Linear Algebra: Applications of Linear Algebra in Image Processing & Cryptography

Due on September 22, 2019

Professor MacArthur

Carson Storm

Image Processing

Description of topic

Digital Image Processing encompasses the transformation of digital images in order to produce an image that is more suitable for analysis or viewing (McConnnell, 2003). The two main motivations for image processing are to change the appearance of the image and to extract information from the image, both rely heavily on concepts from linear algebra. Images are very often represented by matrices whose entries correspond to a numerical representation of the color of the pixel at the same location as the entry. For example, if the top left most pixel is black, if entry in the first column in the first row would be the number representing black. Image processing can be defined simply as performing an operation on the data contained in the image (Gonzale & Woods, 2018).

Most transformations that are used for the purpose of extracting information alter the image or remove extraneous information until the image is in a state that is more easily analyzed. For example, an image can be enhanced by suppressing the details of the image that are not important to the current task (McConnnell, 2003). While altering an image is rarely enough to put an image into a state in which it is easily analyzed, it is often a step towards the desired result. These transformations usually take the form of affine transformation, such as scaling, reflection, translation, rotation or shearing (Gonzale & Woods, 2018). For example, a image may need to be flipped horizontally in order to have the right perspective before further filtering. The removal of extraneous information is often achieved through filtering. Filtering can take two main forms, spatial filter, and frequency filtering. This paper will focus on Spatial filtering, as it is more relevant to Linear Algebra. Spatial filter is filtering that defines a new image in terms of some operation that operates on each pixel in the image (Gonzale & Woods, 2018). Some examples of spatial filtering are noise reduction, increasing contrast or thresholding, were if entries are not in some range they are set to some constant (Gonzale & Woods, 2018). Spatial filter is performed by computing the discrete convolution of the image and some kernel that is specific to the type of filtering (Gonzale & Woods, 2018). Convolution in the context of image processing is related to, but

different than the definition of mathematical convolution, convolution is defined as

$$g(x,y) = \omega * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \omega(s,t) f(x-s, y-t)$$

where $g(x,y)$ is the filter image, $f(x,y)$ is the original image, ω is the kernel, such that every entry is (s,t) is of the form $-a \leq s \leq a$ and $-b \leq t \leq b$ (Lecarme & Delvare, 2013).

Image processing also often used simply to alter a image for display purposes. For example, an application might want to flip a image horizontally, so that it has the perspective we are used to, or it might want to convert the image to grayscale. This type of image processing utilizes the same method described earlier.

Motivation for choice of topic

I chose the topic of image processing to discuss because I have experience implementing image processing algorithms, but before taking this class and researching this topic I had no idea how any of the libraries I was using and in most cases how each part of my algorithms functioned. I relied mostly on the documentation for the libraries and copying and pasting from the internet to get the algorithms to function. I was unsure why I was using certain method calls or why I had to transpose my matrix before passing it to a certain method.

One such time I utilized image processing in the past was for my robotics team. I needed to identify certain objects on the playing field and perform tasks depending on certain tasks depending on whether an object was in view of the camera and where it was. I had toyed with the idea of using machine learning to determine if the object was in view, but the time it would take to train a model and to actually use the neural network was too long. I instead implemented an algorithm that detected the object using computer vision, an application of image processing, I used various types of filtering to look for the color and shape that I knew the object would be to determine whether the object was in view. The process of implementing this algorithm was fairly difficult because I had no idea what I needed to do to get the information I needed from the camera, so I wanted to research this topic so that in the future if I need to implement another algorithm like this I will have a better understanding of what I need to do.

Example of topic

All of the following examples will be demonstrated by operating on Figure 1

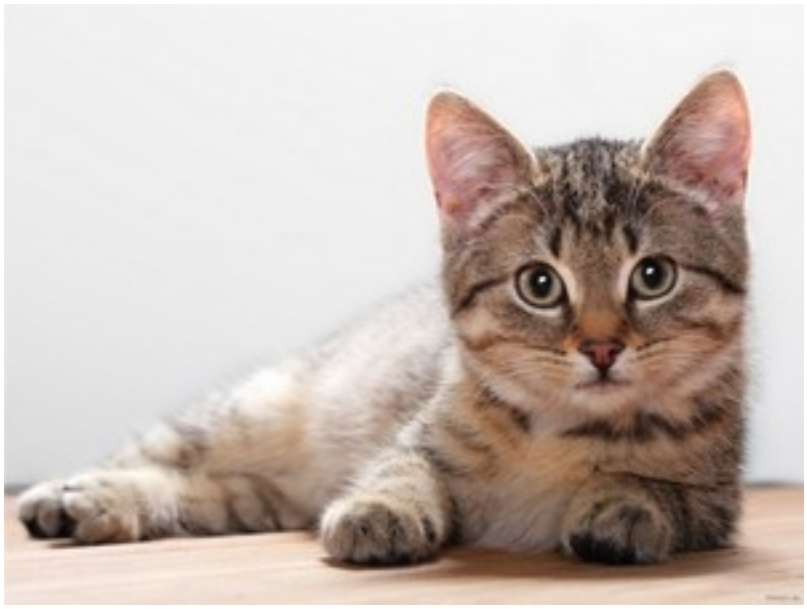

Figure 1: Original Image

Reflection

In order to reflect an image a simple affine transformation is necessary such that every pixel in the resulting image is defined by

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

where (x, y) are the pixel coordinates in the original image and (x', y') are the pixel coordinates in the transformed image. If this process performed on Figure 1 would result in Figure 2.

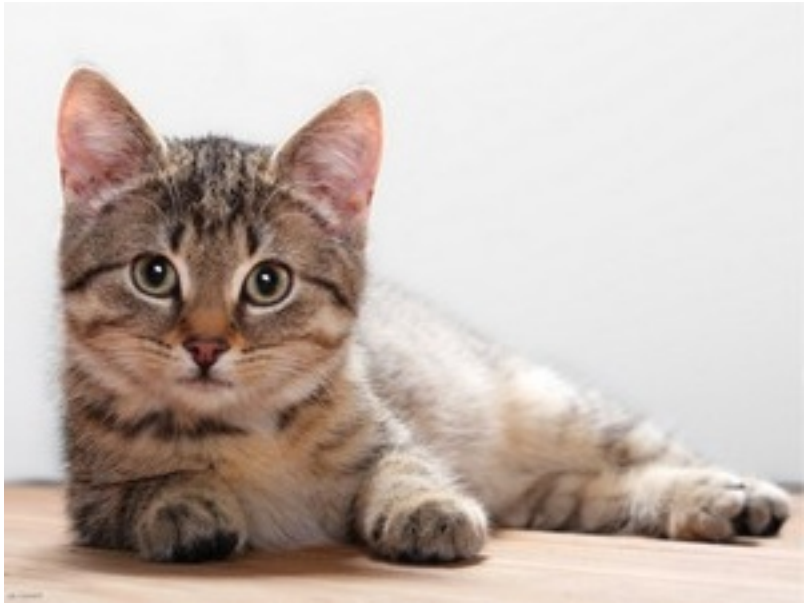

Figure 2: Reflected image

Grayscale

To convert an image to grayscale a simple operation is performed on each pixel such that each pixel in the resulting image is defined by

$$c = 0.3R + 0.59G + 0.11B$$

where c is value of the pixel in the transformed image and R, G, B are the red, green and blue values of the pixel in the original image. If this process performed on Figure 1 would result in Figure 3.

Figure 3: Grayscale image

Blur

To blur an image, the convolution of the matrix and a Gaussian blur kernel can be computed, so that each pixel in the resulting image is defined by

$$g(x,y) = \omega * f(x,y) = \sum_{s=-1}^{1} \sum_{t=-1}^{1} \omega(s,t) f(x-s, y-t), \omega = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

where $g(x,y)$ is the filter image, $f(x,y)$ is the original image, ω is the Gaussian blur kernel. If this process performed on Figure 1 would result in Figure 4.

Figure 4: Blurred image

Laplacian

The laplacian can be used to detect the edges in an image. To perform the laplacian on the an image, each pixel in the resulting image is defined by

$$g(x,y) = \omega * f(x,y) = \sum_{s=-1}^{1} \sum_{t=-1}^{1} \omega(s,t) f(x-s, y-t), \omega = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

where $g(x,y)$ is the filter image, $f(x,y)$ is the original image, ω is the laplacian kernel. If this process performed on Figure 1 would result in Figure 5.

Figure 5: Edge detected image

Cryptography

Description of topic

Motivation for choice of topic

Example of topic

The following piece of text "I love MATH2270" can be encrypted using a matrix by first converting the characters of the text to numbers (in this case we will use there ascii codes).

I		l	o	v	e		M	a	t	h	2	2	7	0
73	32	108	111	118	101	32	77	97	116	104	50	50	55	48

If the following matrix with the following inverse is used as the cipher

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 5 \end{bmatrix}, A^{-1} = \begin{bmatrix} 1 & -2 & \frac{1}{5} \\ 0 & 1 & \frac{-2}{5} \\ 0 & 0 & \frac{1}{5} \end{bmatrix}$$

Then the message can be represented by the following matrix

$$\begin{bmatrix} 73 & 32 & 108 & 111 & 118 \\ 101 & 32 & 77 & 97 & 116 \\ 104 & 50 & 50 & 55 & 48 \end{bmatrix}$$

The product of these two matrices gives use the encrypted message

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 5 \end{bmatrix} \begin{bmatrix} 73 & 32 & 108 & 111 & 118 \\ 101 & 32 & 77 & 97 & 116 \\ 104 & 50 & 50 & 55 & 48 \end{bmatrix} = \begin{bmatrix} 587 & 246 & 412 & 470 & 494 \\ 309 & 132 & 177 & 207 & 212 \\ 520 & 250 & 250 & 275 & 240 \end{bmatrix}$$

The encrypted message is then

$$587, 246, 412, 470, 494, 309, 132, 177, 207, 212, 520, 250, 250, 275, 240$$

To decrypt the message, the matrix is just multiplied by the inverse of the cipher

$$\begin{bmatrix} 1 & -2 & \frac{1}{5} \\ 0 & 1 & \frac{-2}{5} \\ 0 & 0 & \frac{1}{5} \end{bmatrix} \begin{bmatrix} 587 & 246 & 412 & 470 & 494 \\ 309 & 132 & 177 & 207 & 212 \\ 520 & 250 & 250 & 275 & 240 \end{bmatrix} = \begin{bmatrix} 73 & 32 & 108 & 111 & 118 \\ 101 & 32 & 77 & 97 & 116 \\ 104 & 50 & 50 & 55 & 48 \end{bmatrix}$$

This matrix can then be converted back to text to get "I love Math2270"

73	32	108	111	118	101	32	77	97	116	104	50	50	55	48
I		l	o	v	e		M	a	t	h	2	2	7	0

References

Gonzale, R. C., & Woods, R. E. (2018). *Digitial image processing* (4th ed.). New York: Pearson.

Lecarme, O., & Delvare, K. (2013). *The book of gimp: A complete guid to nearly everything.* No Startch Press.

McConnnell, J. J. (2003). *Computer graphics companion.* Wiley.