Assignment 4: Fourier Transform and Frequency Filtering

KEJVI CUPA
Department of Computer Science and Engineering
University of South Florida, Tampa, Florida, USA

1. Introduction and overall description

This assignment focuses on experimenting with Fourier Transform and applying Frequency Filtering methods in the frequency domain. In order to apply the Fourier Transformation, OpenCV was utilized. We needed to display the amplitude spectrum of the fourier transformation of an image that is passed as input. Furthermore, we experimented with low-pass and high-pass filtering in order to smooth images by reducing noise and edge enhancement. Additionally, unsharp masking was also applied on the low and high frequency components of an image. The images were grayscale, but these methods were also applied on Color images in HSV space, by applying the filtering in each component. Similar to previous assignments, the processing needed to be applied up to 3 user provided ROI in addition to their respective additional parameters. Overall, there will be eight new functions that will be implemented: DisplayAmplitude, InverseFourier, lowpass, highpass, unsharp_mask, low_pass_color, high_pass_color, unsharp_mask_low. A more detailed description of each function will be provided in this report.

2. Description of Algorithm

The first algorithm to be implemented is "DisplayAmplitude". This algorithm is used to convert an image to frequency space using Discrete Fourier Transformation via OpenCV. Once the image is converted to frequency space, further processing is needed in order to make the data representation which is currently with complex numbers visible. So the complex number is split into real and imaginary parts, and the magnitude is calculated. Then, quadrant swapping is performed in order to make the center of the image be the actual center and make it easier to display. Lastly, the values are normalized in the range 0-255, in order for a human to see the fourier spectrum. In this algorithm the user can specify up to 3 ROI, and a fourier amplitude spectrum will be generated for each.

The second algorithm to be implemented is "InverseFourier". This is a simple algorithm because it basically converts an image from frequency domain to image domain.

OpenCV is utilized for the application of Inverse Fourier Transformation. Then the image is converted to 8 bits and normalized in the range 0-255 for display. In this algorithm the user can specify up to 3 ROI, and an inverse fourier transformation will be generated for each.

The third algorithm to be implemented is “lowpass”. This algorithm builds on the previous algorithms. It still uses Discrete Fourier Transformation to convert the image to frequency domain, and then in that domain it performs low-pass filtering. We use a circular filter, and calculate the radius with the formula radius = (center_r^2 + center_c^2). The user provides a cutoff threshold to which the radius that is calculated for pixel is compared to. Now we create a mask for the filter. The mask has the same size as the image, and it has 0 and 1 values. If radius <= threshold, pixel value of mask is 1, else 0. Then the mask is used to multiply with the fourier image, and the resulting image is the filtered one. Then we apply Inverse Fourier Transformation to convert back to image domain for display of the results of the filtering. The image will appear blurry. This algorithm up to 3 ROI are specified which are user defined.

The fourth algorithm to be implemented is “highpass”. This algorithm builds on the previous algorithms. It still uses Discrete Fourier Transformation to convert the image to frequency domain, and then in that domain it performs low-pass filtering. We use a circular filter, and calculate the radius with the formula radius = (center_r^2 + center_c^2). The user provides a cutoff threshold to which the radius that is calculated for pixel is compared to. Now we create a mask for the filter. The mask has the same size as the image, and it has 0 and 1 values. If radius >= threshold, pixel value of mask is 1, else 0. Then the mask is used to multiply with the fourier image, and the resulting image is the filtered one. Then we apply Inverse Fourier Transformation to convert back to image domain for display of the results of the filtering. The image will appear with sharpened edges. This algorithm up to 3 ROI are specified which are user defined.

The fifth algorithm to be implemented is “edgesharp-unsharp_mask”. This algorithm builds on the previous algorithms. It still uses Discrete Fourier Transformation to convert the image to frequency domain. Since we are targeting high frequency pixels in that domain, we apply a high pass filter to it. Then we use the formula: *if high-freq mask = multiplier else mask =1.* We update the pixels of the original image accordingly, and display the results of the filtering. The image will appear sharpened. This algorithm up to 3 ROI are specified which are user defined.

The sixth algorithm to be implemented is “lowpass_color”. This algorithm builds on the previous algorithms. Now we are applying it to Color images of HSV color space. We select H, S or V channels by passing the values 0, 1, or 2 respectively as an additional parameter to the function. It still uses Discrete Fourier Transformation to convert the image to frequency domain, and then in that domain it performs low-pass filtering. We use a circular filter, and calculate the radius with the formula radius = (center_r^2 + center_c^2). The user provides a cutoff threshold to which the radius that is calculated for pixel is compared to. Now we create a mask for the filter. The mask has the same size as the image, and it has 0 and 1 values. If radius <= threshold, pixel value of mask is 1, else 0. Then the mask is used to multiply with the fourier image, and the resulting image is the filtered one. Then we apply Inverse Fourier Transformation to convert back to image domain for display of the results of the filtering. This algorithm up to 3 ROI are specified which are user defined.

The seventh algorithm to be implemented is "highpass_color". This algorithm builds on the previous algorithms. Now we are applying it to Color images of HSV color space. We select H, S or V channels by passing the values 0, 1, or 2 respectively as an additional parameter to the function. It still uses Discrete Fourier Transformation to convert the image to frequency domain, and then in that domain it performs low-pass filtering. We use a circular filter, and calculate the radius with the formula radius = (center_r^2 + center_c^2). The user provides a cutoff threshold to which the radius that is calculated for pixel is compared to. Now we create a mask for the filter. The mask has the same size as the image, and it has 0 and 1 values. If radius >= threshold, pixel value of mask is 1, else 0. Then the mask is used to multiply with the fourier image, and the resulting image is the filtered one. Then we apply Inverse Fourier Transformation to convert back to image domain for display of the results of the filtering. This algorithm up to 3 ROI are specified which are user defined.

The final algorithm to be implemented is "edgesharp-unsharp_mask_low". This algorithm builds on the previous algorithms. It still uses Discrete Fourier Transformation to convert the image to frequency domain. Since we are targeting low frequency pixels in that domain, we apply a low pass filter to it. Then we use the formula *if low-freq mask = multiplier else mask =1.* We update the pixels of the original image accordingly, and display the results of the filtering. The image will appear sharpened. This algorithm up to 3 ROI are specified which are user defined.

3. Results

Below are shown the results (images) with different combinations of the respective function parameters.

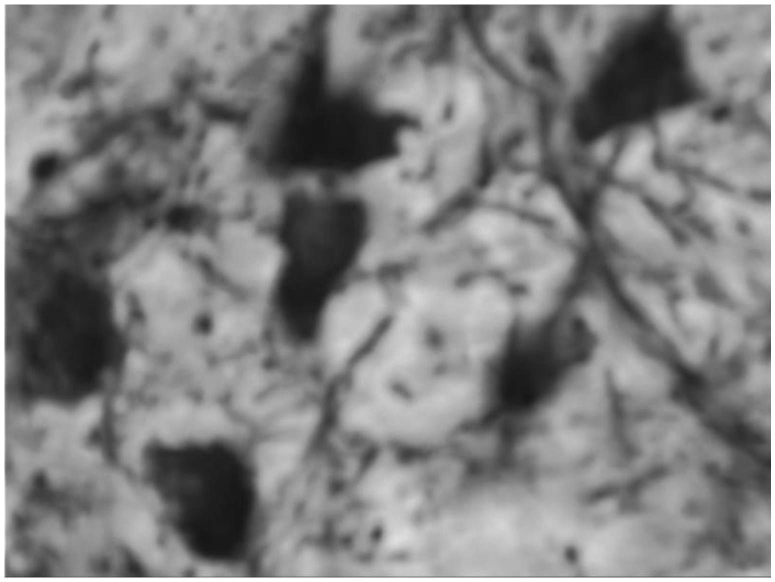


Figure 1: Original Image

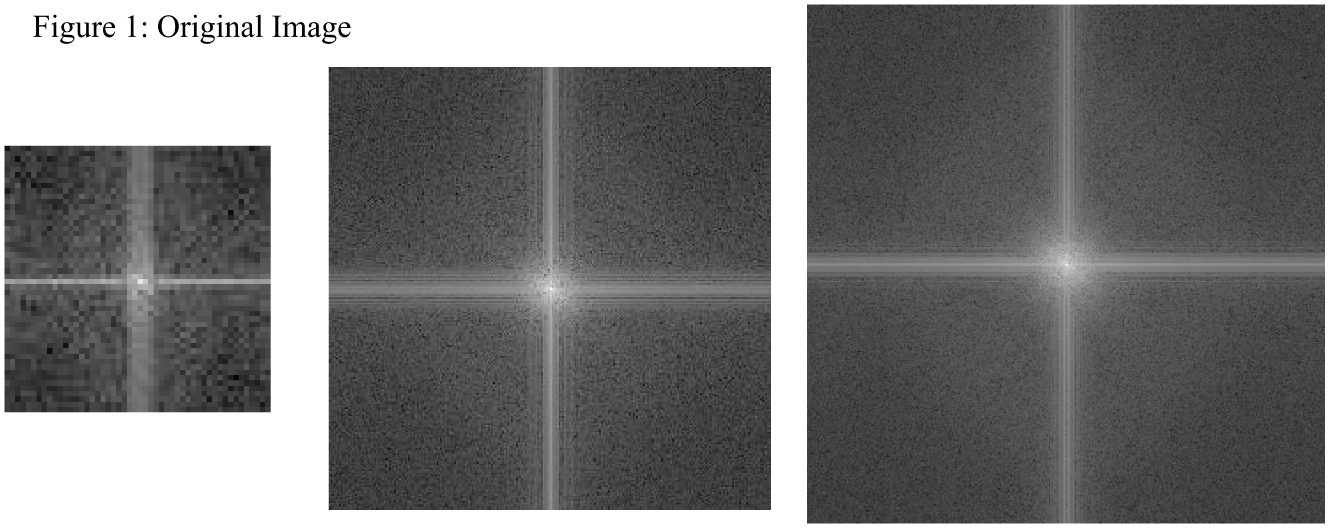


Figure 2: “DisplayAmplitude” function. Utilizing Display Amplitude function to convert image to frequency domain and make it visible to the human eye. ROI 1, 2, 3 in that order. The used parameter configuration is: #roi = 3, x1= 50, y1= 50, Sx1= 150, Sy1= 150, x2 = 250, y2 = 250, Sx2 = 100, Sy2 = 100, x3 = 400, y3 = 400, Sx3 = 150, Sy3 = 150.

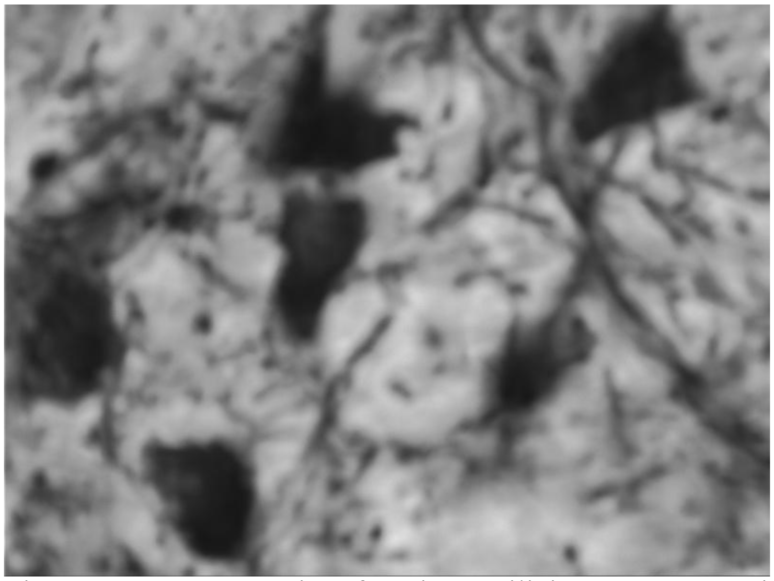


Figure 3: “InverseFourier” function. Utilizing Inverse Fourier Transformation to pass an image input, convert it to frequency domain and then convert back to image domain for display. The output image will be the same as the input image. In this case, the full image was passed as input rather than smaller ROIs to make the fourier representation better visually. The used parameter configuration is: #roi = 1, x1= 0, y1= 0, Sx1= 800, Sy1= 600 or #roi = 3, x1= 50, y1= 50, Sx1= 150, Sy1= 150, x2 = 250, y2 = 250, Sx2 = 100, Sy2 = 100, x3 = 400, y3 = 400, Sx3 = 150, Sy3 = 150.

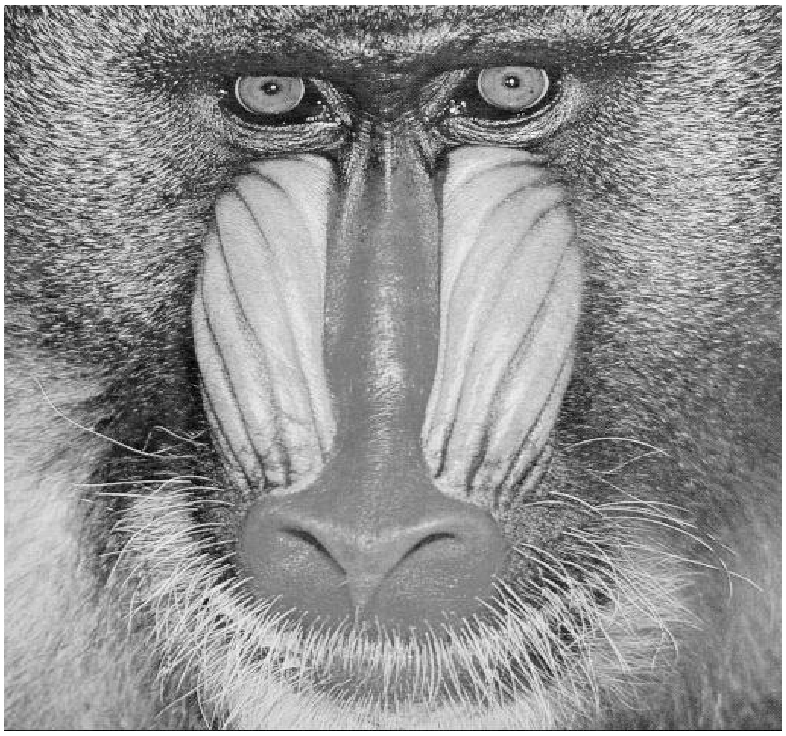


Figure 4: Original Image

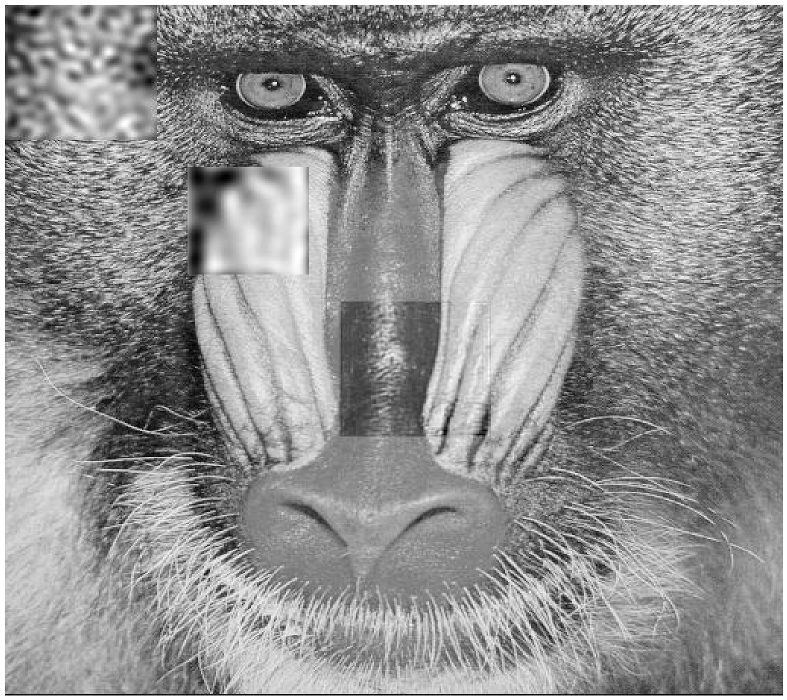


Figure 5: “lowpass” function. Applying low pass filtering to the image in the frequency domain. The resulting image in the ROIs that the filtering was applied on is blurry depending on the threshold value passed. The used parameter configuration is: The used parameter configuration is: #roi = 3, x1= 0, y1= 0, Sx1= 100, Sy1= 100, threshold1 = 10, x2 = 450, y2 = 100, Sx2 = 150, Sy2 = 150, threshold2 = 5, x3 = 280, y3 = 280, Sx3 = 70, Sy3 = 70, threshold3 = 35.

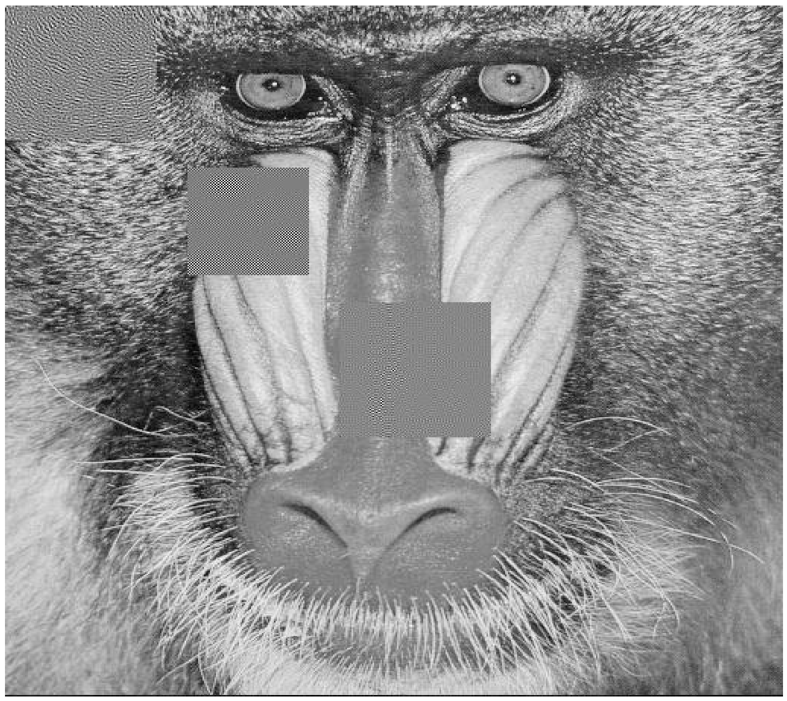


Figure 6: “highpass” function. Applying high pass filtering to the image in the frequency domain. The resulting image in the ROIs that the filtering was applied on shows enhanced edges depending on the threshold value passed. The used parameter configuration is: The used parameter configuration is: #roi = 3, x1= 0, y1= 0, Sx1= 100, Sy1= 100, threshold1 = 30, x2 = 450, y2 = 100, Sx2 = 150, Sy2 = 150, threshold2 = 55, x3 = 280, y3 = 280, Sx3 = 70, Sy3 = 70, threshold3 = 50.

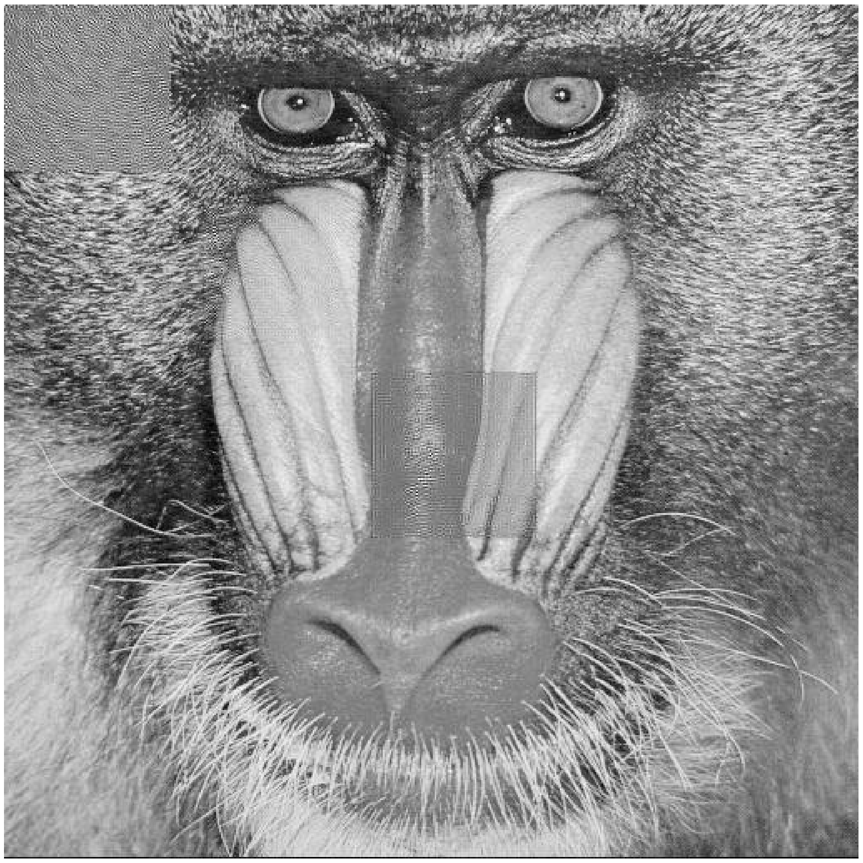


Figure 7: “edgesharp” function. Applying unsharp masking on high frequency pixels of the image in the frequency domain. The resulting image in the ROIs that the filtering was applied on shows a sharpened image depending on the multiplier value passed. The threshold for cutoff frequency is provided internally. The used parameter configuration is: The used parameter configuration is: #roi = 3, x1= 0, y1= 0, Sx1= 100, Sy1= 100, multiplier1= 10, x2 = 450, y2 = 100, Sx2 = 150, Sy2 = 150, multiplier2= 5, x3 = 280, y3 = 280, Sx3 = 70, Sy3 = 70, multiplier3= 7.

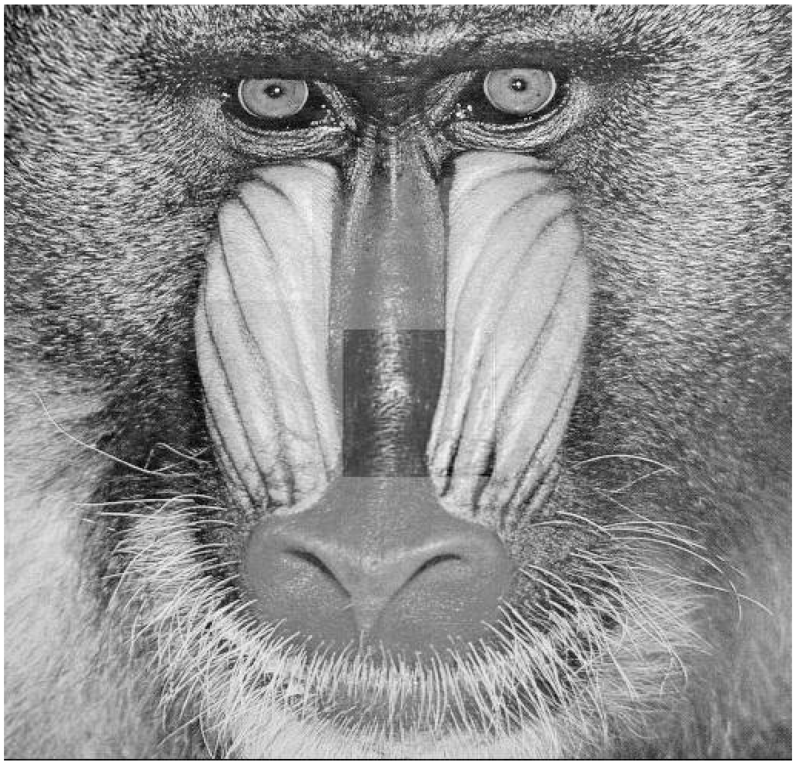


Figure 8: “edgesharp_low” function. Applying unsharp masking on low frequency pixels of the image in the frequency domain. The resulting image in the ROIs that the filtering was applied on shows a sharpened image depending on the multiplier value passed. The threshold cutoff for frequency is provided internally. The used parameter configuration is: The used parameter configuration is:#roi = 3, x1= 0, y1= 0, Sx1= 100, Sy1= 100, multiplier1= 20, x2 = 450, y2 = 100, Sx2 = 150, Sy2 = 150, multiplier2= 5, x3 = 280, y3 = 280, Sx3 = 70, Sy3 = 70, multiplier3= 10.

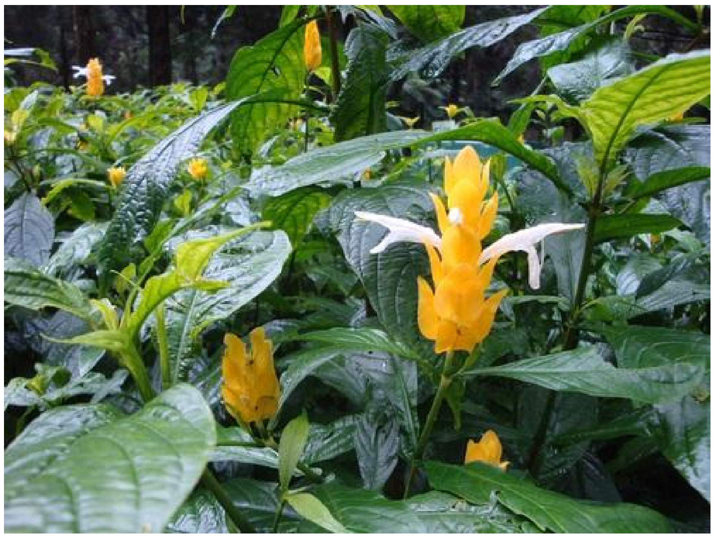
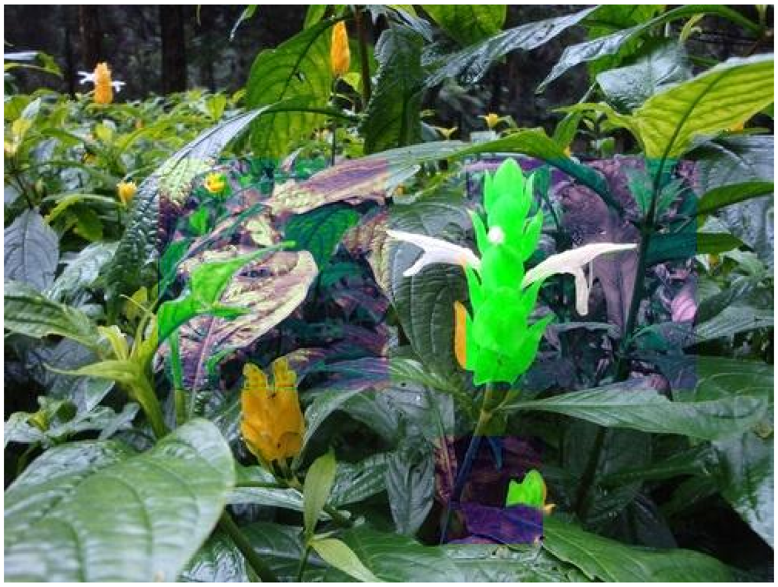


Figure 9: Original Color Image



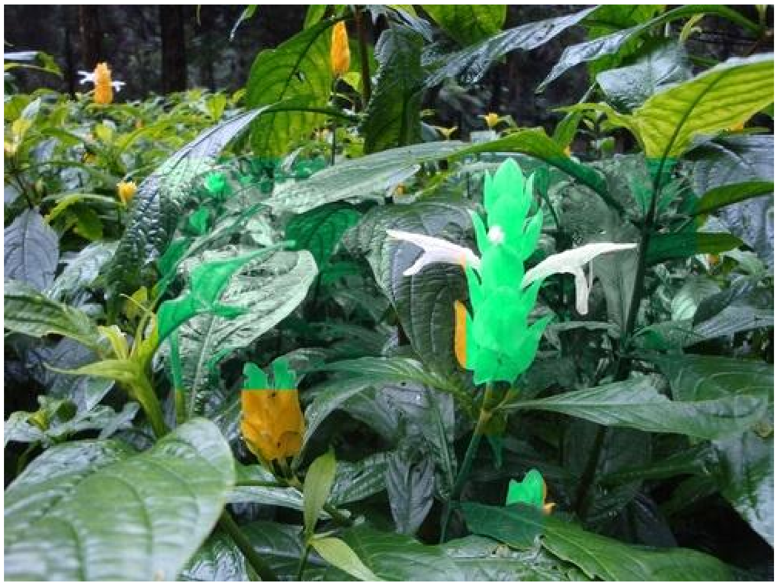


Figure 10: “lowpass_color” function. Applying low pass filtering to the H channel of the color image in the frequency domain. Figures shown are with and without normalization in range 0 -360. The used parameter configuration is: The used parameter configuration is: #roi = 3, x1= 100, y1= 100, Sx1= 150, Sy1= 150, threshold1 = 30, channel1 = 0, x2 = 300, y2 = 100, Sx2 = 150, Sy2 = 150, threshold2 = 50, channel2 = 0, x3 = 280, y3 = 280, Sx3 = 70, Sy3 = 70, threshold3 = 45, channel3 = 0.

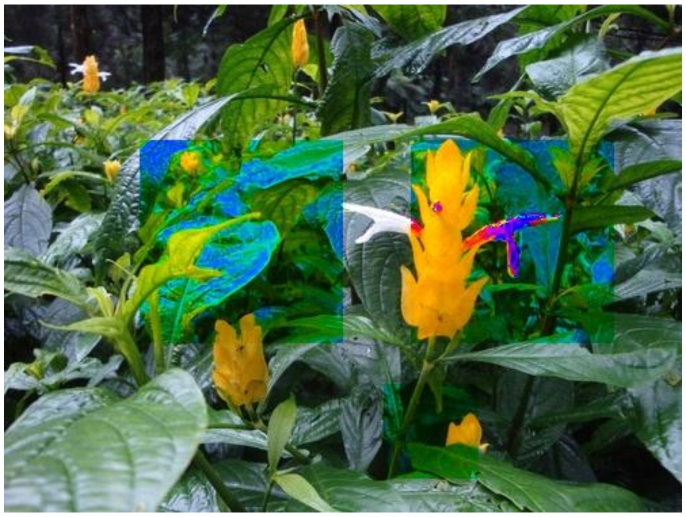


Figure 11: “lowpass_color” function. Applying low pass filtering to the S channel of the color image in the frequency domain. The used parameter configuration is: The used parameter configuration is: #roi = 3, x1= 100, y1= 100, Sx1= 150, Sy1= 150, threshold1 = 30, channel1 = 1, x2 = 300, y2 = 100, Sx2 = 150, Sy2 = 150, threshold2 = 50, channel2 = 1, x3 = 280, y3 = 280, Sx3 = 70, Sy3 = 70, threshold3 = 45, channel3 = 1.

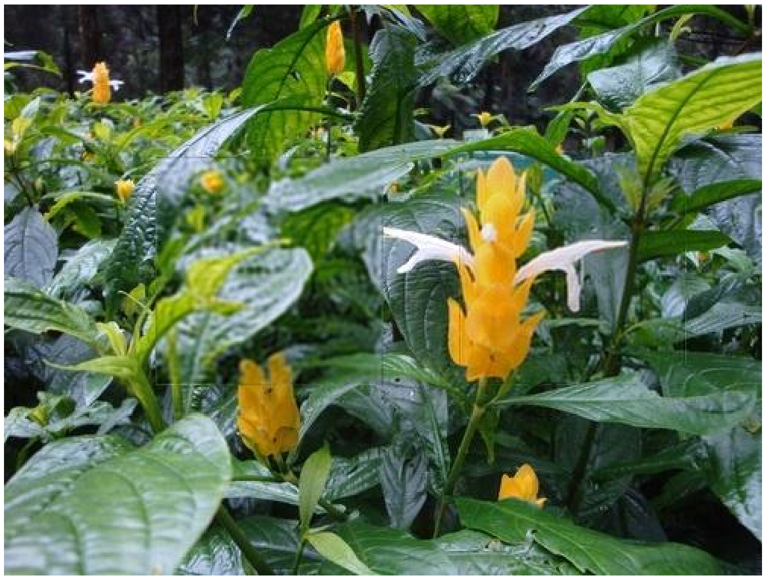
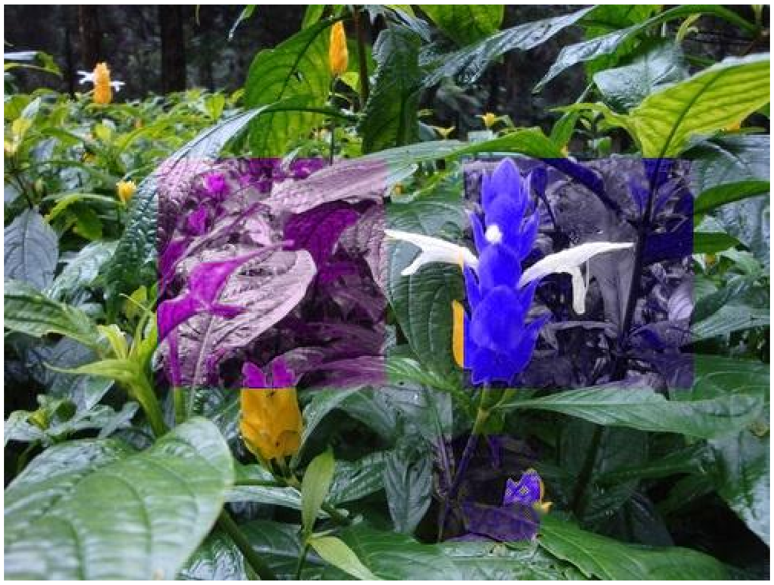


Figure 12: “lowpass_color” function. Applying low pass filtering to the V channel of the color image in the frequency domain. The ROIs will be blurred. This is the V component which has the values, intensities of the pixel. The used parameter configuration is: The used parameter configuration is: #roi = 3, x1= 100, y1= 100, Sx1= 150, Sy1= 150, threshold1 = 30, channel1 = 2, x2 = 300, y2 = 100, Sx2 = 150, Sy2 = 150, threshold2 = 50, channel2 = 2, x3 = 280, y3 = 280, Sx3 = 70, Sy3 = 70, threshold3 = 45, channel3 = 2.



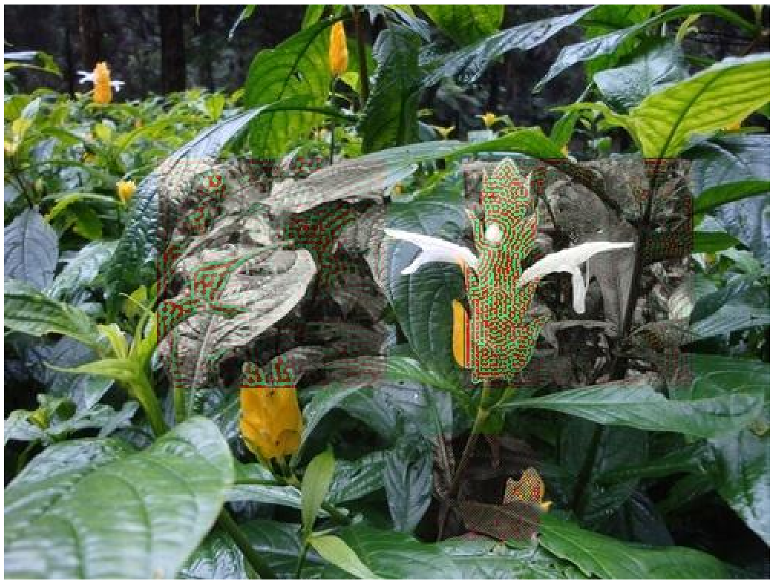


Figure 13: “highpass_color” function. Applying high pass filtering to the H channel of the color image in the frequency domain. The figures are without and with normalization in range 0 - 360. The used parameter configuration is: The used parameter configuration is: #roi = 3, x1= 100, y1= 100, Sx1= 150, Sy1= 150, threshold1 = 30, channel1 = 0, x2 = 300, y2 = 100, Sx2 = 150, Sy2 = 150, threshold2 = 50, channel2 = 0, x3 = 280, y3 = 280, Sx3 = 70, Sy3 = 70, threshold3 = 45, channel3 = 0.

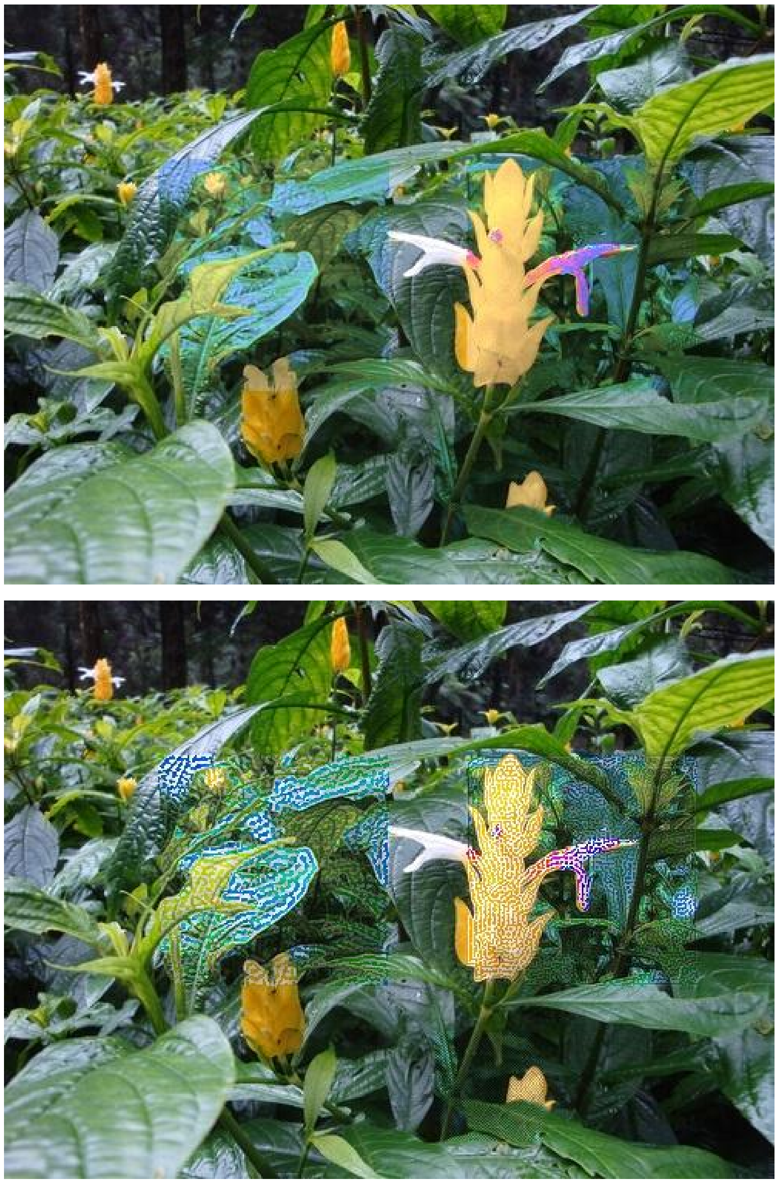


Figure 14: “highpass_color” function. Applying high pass filtering to the S channel of the color image in the frequency domain. The figures are without and with normalization in range 0-1. The used parameter configuration is: The used parameter configuration is: #roi = 3, x1= 100, y1= 100, Sx1= 150, Sy1= 150, threshold1 = 30, channel1 = 1, x2 = 300, y2 = 100, Sx2 = 150, Sy2 = 150, threshold2 = 50, channel2 = 1, x3 = 280, y3 = 280, Sx3 = 70, Sy3 = 70, threshold3 = 45, channel3 = 1.

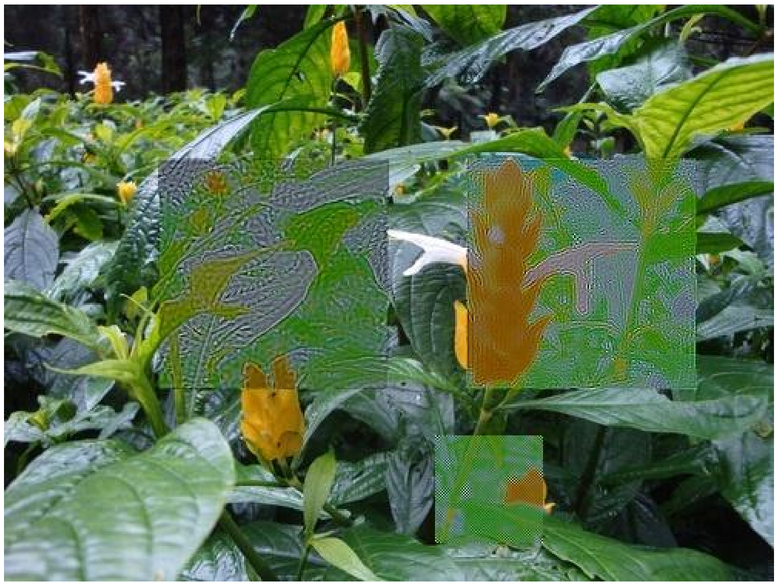


Figure 15: “highpass_color” function. Applying low pass filtering to the V channel of the color image in the frequency domain. The ROIs will have edges sharpened. This is the V component which has the values, intensities of the pixel. The used parameter configuration is: The used parameter configuration is: #roi = 3, x1= 100, y1= 100, Sx1= 150, Sy1= 150, threshold1 = 30, channel1 = 2, x2 = 300, y2 = 100, Sx2 = 150, Sy2 = 150, threshold2 = 50, channel2 = 2, x3 = 280, y3 = 280, Sx3 = 70, Sy3 = 70, threshold3 = 45, channel3 = 2.

4. Discussion of Results

We were able to convert grayscale images from image space to frequency domain using Discrete Fourier Transform from OpenCV. Then after making a few changes for display purposes we were able to visualize the frequency domain of the image as the amplitude spectrum with center on the center of the image rather than the corners. We also use the Inverse DFT to convert back to image space to get the original image back.

We experimented with low pass and high pass filtering on grayscale images in the frequency domain. Based on the results, we saw that low pass filtering performed a sort of smoothing on the image ROI where it was applied. The ROI became a bit blurry depending on the cutoff threshold we passed. Based on the results, we could say that low pass filtering could be useful whenever there is some noise in the image and we want to smooth it. As for high pass filtering, based on the results we saw that it worked as an edge detection, enhancement method. We say that because we were able to output an image consisting of mainly the edges for the ROI that was passed. So with that we can say that high pass filtering can be rather useful in edge detection.

We concluded that low pass filtering works like a smoothing method. We have tested multiple smoothing algorithms in the past. One main advantage that low pass filtering would have is the fact that we can optimize the smoothing effect by effectively changing the cutoff threshold that we pass. A higher threshold wouldn't perform a significant smoothing, but a smaller one would. So depending on the level of noise present in the image, we have a range of values we can use to adapt to the image and perform the right level of smoothing. To me that is a strong advantage.

Based on the results we concluded that high pass filtering works like an edge detection method. We have tested multiple smoothing algorithms in the past like Sobel, Canny etc. While the edges that are detected are not as clear as perhaps what Canny would output, they are nevertheless rather good. Additionally, the level of edge detection in high pass filtering will depend on the cutoff threshold that we pass, so that adds a layer of adaptiveness which can be very useful as compared to other edge detection algorithms out there.

We performed image sharpening using unsharp masking on both low and high frequency pixels of the image in frequency space. The resulting image showed some sharpening in the frequencies that were applied. The edge detection or smoothing was more distinct. The sharpening effect depends on the multiplier value we pass as user input.

Finally, we applied low and high pass filtering to color images in HSV color space. We applied the filtering to each component H,S and V. Since H and S are hue and saturation, applying low and high pass filtering to them didn't produce really good results in terms of smoothing or edge detection. However, applying it to the V (value) component which has the intensity values, provided much better results. The low pass filtering in the V component resulted in smoothing/blurring in the ROIs which is what we were expecting. The high pass filtering in the V component resulted in edge detection of the color image, which again was something we were expecting. So applying low and high pass filtering in color images in HSV color space can be useful when we apply it to the V component.

5. Conclusion

Overall, in this assignment we were able to experiment with Fourier Transformation and frequency filtering methods such as low pass and high pass. We also tested unsharp masking for image sharpening. We converted grayscale and color images from image space to frequency domain using Discrete Fourier Transformation of OpenCV and applying the Inverse Discrete Fourier Transformation to return to image space after the filtering was performed. We saw that low pass filtering can be used for smoothing and blurring and it is very effective at that, while high pass can be used for edge detection and again it was rather effective. While other algorithms can produce really good results as well, filtering in the frequency domain has advantages in terms of adaptivity because of the cutoff thresholds we can use and filters we use. Applying these filters on color images of HSV color space was effective only when applied to the V component. The H and S components changed certain features of the image which wasn't ideal. Ultimately, this assignment provided a productive way to further improve our knowledge in Image Processing.