ENGR 13300
Python Project Resource – Feature detection

Color to Greyscale Images:

In this part of the project, we'll be using grayscale images because they are easier to process. These are also referred to as luminance, Y component, or Y images. Gray scale images typically have 8 bits/pixel, which allows for 256 shades of gray to be represented.

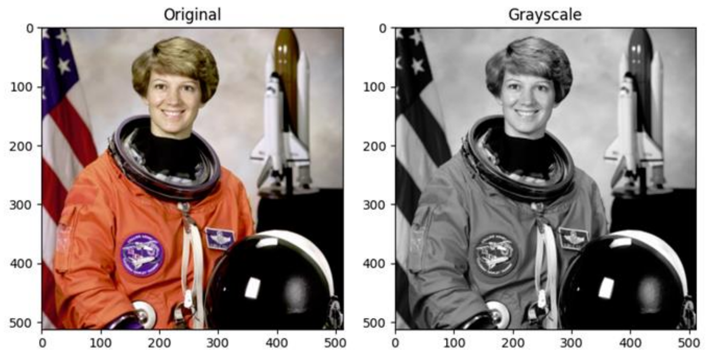

Figure 1. Grayscale Conversion Example of 500x500 pixel image (Source: Image Processing and Analysis Tutorial – Part II, Edward J. Delp, 8/29/2020, slide 11)

It is important to note that not all color scales contribute equally to an image's luminance or brightness when converting RGB images to grayscale. Green contributes the most to the brightness, where blue contributes the least. Two international standards have been developed for converting color images to grayscale images:

- ITU-R Recommendation BT.709: `Y = 0.2126R + 0.7152G + 0.0722B`
- ITU-R Recommendation BT.601: `Y = 0.299R + 0.587G + 0.114B`

In the above equations, Y is the output matrix image, R is the red matrix portion of the image, G is the green matrix portion of the image, and B is the blue matrix portion of the image. Research these standards to determine which method is most appropriate for your project and why.

Noise Smoothing:

Image smoothing is completed on images to reduce the noise in the images, which is important to do before the edge enhancement operation. A noisy image can negatively affect the quality of the edge detection algorithm.

The smoothing operation is a window operation. To implement a window operation, you extract a small NxN region from the image, perform the operation on the pixels in the window, and then replace the value of the center pixel of the NxN region with the result. Thus, for the example in Figure 2, a 3x3 region, or Window, is selected centered around the pixel located in row 3, column 3.

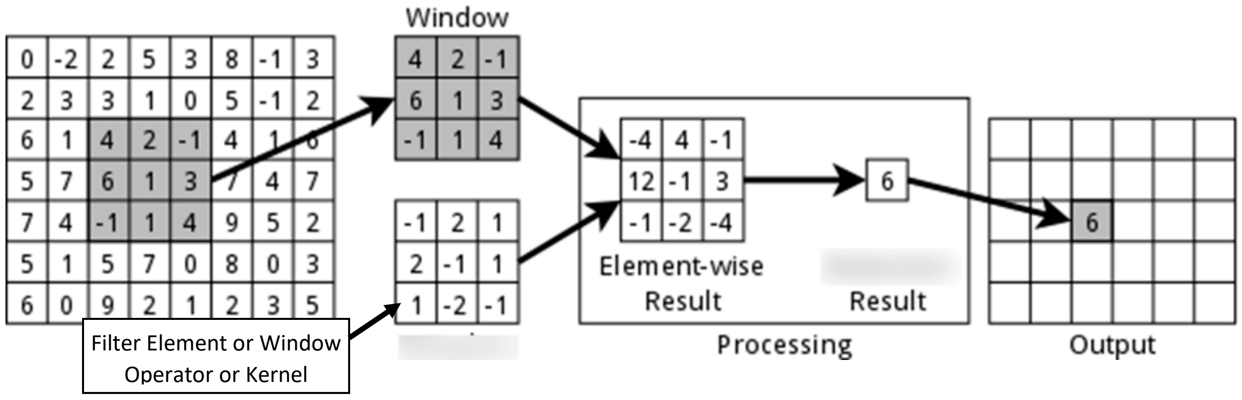

Figure 2. 3x3 Window Operation Example (Source: Image Processing and Analysis Tutorial – Part II, Edward J. Delp, 8/29/2020, slide 22)

For this example, each pixel from the Window is multiplied with the corresponding value of the window operator (also known as filter element or kernel), as shown in the "Element-wise Result". The values of the element-wise result are then summed, and the pixel of the output image is replaced with that value.

The smoothing (or blurring) operation is done by averaging or weighted filters. An example of an averaging filter using a 3x3 kernel is, where each pixel of the output is computed as the average of the surrounding pixels:

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}$$

Note that all weights must sum to one for the smoothing filters, hence the fraction term at the front of the kernel. This is so the kernel sum of the filter values is multiplied by the inverse of the sum. For the Gaussian filter, more weight is given to the central pixels. Below is a 5x5 Gaussian kernel, with Sigma = 1.056.

$$\frac{1}{256}\cdot\begin{bmatrix}1 & 4 & 6 & 4 & 1\\4 & 16 & 24 & 16 & 4\\6 & 24 & 36 & 24 & 6\\4 & 16 & 24 & 16 & 4\\1 & 4 & 6 & 4 & 1\end{bmatrix}$$

Here is an example of the image of Grace Hopper, an American computer scientist and United States Navy rear admiral, which has been smoothed with a simple averaging filter:

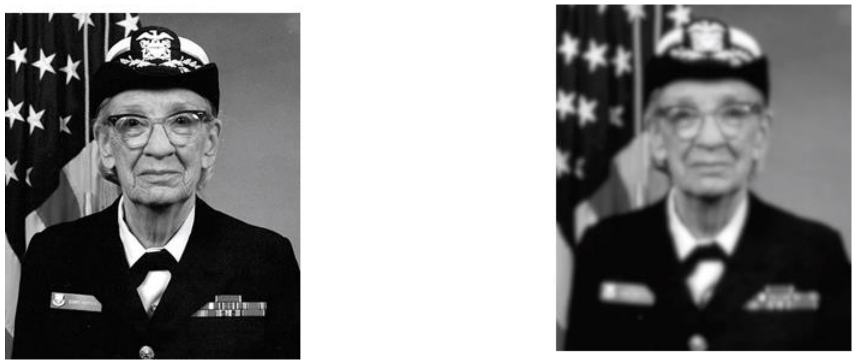

Figure 3. Image Smoothing Example of Grace Hopper (Source: Image Processing and Analysis Tutorial – Part II, Edward J. Delp, 8/29/2020, slide 23)

Gradient operator:

An edge in an image is an abrupt change in the intensity (pixel) values, or intensity gradient, that is perceived as a boundary or change. The edges can be located by taking the derivative of the intensity values across the image, then finding the points where it is a maximum. For example, if we were to plot the pixel values of the image on the left of Figure 4 in the direction of the red line, we would get the plot in the center. Then, if we take a derivative of the center plot, we get to plot on the right.

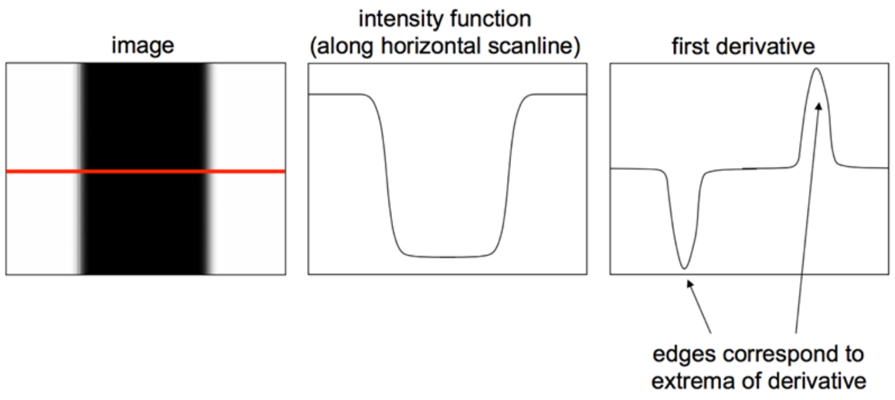

Figure 4. Edges in Images (Source: Image Processing and Analysis Tutorial – Part II, Edward J. Delp, 8/29/2020, slide 26, also http://stanford.edu/)

The Sobel operator is a discrete differentiation operator that can be used to locate the edges of an image by finding the maximums of the gradients in both the x and y directions separately, then computing the magnitude of the vector. Thus, for an image $f(x, y)$, we want to find the partial derivatives, $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$, where the magnitude of the gradient is

$$G = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial xy}\right)^2}$$

This is done by performing window operations using the following kernels:

X – Direction

-1	0	1
-2	0	2
-1	0	1

Y – Direction

-1	-2	-1
0	0	0
1	2	1

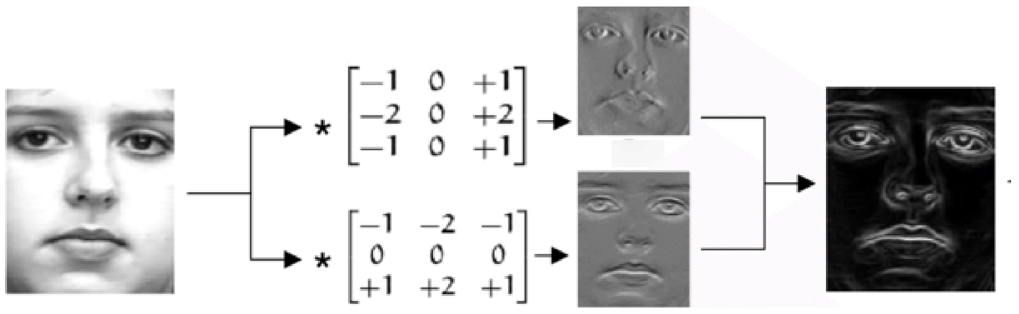
$* \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$
$* \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$

Figure 5. Sobel Edge Enhancement (Source: Image Processing and Analysis Tutorial – Part II, Edward J. Delp, 8/29/2020, slide 33)

It is important to be cognizant of the data type of the images, particularly when doing the derivatives. If the data type is “uint8” (unsigned integer), negative values will not be properly stored, and thus the results will not be correct.