Armaan Bandali

301322810

January 17, 2020

ENSC 474

Assignment 1

The purpose of this assignment is to familiarize ourselves with MATLAB's basic image processing functions and manipulate the colour arrays composing images in order to achieve desired results.

The 2D sinc function in Question 1 can be expressed in general form as seen in Equation 1. The 2D sinc function displays attenuating oscillatory behaviour radially about its centre and thus must be observed near its centre to witness this behaviour. Requiring that the x-axis range from 100 to 355 and the y-axis from 0 to 255, applying shifts to Equation 1 produces Equation 2, which is centred at our new coordinate viewspace.

$$f(x,y) = \frac{\sin\left(\sqrt{x^2+y^2}\right)}{\sqrt{x^2+y^2}}$$

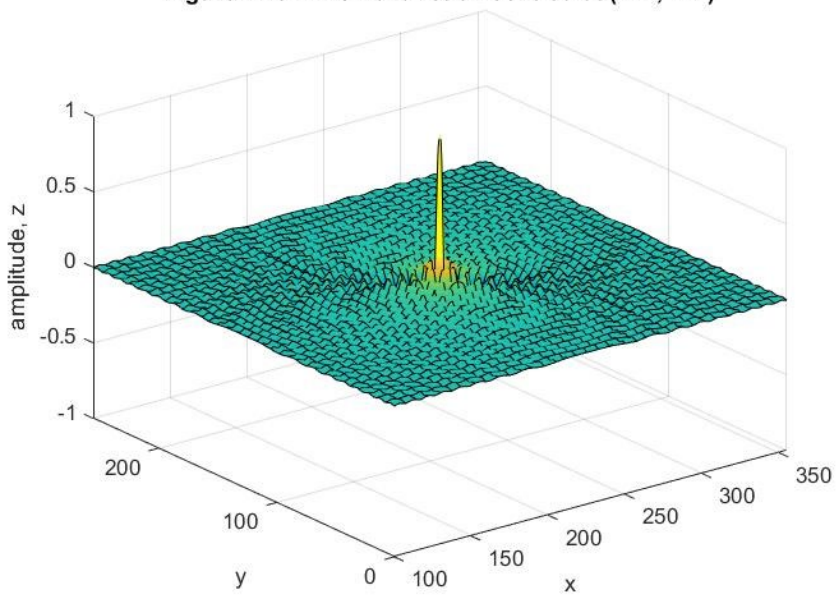
Equation 1. General form of a 2D sinc function centred at the origin

$$f(x,y) = \frac{\sin\left(\sqrt{(x-227)^2+(y-127)^2}\right)}{\sqrt{(x-227)^2+(y-127)^2}}$$

Equation 2. Shifted 2D sinc function

Plotting this function produces the surface plot seen in Figure 1-1a. In line with our prediction, the plot illustrates the sinc function with a maximum amplitude of 1 at its centre and oscillating radially outward while decreasing as x and y increase.

Figure 1-1a. 2D sinc function centred at (227, 127)



Question 2 introduces the basics of image processing by reading in an image and separating it into its separate colour channels. The imread() function of MATLAB takes an image and produces the three dimensional RGB colour matrix for each pixel. A z value of 1 gives the red channel, 2 gives green, and 3 gives blue. imshow() is used to display an image on the screen by reading a colour matrix. To produce the images seen in Figure 1-2a, 1-2b, and 1-2c, 3D colour arrays need to be read so my algorithm accounts for this by adding two planes of zeroes at the appropriate z-value. This process is equivalent to setting the image contribution of the two unused colour channels to zero.

Figure 1-2a. Red Channel

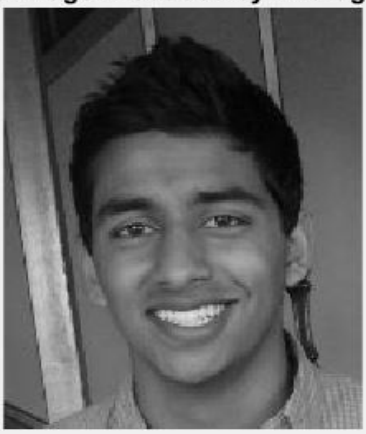
Figure 1-2b. Green Channel

Figure 1-2c. Blue Channel

Figure 1-2d. Original Image

The three colour channels can be averaged to produce the grayscale of the original image. The grayscale is simply a 2D array in this case produced by dividing the sum of the red, green, and blue vectors by 3. The product of this colour matrix when shown with imshow() can be seen in Figure 1-3.

Figure 1-3. Grayscale Image Produced by Averaging Colour Channels



The grayscale matrix is composed of light intensity values rather than the individual colour channels values of the original image. Using the surf() function instead of the imshow() function allows us to see the intensity of each pixel (Figure 1-4); for example, the teeth in the image are particularly bright and we expect to see high intensity values for those pixels, which we do. We can be especially confident in this analysis due to a different view taken perpendicular to the xy-plane which allows us to map intensity values visually back to the image.

Figure 1-4. Grayscale Surface Plot of Image Intensities

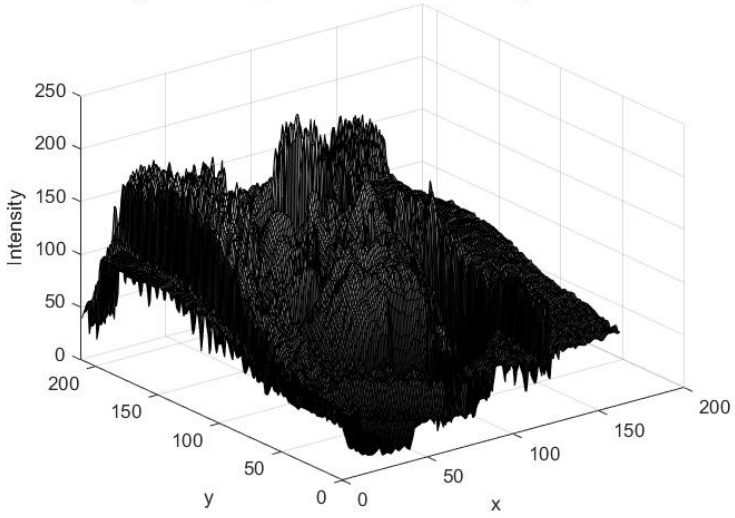
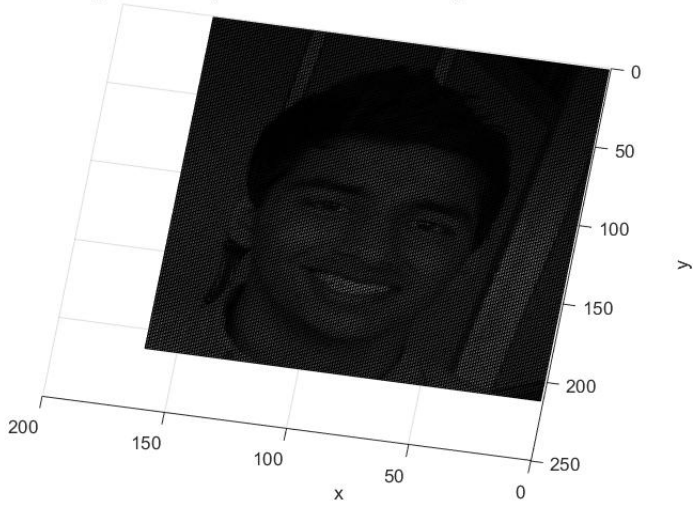
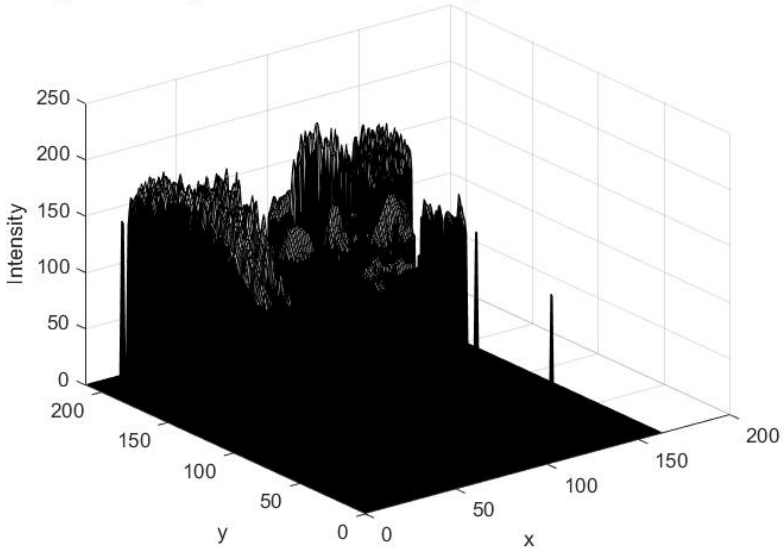


Figure 1-4. Grayscale Surface Plot of Image Intensities



The practical application of intensity values obtained from the grayscale array is to filter pixels based on intensity and only plot the desired pixels. In this case, my algorithm to filter out pixels with intensity value less than 129 involved scanning top-down from left to right and setting undesired pixels to an intensity of zero. The intensity plot of the filtered grayscale array can be seen in Figure 1-5. A large portion of the outer area of the image has been filtered out when compared to the original grayscale.

Figure 1-5. Grayscale Surface Plot of Image Intensities Greater than 128



These basic image processing techniques will allow us to further manipulate images through mathematical algorithms. An introduction to MATLAB's capabilities should provide a solid basis for learning more powerful techniques.