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 morphological 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.

What Are Image Filters and Convolution

Image filtering has a wide variety of applications, such as blurring artifacts, enhancing an image, reducing noise, extracting features etc. Filtering works by convolving a small matrix kernel with the image matrix. For example, let's take a 3x3 averaging kernel K in figure 1a, as well as a 5x5 image I whose pixel values and shading are also shown in figure 1b and figure 1c.

$$K = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \qquad I = \begin{bmatrix}50 & 0 & 0 & 50 & 100\\ 0 & 0 & 50 & 100 & 150\\ 100 & 50 & 50 & 100 & 200\\ 150 & 100 & 100 & 100 & 200\\ 250 & 200 & 50 & 50 & 250\end{bmatrix}$$

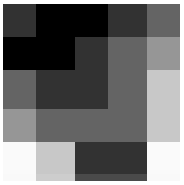


figure 1a figure 1b figure 1c

Figure 1 – 3x3 kernel and a 5x5 image

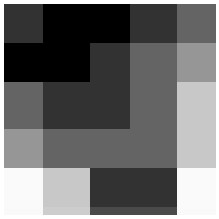
The 3x3 kernel window's center will sweep along the first row of the image, reach the end, and then move onto the next row, until all the rows have been covered. At each position, the 3x3 kernel does an element wise multiplication and overall sum with a subsection of the image matching the kernel size. For example, when the kernel center reaches the third row and third column, the kernel will be applied on the following 3x3 subset of the image I,

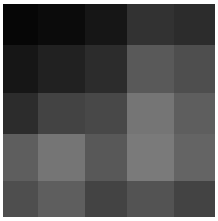
$$value = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} * \begin{bmatrix}0 & 50 & 100\\ 50 & 50 & 100\\ 100 & 100 & 100\end{bmatrix}$$
$$= \frac{1}{9}(1\cdot 0 + 1\cdot 50 + 1\cdot 100 + 1\cdot 50 + \ 1\cdot 50 + 1\cdot 100 + 1\cdot 100 + 1\cdot 100 + \ 1\cdot 100) = 72$$

The element wise multiplication and sum results in a value of 72, which will replace the value in the output at the third row and third column. This process can be summarized with the following equations. For a pxp kernel K, and mxn input image X, a resulting mxn output image Y is,

$$Y(m,n) = \sum_{i=-q}^{q}\sum_{j=-q}^{q} X(m+i, n+j) * K(q+1+i, q+1+j)$$

Where $q = floor(p/2)$ and p is typically an odd number (example, 3x3 or 5x5 kernels). Figure 2a and 2c show the input image X and output image Y when the averaging kernel from figure 1a is applied to X.





$$X = \begin{bmatrix} 50 & 0 & 0 & 50 & 100 \\ 0 & 0 & 50 & 100 & 150 \\ 100 & 50 & 50 & 100 & 200 \\ 150 & 100 & 100 & 100 & 200 \\ 250 & 200 & 50 & 50 & 250 \end{bmatrix} \qquad Y = \begin{bmatrix} 6 & 11 & 22 & 50 & 44 \\ 22 & 33 & 44 & 89 & 78 \\ 44 & 67 & 72 & 117 & 94 \\ 94 & 117 & 88 & 122 & 100 \\ 78 & 94 & 67 & 83 & 67 \end{bmatrix}$$

Figure 2a – input image Figure 2b – output image

Figure 2 – Applying an averaging kernel over an image

Figure 3. Low resolution image and average filter

Figure 3 shows a low-resolution image and the output after using the averaging kernel on it. We can see that it removes unwanted harsh edges from the image but creates more blurry text.

Equations

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,

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

$$= \begin{bmatrix} I_{i-1,j-1}W_{-1,-1} & I_{i,j-1}W_{0,-1} & I_{i+1,j-1}W_{+1,-1} \\ I_{i-1,j}W_{-1,0} & I_{i,j}W_{0,0} & I_{i+1,j}W_{+1,0} \\ I_{i-1,j+1}W_{-1,+1} & I_{i,j+1}W_{0,+1} & I_{i+1,j+1}W_{+1,+1} \end{bmatrix}$$

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

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

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 colour 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, 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.

This filter applies a kernel to remove small particles in an image, usually corresponding to noise. 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. A simpler implementation of an erosion filter is to create a minimum filter. For a minimum filter, each center pixel in the original image will be replaced by the smallest value in its neighborhood. This removes small noise pixels, but bigger objects will appear thinner and smaller because the boundaries might disappear. Figure 5 shows an image before and after using a 5x5 kernel for erosion

Figure 5. Performing erosion on an image

Dilation

In dilation, the opposite of erosion occurs. It is used to make desired features in bitonal images larger. 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. If there are holes within desired features, this filter can remove them. A simpler implementation of this filter is the maximum filter, which replaces a pixel in an image with the largest value in its neighborhood. It makes objects bigger and helps fill up holes in the image, but noise is amplified too. Figure 6 shows an image before and after using a 5x5 kernel for dilation

Figure 6. Performing dilation on an image

Opening

To preserve overall morphology and size of objects while removing noise, an opening operation can be used which applies a erosion filter followed by a dilation filter. The erosion removes image noise, however, shrinks the object of interest. The following dilation enlarges the object, preserving the size with respect to the input image prior to erosion. Figure 7 shows how the small objects disappear while still preserving the object of interest.

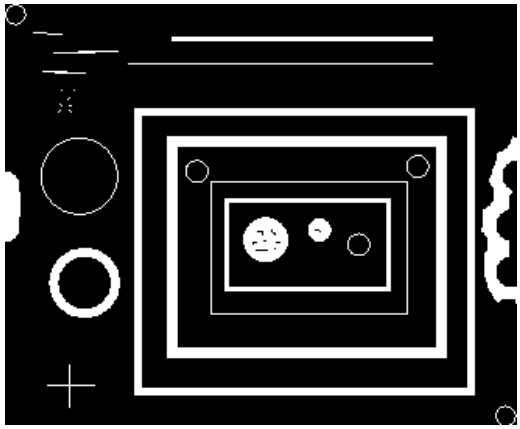
Figure 7. Opening operation on an image

Closing

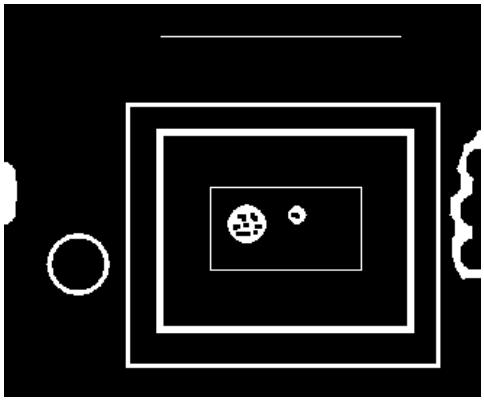
Closing applies a dilation filter followed by an erosion. This dilation removes image holes but enlarges the object of interest. The following erosion shrinks the object, preserving the size with respect to the input image prior to dilation. Figure 8 shows how the object maintains its original size while still having all the empty holes closed.

Figure 8. Performing closing on an image

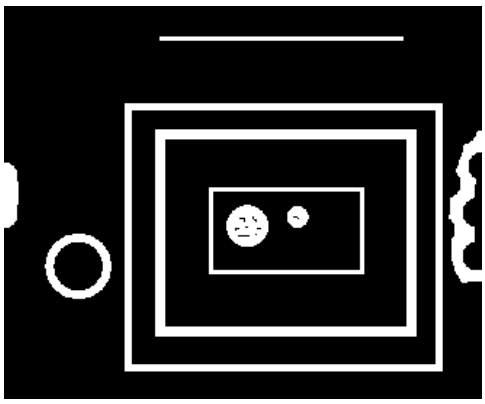
Examples



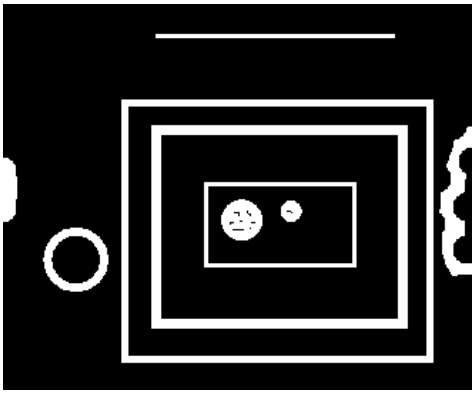
original



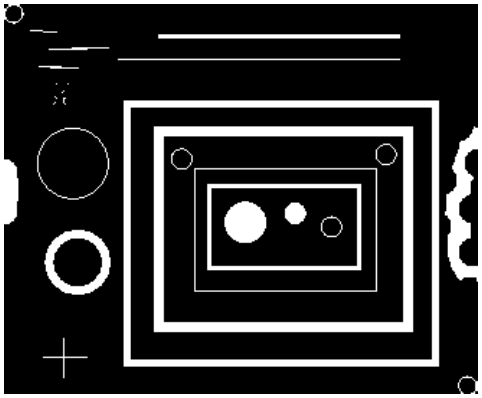
eroded



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.

Gaussian

A kernel that assists with blurring but has a greater focus on the center pixel is the Gaussian kernel. This is typically used to delete noise from an image. Unlike the averaging kernel which is uniform, it uses a Gaussian value spread; a 3x3 example is shown below. The bigger the kernel and its standard deviation, the more the image is smoothed out. Figure 4 shows the original low-resolution image and the output after using a Gaussian Kernel on it. Notice that the text is not as blurry as when doing an average filter.

$$K = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Figure 4. Low resolution image and Gaussian filter

Another example of a Gaussian 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 | Gaussian 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

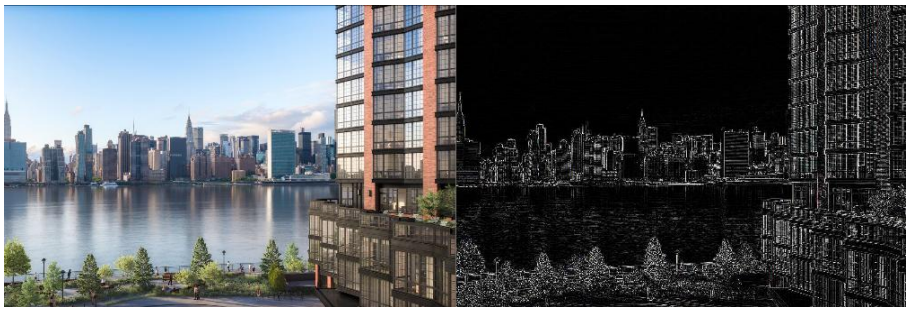
The simplest high pass filter that can be used is of the following form, which ends up highlighting the centre pixel and works like a first-order derivative filter.

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

Laplacian

The Laplacian filter is to sharpening images as the Gaussian 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,

$$G_x = k_x \otimes I$$

$$G_y = k_y \otimes I$$

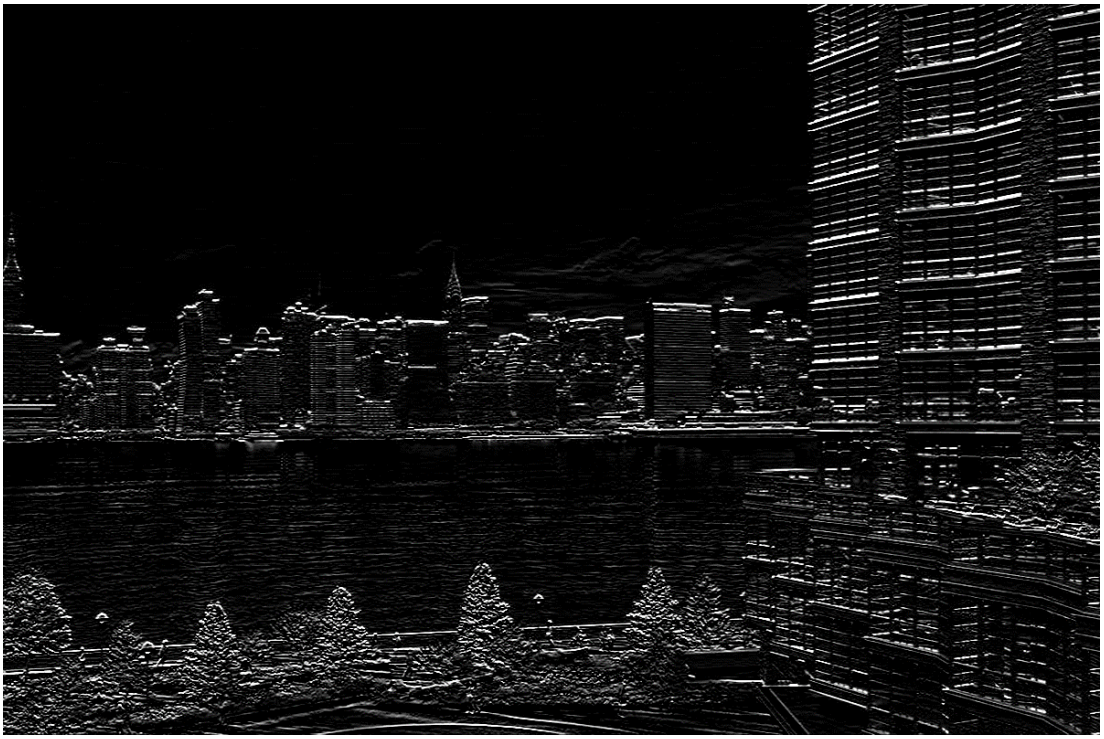
Combining the two images yields a magnitude and gradient image,

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

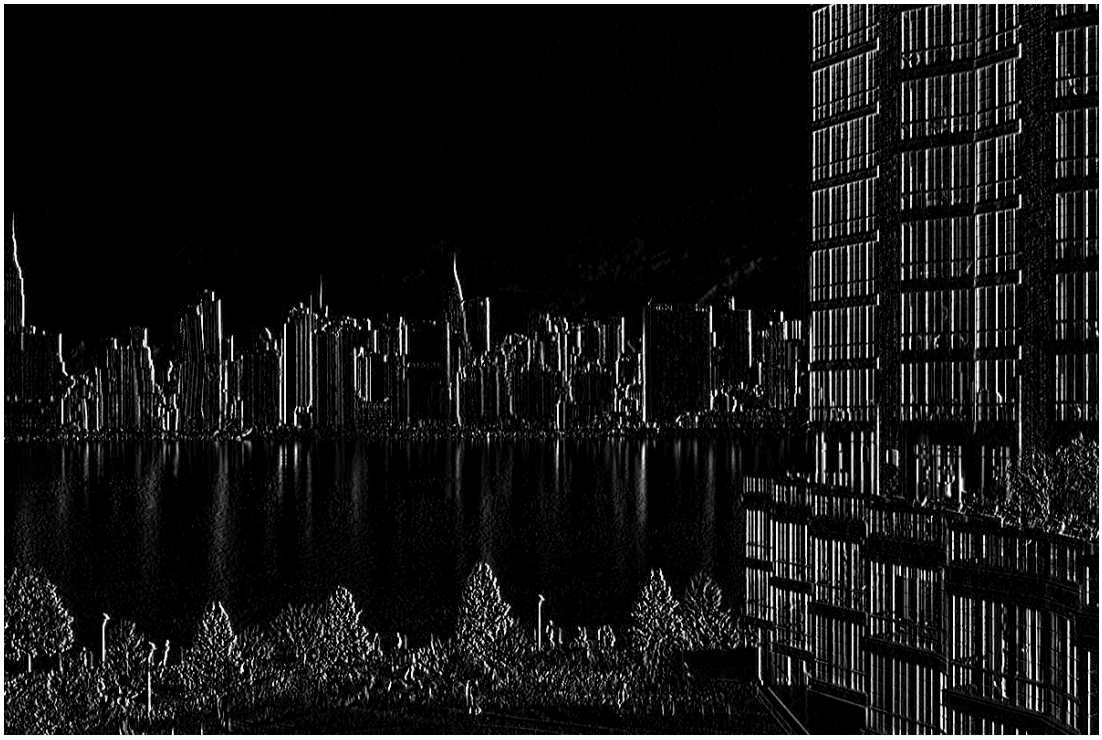
$$\Theta = \arctan(G_y \,/\, G_x)$$

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

Solutions for teaching and research. Made in Canada.