

Photograph Manipulation Through Kernel Convolutions

There are many reasons why an individual would want to digitally edit a photograph. Two of such reasons include the simple editing for use by an everyday individual or, more in tune with my interest, editing in computer science for machine learning and identification.

Kernels are generally a three by three matrix that have different values based off of the process that is wanted. The process involves passing the kernel matrix over the photo matrix pixel by pixel to create a product image. The table below highlights some of the basic kernel convolutions. While these examples are very simple, there are kernels that are perform many types of operations.




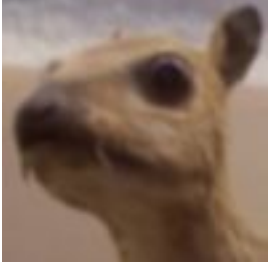
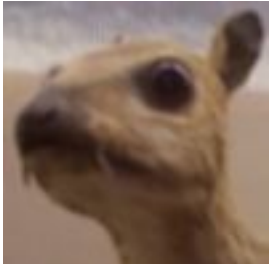

<p>Identity:</p> $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$		<p>Gaussian Blur 3x3:</p> $\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
<p>Sharpen:</p> $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$		<p>Gaussian Blur 5x5:</p> $\frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	
<p>Box Blur</p> $\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$		<p>Unsharp masking 5x5:</p> $\frac{-1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

Figure 1

Box blur is the most basic of kernel convolutions. It simply takes the average value for the given pixel and its eight neighbors. Gaussian blur is a more common blur that gives weights to the surrounding eight neighbors of a pixel based off of the distance from the center pixel. An interesting fact to point out about the above kernels is the sum of all the elements of each kernel matrix with the associated scalar is one in all the above cases. This is to preserve some of the properties of the image, such as average pixel value and intensity. Figure 2 demonstrates the process of applying a kernel on an image with a simple translation kernel.

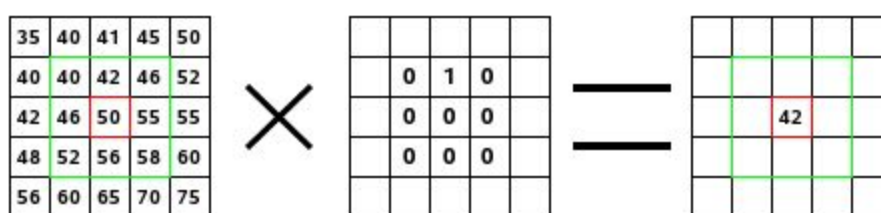


Figure 2

Image Source: <https://docs.gimp.org/en/images/filters/examples/convolution-calculate.png>

The strength of these operators come from their ability to be chained to create several effects. Photo editing apps such as Snapchat and Instagram use several of these kernels to create the popular effects like the vintage photograph look and the highly saturated filter Claredon. A common practice when running a photo through a series of convolutions is to first run a Gaussian Blur over the image to remove some of the granularity that is produced by the camera.

One edge case that must be considered when implementing a program that handles kernel convolutions is how to treat pixels that are found along the border of the picture. This is because the convolution needs data from pixels that do not exist. There are three main ways to handle these edge cases. These are to extend, wrap, and crop. The resulting image of a crop case is smaller than the original. Any convolution that would require non-existent data is simply

ignored and the source pixel is absent in the resulting image. Wrap on the other hand takes any non-existent data from the opposite edge. For example if a convolution is taking place on the top of an image and needs data above the photo, it simply uses data from the bottom. Lastly, extend uses the closest pixel that exist in the source image. All three methods have their benefits and downsides that need to be examined in a case by case basis. As shown above, these kernels perform a number of tasks that are used by everyday photograph editors, however, the power of kernel convolutions are not limited to the applications of photo hobbyists. My interest in kernel convolutions arises from some of the stronger kernels that will be highlighted below.

Some of the more popular kernel convolutions are the kernels involved in edge detection. Edge detection kernels are used throughout deep learning software in order to identify objects in images. Deep Convolutional Neural Networks are popular neural networks that use kernel convolutions to convert images into easier to handle data. If we compare some of the earlier images to the images below, it demonstrates how the data is trimmed and the edges are more cleanly defined. These images are produced by first translating the image to a grayscale then applying the associated kernel.




$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$
		

Figure 3

Image Source: Michael Plotke hosted on wikipedia found at [en.wikipedia.org/wiki/Kernel_\(image_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing))

Sobel edge detection continues to build on previous kernel work. Sobel edge detection takes advantage of two kernels. Of the two kernels, one measures the edges in the x direction and the other measures the edges in the y direction. The two directions are combined using the Pythagorean Theorem to create an image highlighting the edges similar to before, however; the strength of the Sobel method is from the separation of the two directions. Taking the inverse tangent of the y direction over the x direction, you are left with a result that gives the angle of orientation for a specific spot in relation to the rest of the image. This is illustrated in Figure 4. The angles in the 4th image are converted into color to give a visual representation.

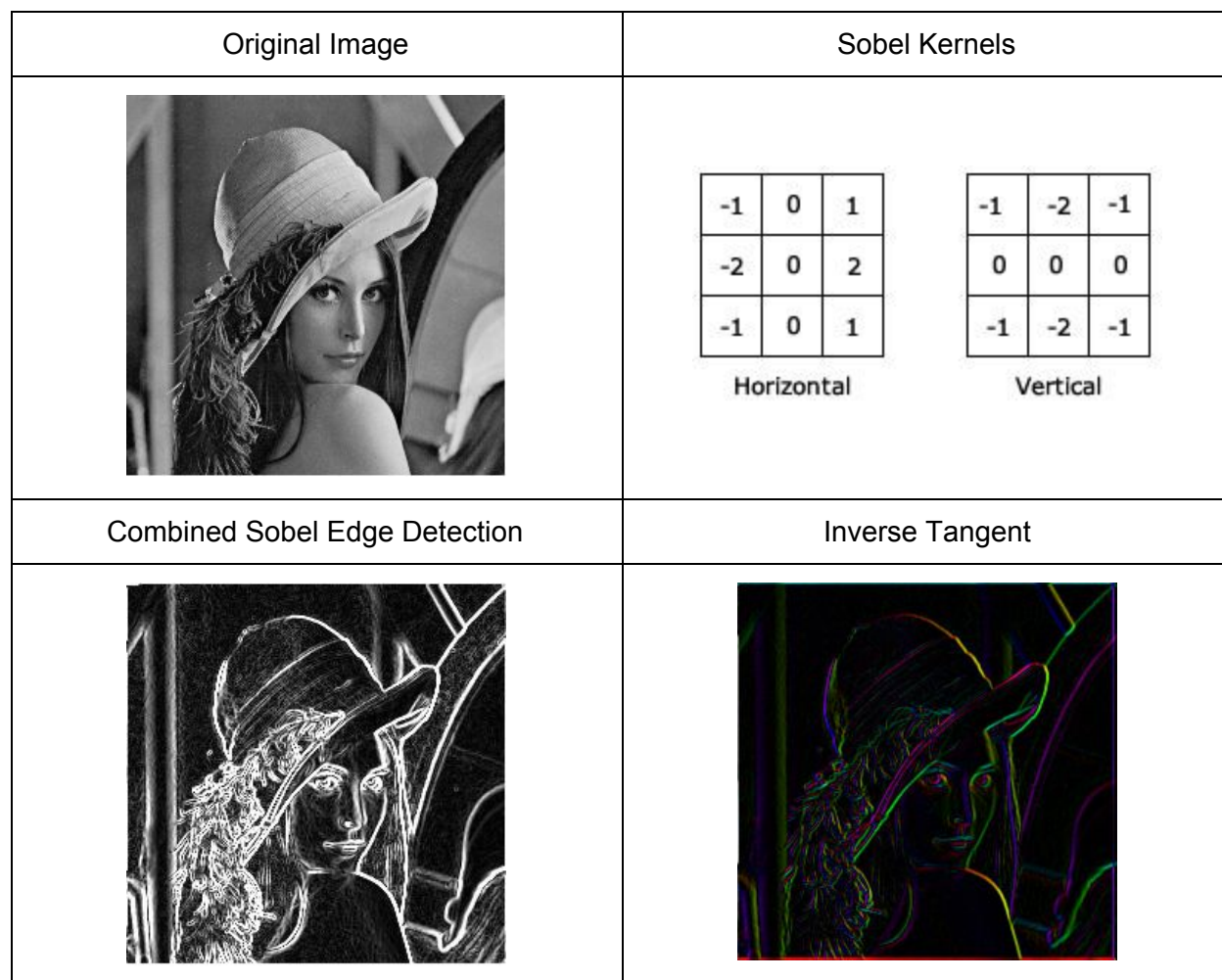


Figure 4

Image Source: iplt.org/_images/sobel.png, wikis.ece.iastate.edu/cpre584/images/b/bd/Sobel_edge_detection.jpg,
<http://aishack.in/static/img/tut/sobel-kernels1.jpg>

In summary, kernels are matrices that are applied to pictures to create a new photo. Different kernels have different effects and uses. Kernels can be chained together to extract data from images. Kernels are the backbone of certain computer neural networks and are helpful in identifying objects in images.

Sources

Images in order of appearance:

1. Done by Michael Plotke hosted at [en.wikipedia.org/wiki/Kernel_\(image_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing))
2. <https://docs.gimp.org/en/images/filters/examples/convolution-calculate.png>
3. iplt.org/_images/sobel.png
4. <http://aishack.in/static/img/tut/sobel-kernels1.jpg>
5. wikis.ece.iastate.edu/cpre584/images/b/bd/Sobel_edge_detection.jpg

Video References:

1. How Blurs & Filters Work ~ https://www.youtube.com/watch?v=C_zFhWdM4ic
2. Finding the Edges ~ <https://www.youtube.com/watch?v=uihBwtPIBxM>
3. Edge Detector ~ <https://www.youtube.com/watch?v=sRFM5IEqR2w>

C# Kernel Exploration Software:

<http://bit.ly/computerphileEdge>

Scholarly References:

Lecarme, Olivier, and Karine Delvare. *The Book of GIMP: A Complete Guide to Nearly Everything*. N.p.: No Starch Press, 2013. Print.

Shapiro, Linda G., and George C. Stockman. *Computer Vision*. Upper Saddle River, NJ: Prentice-Hall, 2001. Print.