**Basic Photo Processing: The Math Behind Blurring an Image and Its Significance**

The problem given to CS110 students at WVU was to analyze a 512x512 pixel photo and process the image in some way, shape, or form that could be considered useful. For my project, I decided to write a method that blurs an image, utilizing a Gaussian Blur. The significance of blurring an image is that it can reduce noise within the image for further image processing, acting as a preliminary filter. Such filters are quite often used in computer vision, specifically in facial recognition algorithms as an excessive amount of noise prevents the algorithm from properly functioning. So how does the Gaussian Blur work? Well the concept of a Gaussian Blur, in lamest terms, is taking the average of the RGB color values of a pixel and the surrounding pixels (the ones directly adjacent to it). Then after getting the three different averages (one for red, blue, and green), the pixels are all set to these values. This process is repeated throughout the image, resulting in the desired blur. So, in terms of achieving this through code, the photo would need to be loaded in as a two-dimensional array of pixels, with each element/pixel containing a set of RGB values. This array would then need to be iterated through, using two for loops. At each element, two more for loops would need to be run to be able to get the values of the surrounding pixels for the designated element. This would then have to enter an if-statement, that would check if the surrounding pixels were within the bounds of the picture array. If they were, then they could be used in the average calculation. If not, the for loops would just continue to the next adjacent pixel. To blur certain sections of the image, while leaving the others intact, the code above would need to be put in an if-statement that has the condition you wish to use. In the code provided in the GitHub Repository, the condition is simply if the pixel is between 250 and 350, both in row and column, the pixel does not get blurred. This condition could be further advanced upon, like sorting based on a specific color. Take a look for yourself in the provided code!