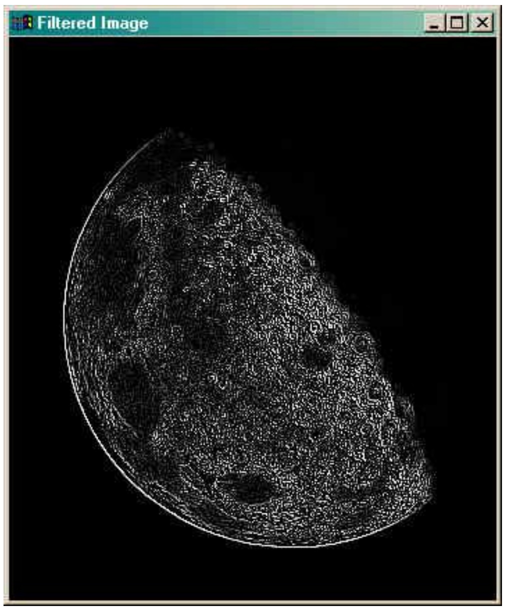
The Laplacian operator is implemented in IDL as a convolution between an image and a kernel. The Convol function is used to perform the convolution. The Laplacian kernel can be constructed in various ways, but we will use the same 3-by-3 kernel. Since the input image is represented as a set of discrete pixels, we have to find a discrete convolution kernel that can approximate the second derivatives in the definition of the Laplacian. Two commonly used small kernels are shown.

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

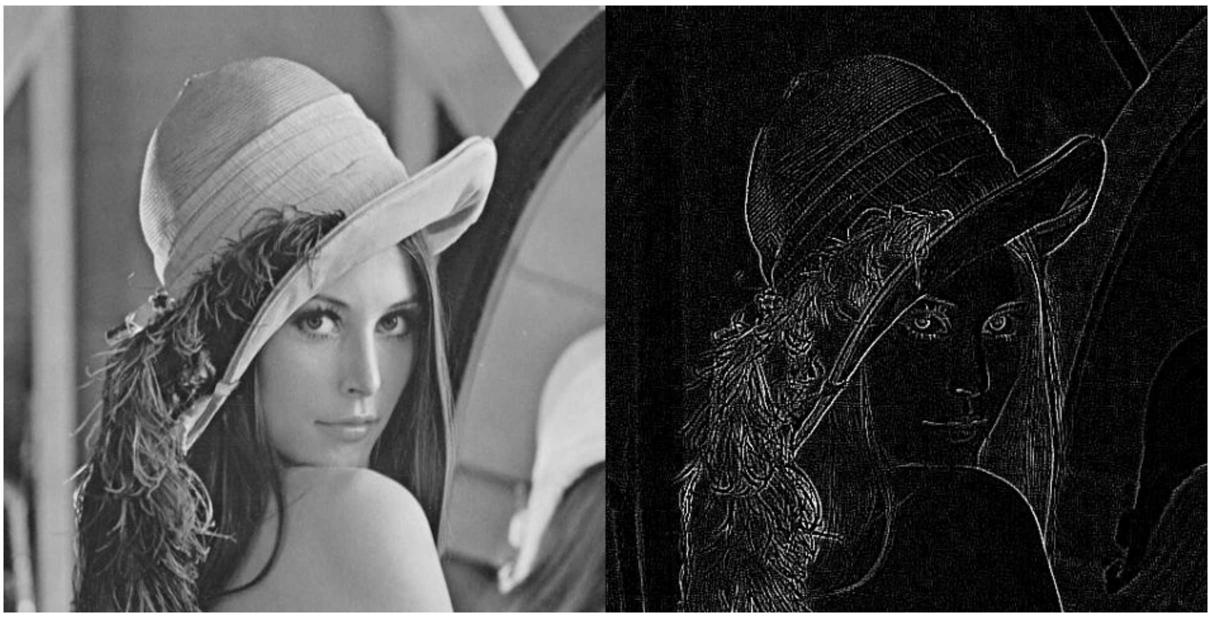
In image convolution, the kernel is centered on each pixel in turn, and the pixel value is replaced by the sum of the kernel multiplied by the image values. In the particular kernel we are using here, we are counting the contributions of the diagonal pixels as well as the orthogonal pixels in the filter operation. This is not always necessary or desirable, although it works well here.

The results are shown in the figure below.



Note the amount of fine detail found in the image with this filter. What we want to do is add this information back into the original image in order to sharpen the image. To do this properly, we have to scale the filtered data into the same 0 to 255 values of the original image.

Another example:



Sobel Filtering

The Sobel operator is used in image processing, particularly within edge detection algorithms. Technically, it is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. At each point in the image, the result of the Sobel operator is either the corresponding gradient vector or the norm of this vector. The Sobel operator is based on convolving the image with a small, separable, and integer valued filter in horizontal and vertical direction and is therefore relatively inexpensive in terms of computations. On the other hand, the gradient approximation that it produces is relatively crude, in particular for high frequency variations in the image.

In simple terms, the operator calculates the gradient of the image intensity at each point, giving the direction of the largest possible increase from light to dark and the rate of change in that direction. The result therefore shows how "abruptly" or "smoothly" the image changes at that point, and therefore how likely it is that that part of the image represents an edge, as well as how that edge is likely to be oriented. In practice, the magnitude (likelihood of an edge) calculation is more reliable and easier to interpret than the direction calculation.

Mathematically, the gradient of a two-variable function (here the image intensity function) is at each image point, a 2D vector with the components given by the derivatives in the horizontal and vertical directions. At each image point, the gradient vector points in the direction of largest possible intensity increase, and the length of the gradient vector corresponds to the rate of change in that direction. This implies that the result of the Sobel operator at an image point which is in a region of constant image intensity is a zero vector and at a point on an edge is a vector which points across the edge, from darker to brighter values.

The operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives - one for horizontal changes, and one for vertical. If we define A as the source image, and Gx and Gy are two images which at each point contain the horizontal and vertical derivative approximations, the computations are as follows:

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A}$$

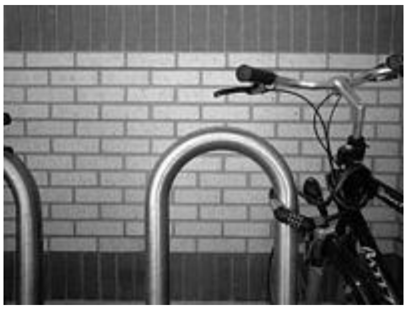
At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude, using:

$$\mathbf{G} = \sqrt{{\mathbf{G}_x}^2 + {\mathbf{G}_y}^2}$$

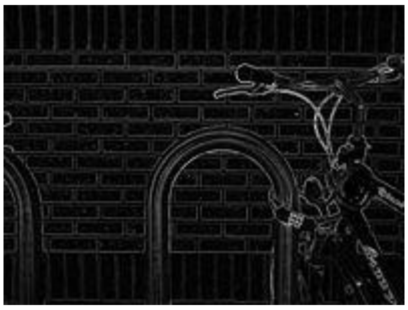
Using this information, we can also calculate the gradient's direction:

$$\mathbf{\Theta} = \operatorname{atan}\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$$

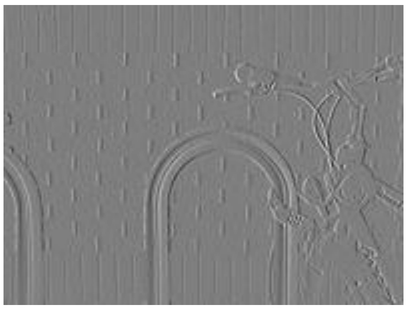
An example:



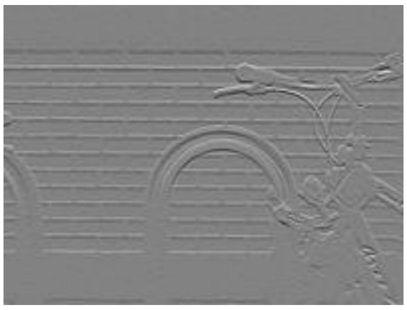
Original Image



Normalized gradient magnitude from Sobel operator

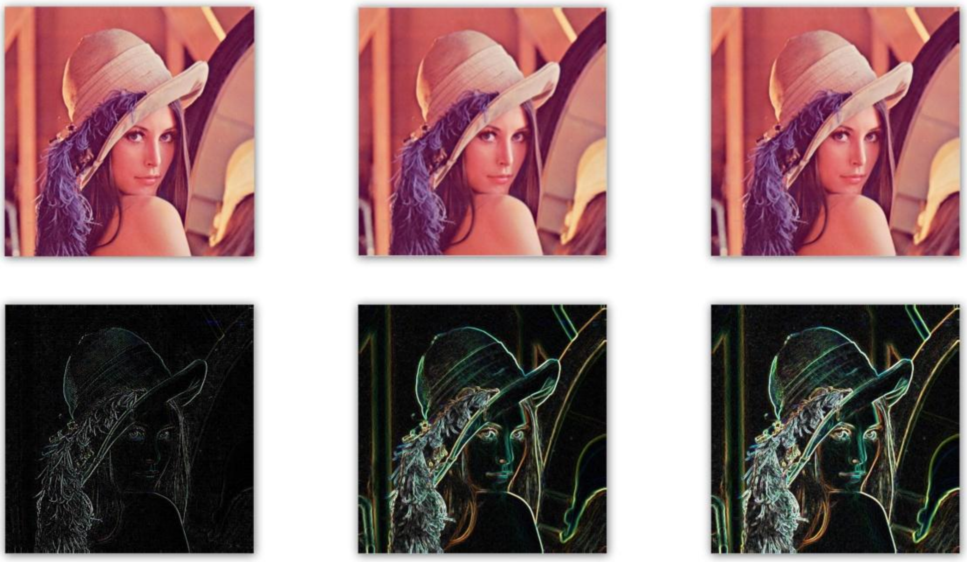


Normalized x-gradient from Sobel operator

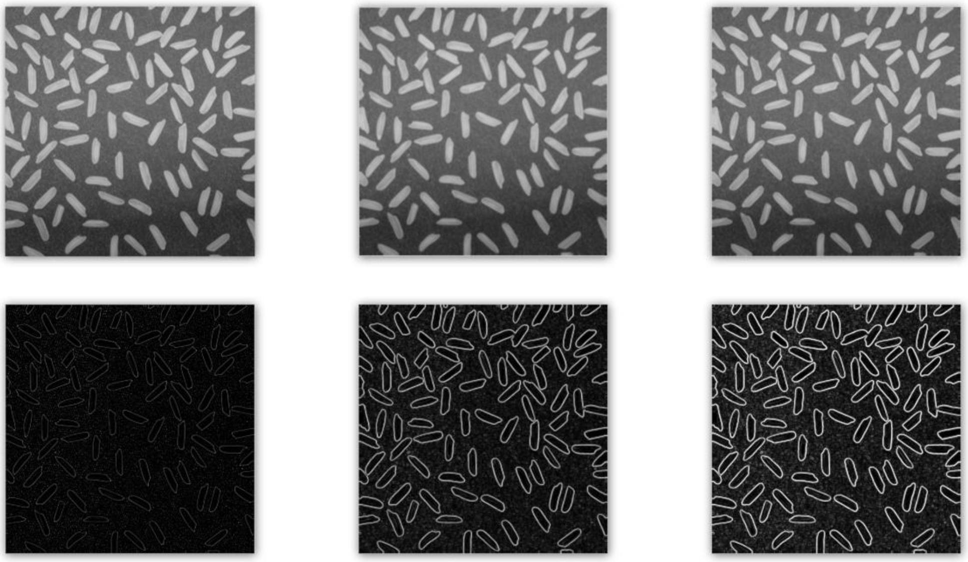


Normalized y-gradient from Sobel operator

Results:



(L-R) Original, Mean, Median, Laplacian, Gradient, Sobel



(L-R) Original, Mean, Median, Laplacian, Gradient, Sobel

Observation and Inference:

1) Median filtering is one kind of smoothing technique, as is linear Gaussian filtering. All smoothing techniques are effective at removing noise in smooth patches or smooth regions of a signal, but adversely affect edges. Often though, at the same time as reducing the noise in a signal, it is important to preserve the edges. Edges are of critical importance to the visual appearance of images, for example. For small to moderate levels of (Gaussian) noise, the median filter is demonstrably better than Gaussian blur at removing noise whilst preserving edges for a given, fixed window size. However, its performance is not that much better than Gaussian blur for high levels of noise, whereas, for speckle noise and salt and pepper noise (impulsive noise), it is particularly effective. Because of this, median filtering is very widely used in digital image processing.
2) Median filtering is more effective than mean and Gaussian filter for removing random noise.
3) For gradient and sobel filters, adding the root of the squares of the x and y values give a better enhanced result.
4) Gradient detects thick edges, laplacian detects 2nd order finer edges, while sobel detects both with better highlighting of thinner edges.
5) The Sobel operator, while reducing artifacts associated with a pure central differences operator, does not have perfect rotational symmetry. Scharr looked into optimizing this property. Filter kernels up to size 5 x 5 have been presented there, but the most frequently used one is:

$$\begin{bmatrix} +3 & +10 & +3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix} \qquad \begin{bmatrix} +3 & 0 & -3 \\ +10 & 0 & -10 \\ +3 & 0 & -3 \end{bmatrix}$$

Scharr operators result from an optimization minimizing weighted mean squared angular error in Fourier domain. This optimization is done under the condition that resulting filters are numerically consistent. Therefore they really are derivative kernels rather than merely keeping symmetry constraints.