QUANSER
INNOVATE·EDUCATE

Concept Review

Image Filters

What is Image Filtering?

Images map 3-dimensional real world information into a 2-dimension plane for viewing. The inherent nature of light as well as the camera sensor introduces noise in the images. Removing this noise is vital to extracting useful information from the image, which is done using image filters. Additional filters can extract useful information by using morophological operations, to detect vertical or horizontal lines, edges, etc. This document will first cover the idea behind convolution, and then introduce a variety of image filters, both spatial and temporal.

Convolution

With images represented as matrices, filters are applied using the mathematical convolution operation. Consider an image I with 640 columns and 480 rows ($480p$). Consider a kernel k (small matrix representing our operation) with p columns and p rows. Convolution then involves applying the (p, p) kernel to numerous (p, p) subsections of the entire $(640, 480)$ image I using element wise multiplication.

$$k = \begin{bmatrix} W_{-1,-1} & W_{0,-1} & W_{+1,-1} \\ W_{-1,0} & W_{0,0} & W_{+1,0} \\ W_{-1,+1} & W_{0,+1} & W_{+1,+1} \end{bmatrix} \tag{1}$$

Where k is a $(3, 3)$ kernel in the example shown, but could generally be (p, p) where p is typically odd. If $I_{i,j}$ is the (p, p) subsection of image I centered around row i and col j, then the convolution operator replaces the value in the output image I^O at row i and col j based on the following,

$$\begin{aligned} I^O_{i,j} &= \begin{bmatrix} I_{i-1,j-1} & I_{i,j-1} & I_{i+1,j-1} \\ I_{i-1,j} & I_{i,j} & I_{i+1,j} \\ I_{i-1,j+1} & I_{i,j+1} & I_{i+1,j+1} \end{bmatrix} \otimes \begin{bmatrix} W_{-1,-1} & W_{0,-1} & W_{+1,-1} \\ W_{-1,0} & W_{0,0} & W_{+1,0} \\ W_{-1,+1} & W_{0,+1} & W_{+1,+1} \end{bmatrix} \\ &= \begin{bmatrix} I_{i-1,j-1}W_{-1,-1} & I_{i,j-1}W_{0,-1} & I_{i+1,j-1}W_{+1,-1} \\ I_{i-1,j}W_{-1,0} & I_{i,j}W_{0,0} & I_{i+1,j}W_{+1,0} \\ I_{i-1,j+1}W_{-1,+1} & I_{i,j+1}W_{0,+1} & I_{i+1,j+1}W_{+1,+1} \end{bmatrix} \end{aligned} \tag{2}$$

Varying kernel shapes and sized produce a variety of effects on images. Spatial filters are primarily based on selecting unique kernel shapes to amplify or extract certain morphological characteristics found in the input images.

Spatial Filters

Basic Morphological Filters

These filters do not use a kernel as such, but apply a function $f_k()$ to the image subset

$$I^{O}_{i,j} = f_k\left(\begin{bmatrix} I_{i-1,j-1} & I_{i,j-1} & I_{i+1,j-1} \\ I_{i-1,j} & I_{i,j} & I_{i+1,j} \\ I_{i-1,j+1} & I_{i,j+1} & I_{i+1,j+1} \end{bmatrix}\right) \tag{3}$$

Minimum

As the name suggests, the function f_k selected here is the minimum function. This sets a value to the lowest (or darkest) pixel value found in its immediate neighborhood. This can be used to remove small/thin blobs/features in an image as shown below.

Median

The function f_k sets the output value to the median of the input values. This tends to preserve overall shape and morphology and is very useful for removing noise as shown below.

Maximum

The function f_k sets the output value to the maximum of the neighborhood input values. This amplifies small/thin blobs/features as shown below.

Bitonal Morphological Filters

These next few filters are typically applied to bitonal images (black and white only), which are typically the result of some boolean criterion. For example, thresholding a grayscale image for a certain color can be done using the following comparison,

$$I_{ij}^{O} = \begin{cases} 255 \; or \; \text{TRUE} & I_{ij} > t \\ 0 \; or \; \text{FALSE} & I_{ij} \leq t \end{cases}$$

Where the parameter t is between 0 and 255. As a result, the output image I^O is bitonal.

Erosion

Similar to the minimum filter, the function f_k sets the output pixel value to FALSE if at least one of the pixels in the input neighborhood or subsection is FALSE. This tends to make blobs/features in the image smaller. Use this filter to remove small particulates or noise in the image. Note that the noise in the original image disappears, but the holes get bigger.

Dilation

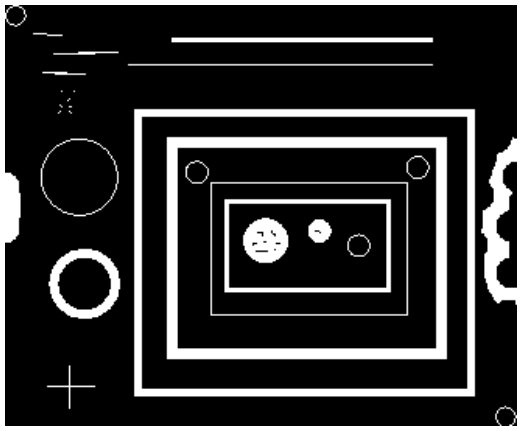
Similar to the maximum filter, the function f_k sets the output pixel value to TRUE if at least one of the pixels in the input neighborhood or subsection is TRUE. This tends to make blobs/features in the image larger. Use this filter to fill holes in large objects that are valid. Notice that the holes in the original image disappear, but the noise is amplified too.

Opening

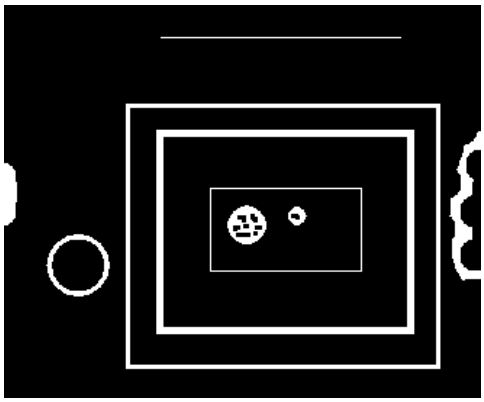
To preserve overall morphology and size of objects while removing noise, you can use a Erosion filter followed by a Dilation filter. This is called Opening an image. This will first remove noise, but also dilate the shrunk objects to bring them back to the original size.

Closing

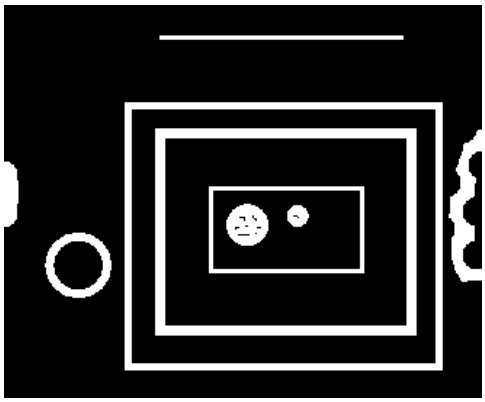
To preserve overall morphology and size of objects while still filling holes, you can use a Dilation filter followed by an Erosion filter. This is called Closing an image. This will first fill holes, but then shrink the enlarged objects to bring them back to the original size.



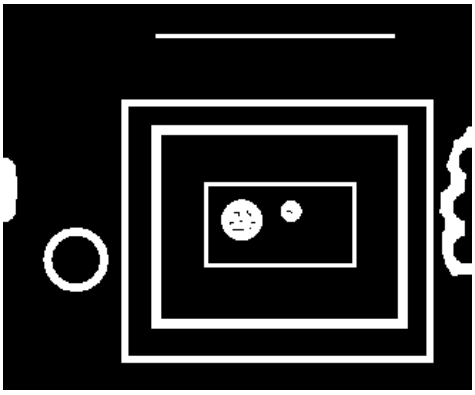
original



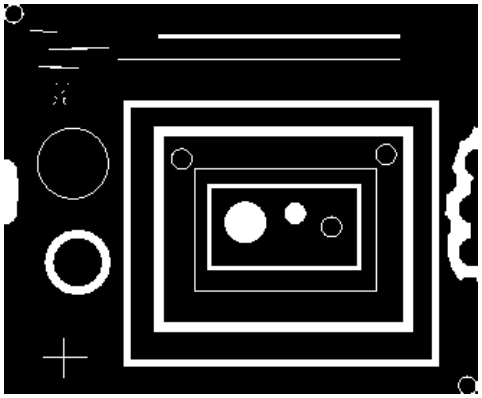
eroded



dilated



opened



closed

Blurring

Blurring filters smooth out rapid changes in pixel intensities by averaging over the kernel grid. This represents optical blur very naturally, and blurring the image is often the very first step in any image processing pipeline.

Low-pass

A low-pass kernel looks like the following,

$$k = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

As such, this filter applies a linear combination of the pixels in the neighborhood subsection of the input image, smoothing out pretty evenly.

Guassian

The Guassian filter is another blurring filter that uses a Guassian kernel,

$$k = \begin{bmatrix} 0 & 1/8 & 0 \\ 1/8 & 1/2 & 1/8 \\ 0 & 1/8 & 0 \end{bmatrix}$$

The filter is not as harsh as low-pass filter and prioritizes the central pixels far more than the edge pixels.

Original image | Low-pass applied | Guassian filter applied

Sharpening

As opposed to blurring, sharpening an image amplifies variations in pixel intensities in a neighborhood. To do this, a filter that can highlight edges is first used (opposite of blurring), and then the result is added to the source image to sharpen it.

The filter to highlight edges and changing pixel intensities typically consists of a positive value at a point of interest and negative values in other parts of the kernel so as to highlight the area of interest the most. This ends up enhancing edges and other discontinuities (noise) in the image, while undermining areas with slowly varying values.

High-pass

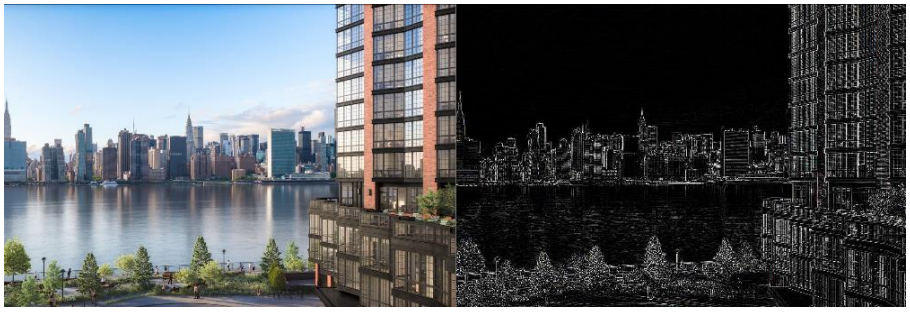
The simplest high pass filter that can be used is of the following form, which ends up highlighting the centre pixel and works like a first-order derivative filter.

$$k = \begin{bmatrix} -1/9 & -1/9 & -1/9 \\ -1/9 & +8/9 & -1/9 \\ -1/9 & -1/9 & -1/9 \end{bmatrix}$$

Laplacian

The Laplacian filter is to sharpening images as the Guassian filter is to blurring. It tends to not be as harsh, while representing the second derivative in the definition of a Laplacian.

$$k = \begin{bmatrix} 0 & -1/5 & 0 \\ -1/5 & +4/5 & -1/5 \\ 0 & -1/5 & 0 \end{bmatrix}$$



Original image High-pass applied

Original image Laplacian filter applied

Sharpening

Adding the high-frequency result from the filters above to the original yields a sharpened image. For the image on the right below, the Laplacian filtered image was added to the original.

Original image Sharpened image

Sobel

Another popular filter for detecting edges is the Sobel filter. It uses two kernels to detect edges in the x and y directions, and then combines the information to create an isotropic filter overall. The two kernels are,

$$k_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

$$k_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Applying the kernels to the input image I produces two gradient images,

$$G_x = k_x \otimes I$$

$$G_y = k_y \otimes I$$

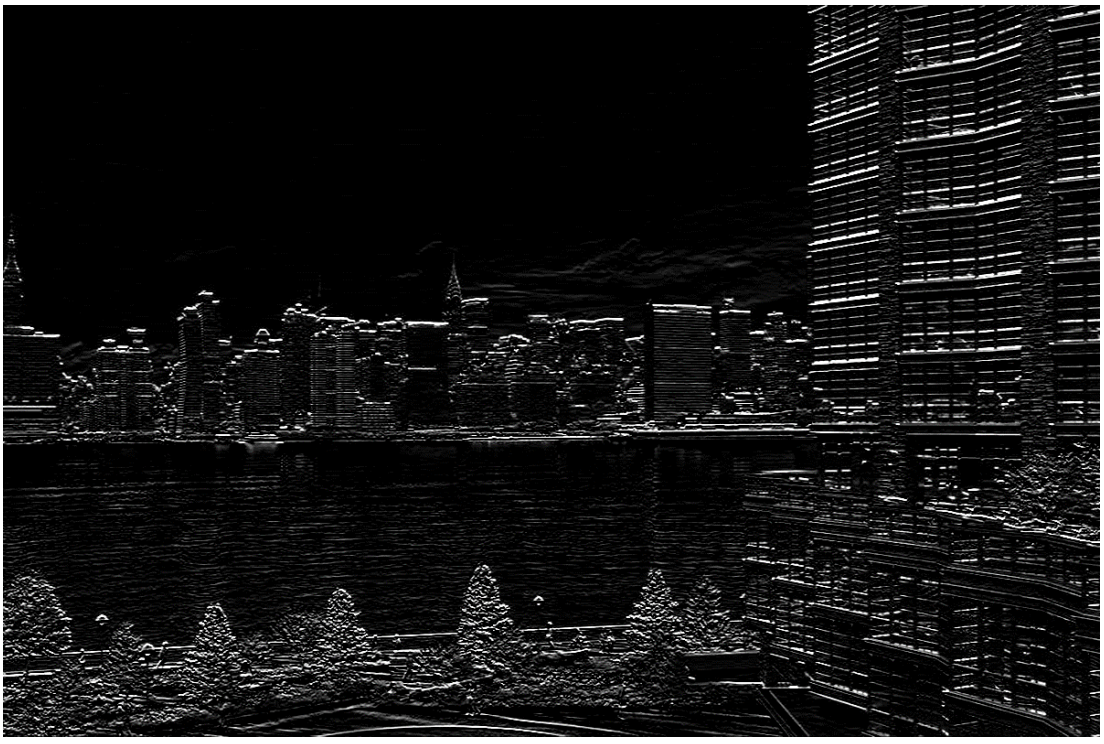
Combining the two images yields a margnitude and gradient image,

$$G = \sqrt{G_x^2 + G_y^2}$$

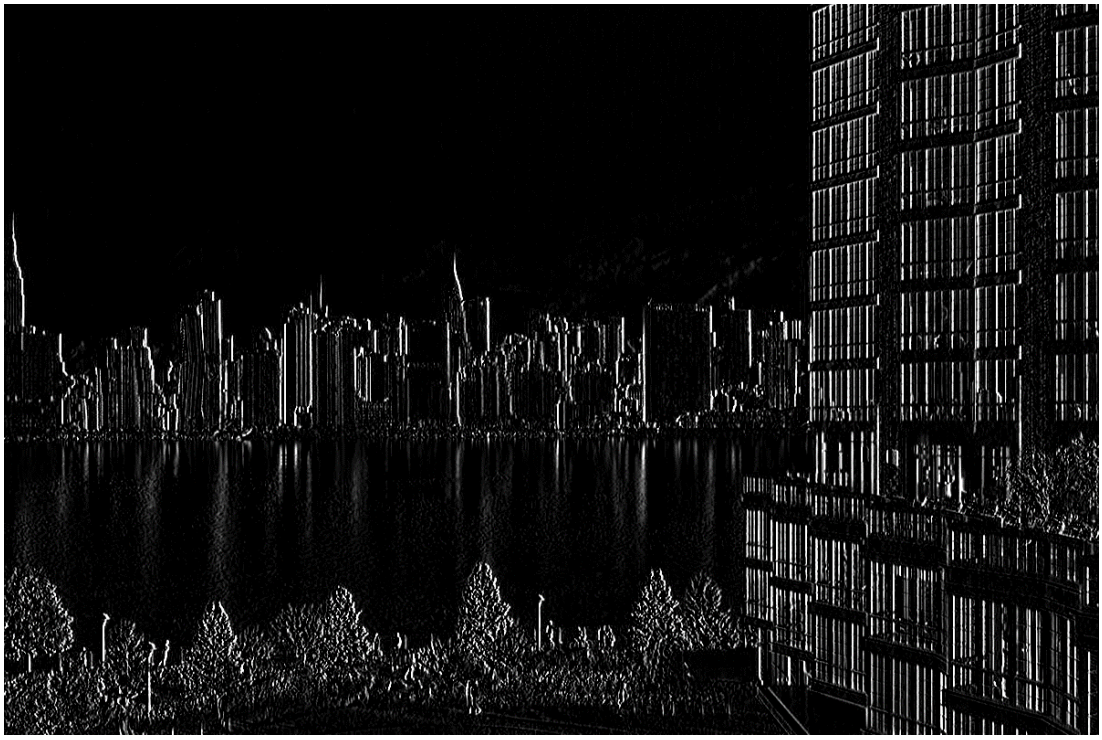
$$\Theta = \arctan(G_y \,/\, G_x)$$

The results of using this filter on a source image is shown below.

Grayscale original



Sobel horizontal gradient image G_x



Sobel vertical gradient image G_y

Sobel gradient image G

Quanser Inc.
119 Spy Court
Markham, Ontario
L3R 5H6
Canada

info@quanser.com
Phone: 19059403575
Fax: 19059403576
Printed in Markham, Ontario.

For more information on the solutions Quanser Inc. offers, please visit the web site at:
http://www.quanser.com