A Brief Explanation of The Images and Videos

Quin Darcy

Feb, 15 2021

We start with an image which can be thought of as a grid, where in each cell is a pixel identified by an RGB tuple. This tuple determines the color of the pixel. The tuples are in the form (R, G, B), where $R, G, B \in \mathbb{Z}_{255}$. The program scans through (from left to right and top to bottom) and at each pixel, it stores a 3x3 window whose top left cell is the current pixel being scanned, call it $P_{i,j}$.

The second element of this program is called a *kernel*. This kernel, call it K, is a 3x3 matrix whose entries are real numbers. In this case our kernel is

$$K = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

From this point forward, we deviate quite a bit from what is typically done with the window and kernel. These objects have many uses, one of which is edge detection and to understand this more, refer to image convolution.

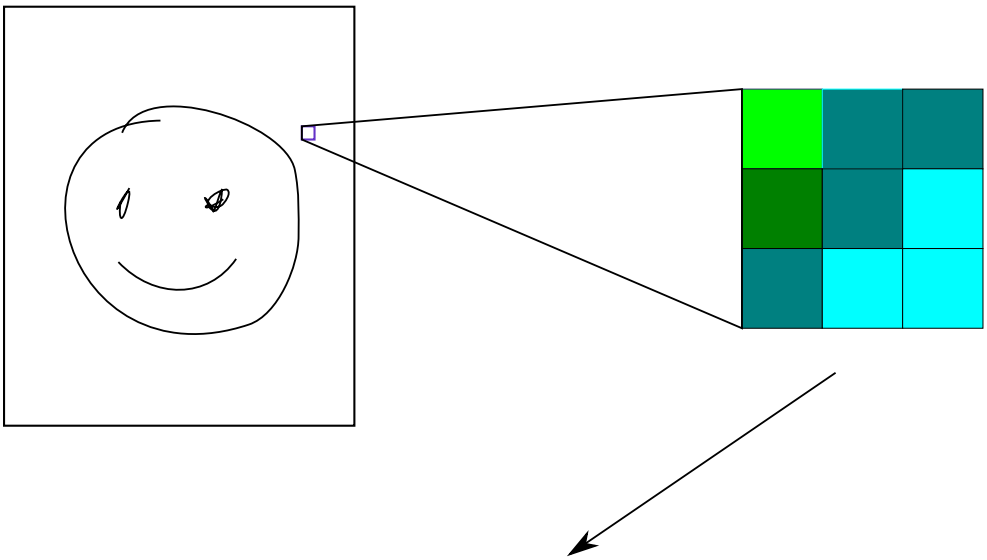
Instead we are going to first take every tuple in the window and make 3 matrices. A matrix, R, whose entries are the red components of the tuples in the window, another matrix, G, whose entries are the green components, and a third matrix, B, whose entries are the blue components. With these matrices we will calculate the following:

$$R' = \det(RK - KR), \quad G' = \det(GK - KG), \quad B' = \det(BK - KB).$$

After this we are left with a new tuple (R', G', B'). Note that we if any of the components are greater than 254 or less than 0, we replace it with its equivalence class modulo 255. With this new tuple, the last thing we do is replace the center of the window with this tuple by setting $P_{i+1,j+1} = (R', G', B')$. This completes the process and the program then goes on the the next pixel $P_{i+1,j}$ and repeats the process. This is done until the entire image has been scanned.

It is quite apparent that defining R', G', B' in the way that we did was arbitrary and not necessarily meaningful. However, having just learned about the Lie bracket (abuse of terminology since we are not in a Lie algebra), it was interesting to see its effects on an image when applied this way.

Below is a diagram of the process.



$$(0,255,0) \quad (0,128,128) \quad (0,128,128)$$

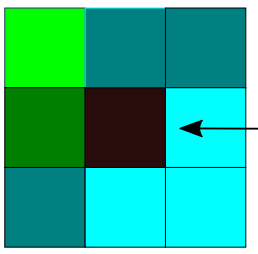
$$(0,128,0) \quad (0,128,128) \quad (0,255,255)$$

$$(0,128,128) \quad (0,255,255) \quad (0,255,255)$$

$$R = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad G = \begin{pmatrix} 255 & 128 & 128 \\ 128 & 128 & 255 \\ 128 & 255 & 255 \end{pmatrix} \quad B = \begin{pmatrix} 0 & 128 & 128 \\ 0 & 128 & 255 \\ 128 & 255 & 255 \end{pmatrix}$$

$\det(RK - KR) = 0$ $\det(GK - KG) = 0$ $\det(BK - KB) = 29114880$

$\mod(255)$ $\mod(255)$ $\mod(255)$



$(0,0,0)$

Figure 1: Program

Another thing worth noting is that in the RGB color space, if you take a point (R, G, B), then its "negative" is given by $(255 - R, 255 - G, 255 - B)$. This effect is used in a few of the images. There are a handful of other techniques used in the images, like converting from the RGB color space to the HSV (Hue, Saturation, Value) color space and then running the program. Anyways, please enjoy!