Computer Vision: Image Enhancement
Allen Lau

I. Writing Assignments

1. (20 points) Generate the histogram of the image you are using, and then perform a number of histogram operations (at least including contrast enhancement, thresholding, and equalization) to make the image visually better for either viewing or processing (10 points). If it is a color image, please first turn it into an intensity image and then generate its histogram. Try to display your histograms of the original and the processed images (5 points), and make some observations of the images based on their histograms (5 points). What are the general distributions of the intensity values of each histogram? How many major peaks and valleys does your histogram have, and how do they behave? How could you use the histograms to understand, analyze or segment the image?

Figure 1: ID Picture Intensity Image

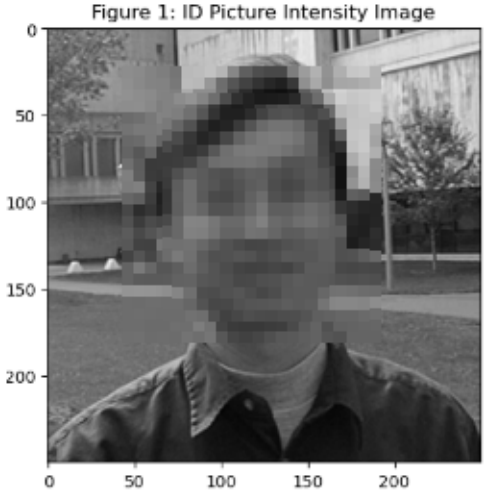


Figure 2: Distribution of Intensity Values, ID Picture

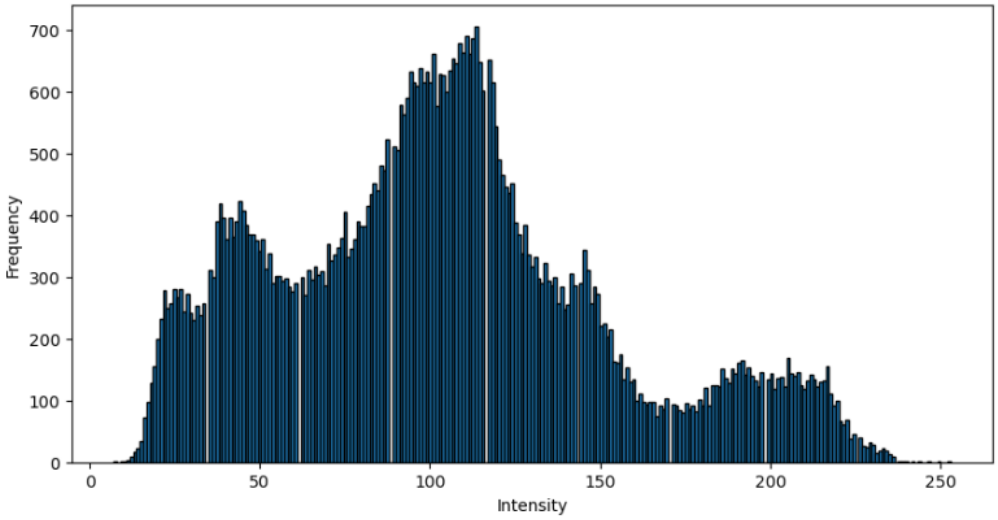


Figure 1 displays the intensity image of the colored ID picture, which was created via the simple average approach (summing the RGB values of each pixel location together and then dividing by 3). The histogram of this intensity image is seen in Figure 2 above, where it is observed that the mode exists at around 115 intensity magnitude and most of the values exist between ~10 to ~240. This broad range of intensity values indicates that there is good contrast in the image. Additionally, the distribution is slightly left-skewed, which means that it is slightly darker of an image. These observations are visually noticed since areas like the shirt, grass, and hair are darker (lower intensity values), but we do see contrast between them and lighter areas like the building in the background. Moreover, there exists 3 major peaks and 2 major valleys, which can be used as starting points for picking threshold values for segmentation or analyzing what exists in the image.

For contrast enhancement, the goal is to stretch the distribution of pixel intensities such that they utilize the entire range of intensity values from 0 to 255. This is done via implementing the following equation for linear scaling, where I' is the new pixel intensity, K is 255 (upper bound of intensity value), Imin is the minimum intensity value in the original image, and Imax is the maximum intensity value in the original image:

$$I' = \left[\frac{K}{Imax - Imin}\right] * I + \left[\frac{K}{Imax - Imin}\right] * Imin$$

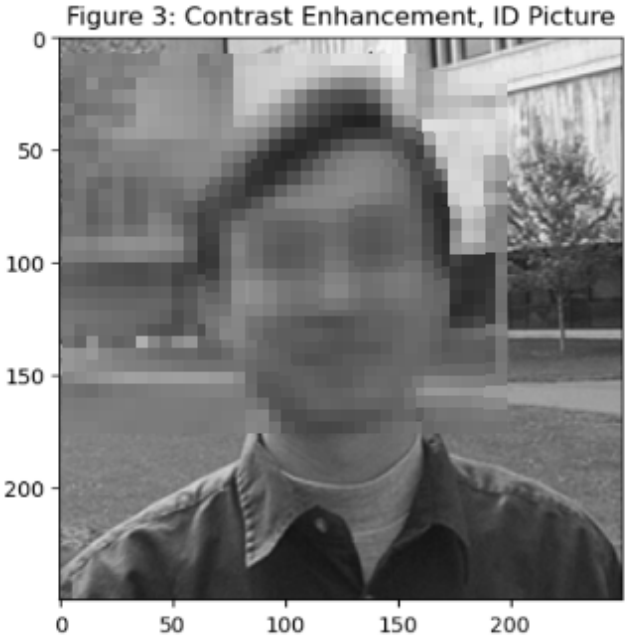


Figure 4: Distribution of Intensity Values, Contrast Enhancement

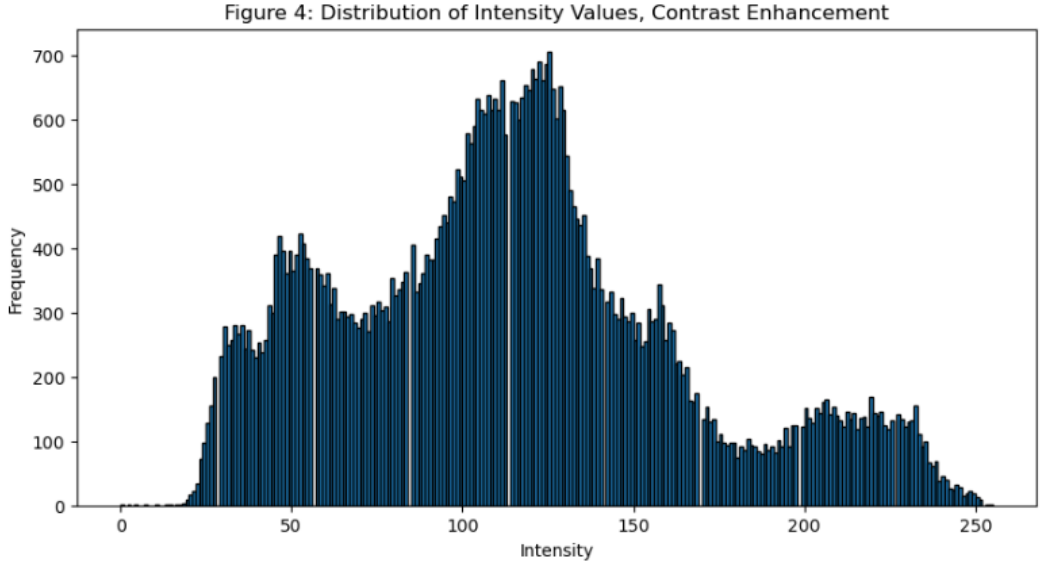


Figure 3 above displays the ID picture after contrast enhancement and Figure 4 displays its histogram of pixel intensities. It is observed that the histogram now spans from 0 to 255 instead of majorly from ~10 to ~240, which means there is more difference between the lightest and darkest parts of the image. It is also observed that the histogram has shifted slightly right, which indicates that the contrast enhanced image is slightly lighter than the original image. However, since the original image already spans much of the entire range, the changes to the image are difficult to visually see.

To threshold the image, values of 45, 115, and 210 are chosen since they are the peaks of the histogram and 70 and 175 since they are valleys. The thresholded images are resulted from setting pixel values above the threshold in the original image to 255 and pixel values below to 0. Picking the peaks or valleys of the histogram as threshold values are useful to segment objects within the image since objects often correspond to peaks and backgrounds often correspond to valleys. The results are seen in the below figures:

Figure 5: ID Picture Thresholding via Peak Values

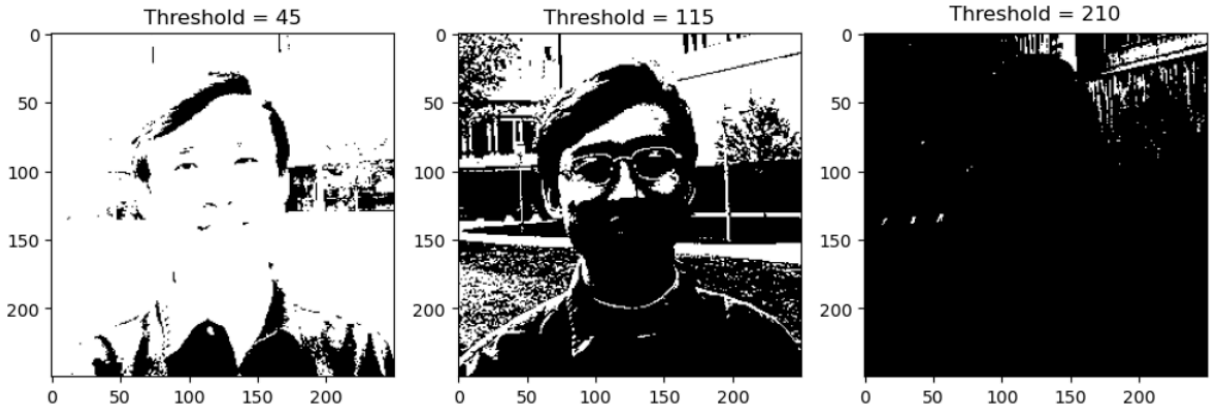
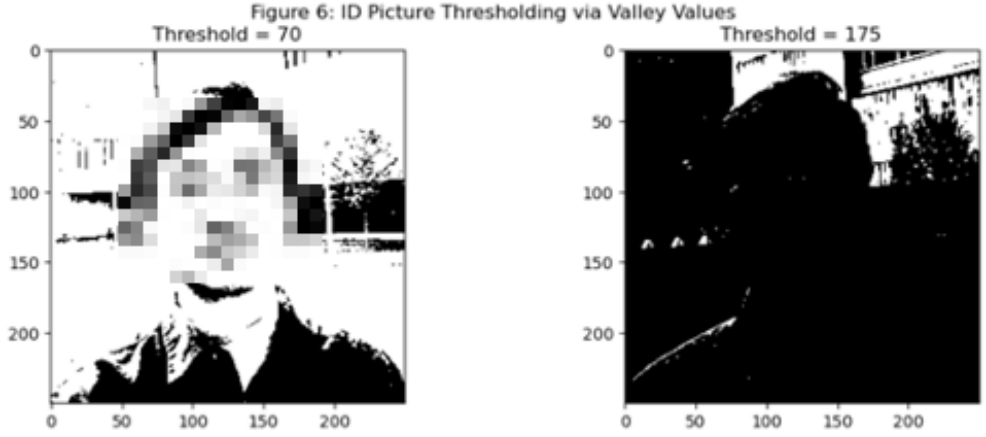
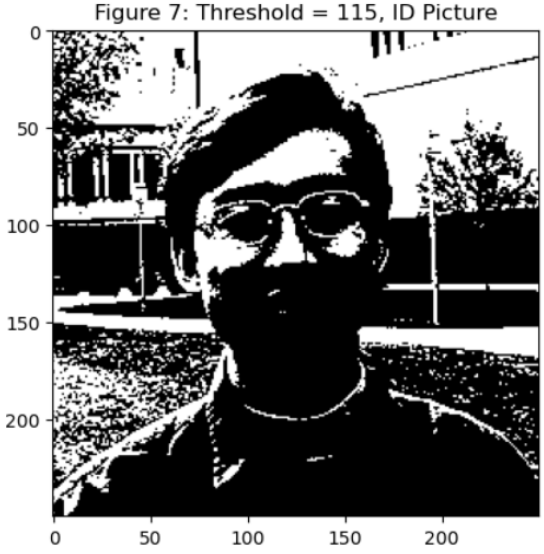


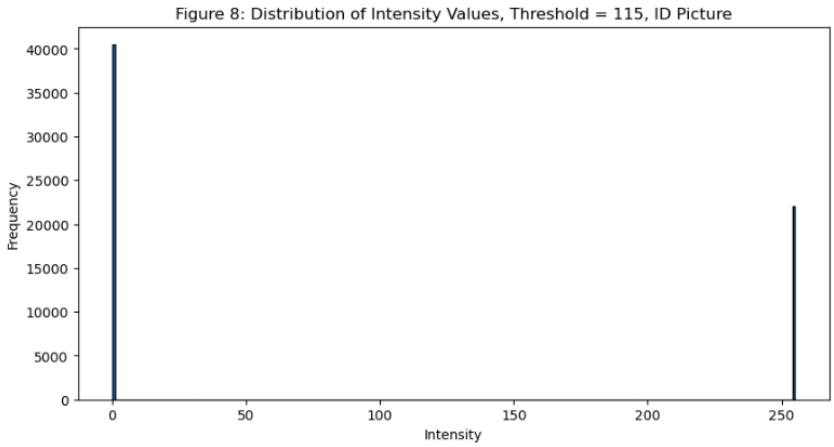
Figure 6: ID Picture Thresholding via Valley Values



It is observed that using a threshold of 115 results in the best visual performance since it clearly segments the person within the image from the background, while also keeping some of the details within the image, like the glasses, trees, and structure in the background building. Figures 7 and 8 depict the best thresholded image and its histogram below, where it is seen that there are more pixels that are 0 than 255:

Figure 7: Threshold = 115, ID Picture



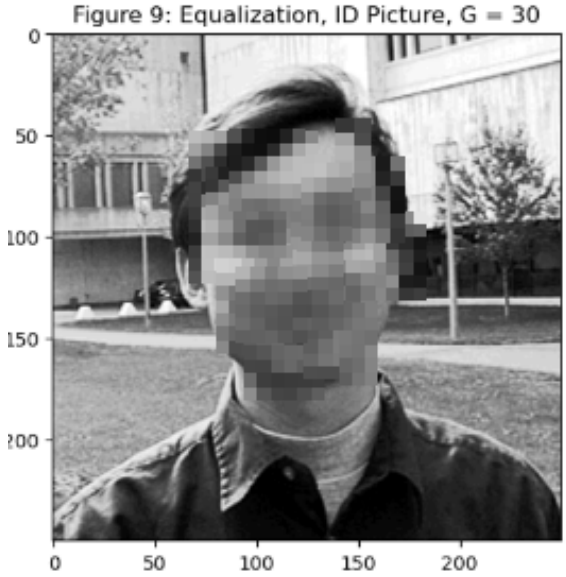


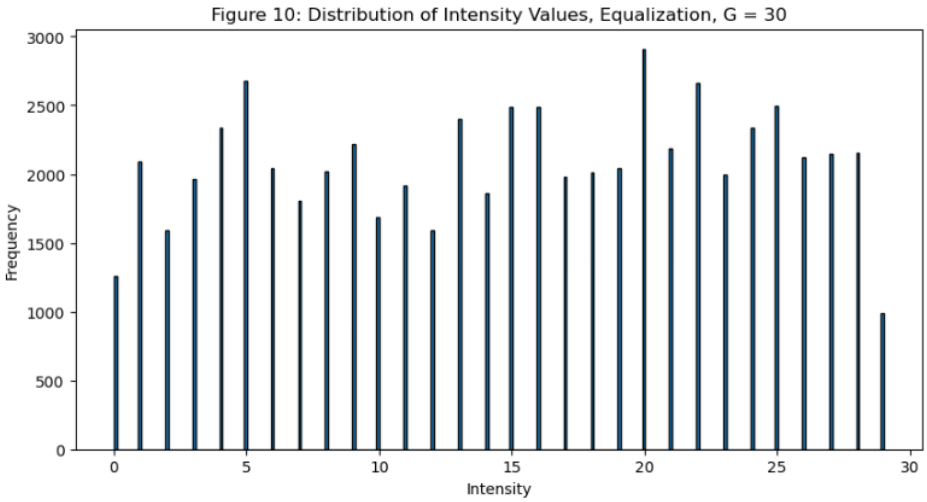
The concept of equalization describes the process of redistributing the intensity values of an image, such that the distribution of intensity values is more uniform. This allows for an enhancement in contrast in the image since intensity value ranges are used more uniformly and throughout the range from 0 – 255. To perform equalization, the below equation is used, where I' is the equalized intensity value, I is the intensity value of the original image, CH(I) is the value at I of the cumulative histogram, and Np is the number of pixels occupying each gray level. To calculate Np, the image shape (M, N) and number of gray levels G are used:

$$I' = MAX\left(0, round\left(\frac{CH(I)}{Np}\right) - 1\right)$$

$$Np = \left(\frac{MN}{G}\right)$$

Figure 9 depicts the equalized ID image using a defined 30 gray levels and Figure 10 depicts the corresponding distribution:





It is observed that the image has higher contrast than the original image in Figure 1. For example, the background building appears much lighter and the shirt appears much darker. This is because we are redistributing the pixel intensities to be more uniform, so more pixels exist in the lowest and highest pixel intensity bins than the original image. In other words, we have removed the peaks and valleys of the original distribution.

2. (20 points) Apply BOTH the 1×2 operator and Sobel operator to your image and analyze and compare the results of the gradient magnitude images (including vertical gradients, horizontal gradients, and the combined) (10 points). Please don't forget to normalize your gradient images, noting that the original vertical and horizontal gradients have both positive and negative values. I would recommend you to display the absolute values of the horizontal and vertical gradient images. Does the Sobel operator have any clear visual and processing advantages over the 1×2 operator? Any disadvantages (5 points)? If you subtract the normalized 1×2 gradient image from the normalized Sobel gradient image, are there any residuals? You might use two different types of images: one ideal man-made image, and one image of a real scene with more details (5 points).

The 1x2 and Sobel Kernels are depicted below and are used to detect the vertical, horizontal, and combined gradient images, via convolutional operations:

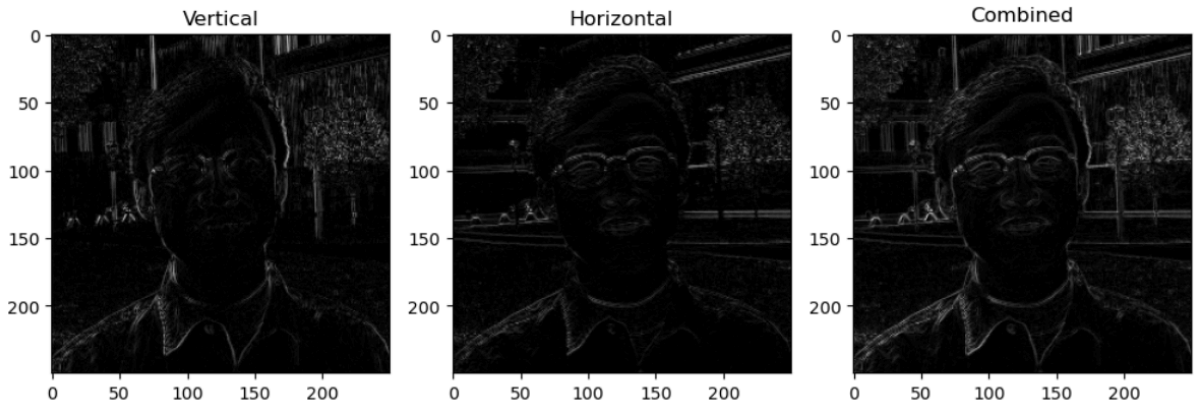
$$1x2\ Operator: [-1,1]$$

$$Sobel\ Operator: s1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, s2 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

To address issues on the border of the image while performing the convolutional operations, for the purposes of this report, zero-padding, or adding a border of zero intensity pixels to the border of the image, is used. It is important to note that because

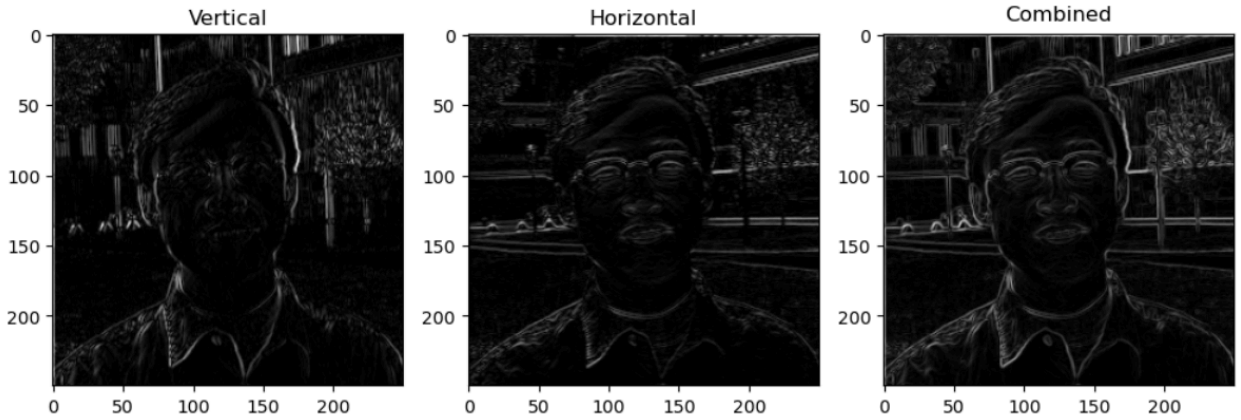
zero-padding is used, edges may be detected on the border of the image. The resulting gradient images for the 1x2 operator are shown below:

Figure 11: 1x2 Kernel Vertical, Horizontal, Combined Gradient Images



The resulting gradient images for the Sobel operator are shown below:

Figure 12: Sobel Kernel Vertical, Horizontal, Combined Gradient Images



From an initial inspection, it is apparent that the two kernels do emphasize vertical and horizontal oriented lines based on the configuration of the kernel. For example, both results show strong edges (as indicated by larger values of intensity value, or closer to white) like the vertical lines of the hair's side, building's corners, or lamp posts or the horizontal lines of the glasses, walkway, and building's roof clearly in the respective vertical and horizontal images. In terms of the differences observed, it is apparent that the Sobel kernel performs better with capturing edges that are not directly oriented vertically or horizontally. For instance, for more complex edges like the nose and mouth, which consists of many small, slanted lines, the Sobel kernel performs better at capturing this information by assigning a lower intensity value based on the magnitude of the component in either the horizontal or vertical directions. In addition, the magnitude and direction of the gradient at each pixel can be calculated by combining the information in the vertical and horizontal gradient images, as seen in Figure 12.1

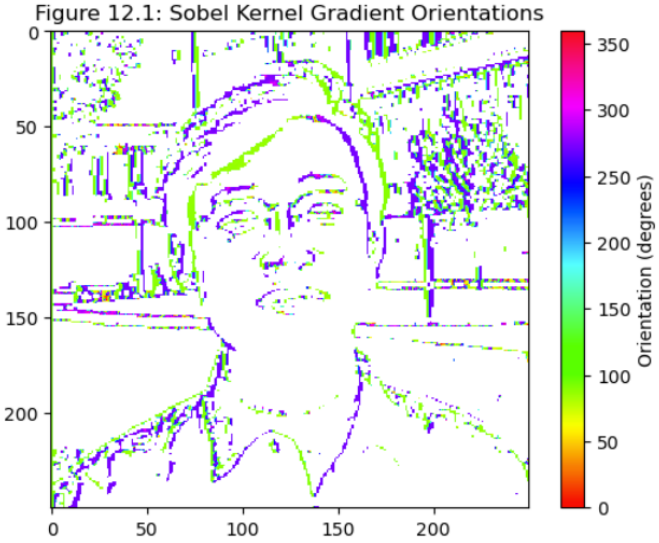
below. In contrast, the 1x2 operator is useful for directly finding the horizontal and vertical edges. From a processing perspective, the 1x2 operator has a clear advantage due to only having two coefficients so it is faster to compute than the Sobel operator.

The magnitude and orientation equations for the Sobel Kernel are as follows:

$$Edge\ Magnitude = sqrt(s1^2 + s2^2)$$

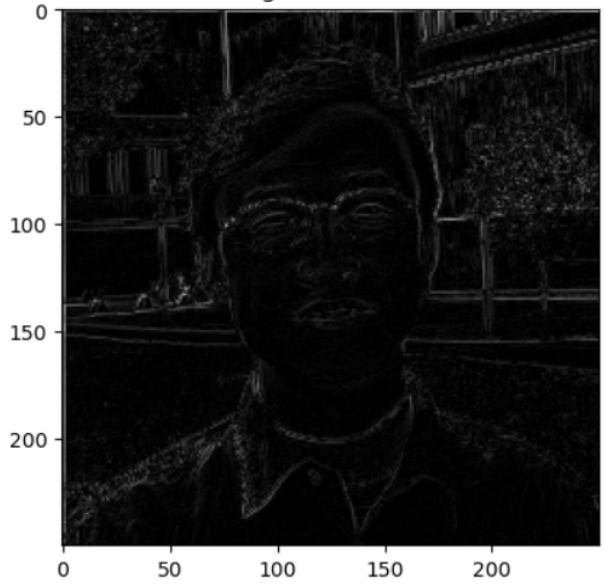
$$Edge\ Direction = \tan^{-1}\left(\frac{s1}{s1}\right)$$

Figure 12.1: Sobel Kernel Gradient Orientations



To compute the residuals between the Sobel and 1x2 Operator results, the difference between the combined gradient images is resulted as seen below in Figure 13. The residuals plot is useful for checking what information the Sobel filter captured that was not captured by the 1x2 filter.

Figure 13: Combined Gradient Images Residuals between Sobel and 1x2 Kernels



As seen in Figure 13, the lighter the value, the greater the resultant residual between the Sobel and 1x2 results. It is observed that some of the greatest residuals are seen in edges that are not completely vertical or horizontal. This is because the 1x2 kernel does not capture lines that are not directly horizontal or vertical well. For example, the slanted lines of the shoulder, collar, and some building features are observed to be high residuals. It is observed that certain vertical and horizontal features do show as higher residuals, which may be attributed to the differences in the resultant intensity magnitudes after applying the Sobel and 1x2 kernels to the image. For instance, Sobel may assign a value of 150 to a pixel but 1x2 may assign a value of 80, even though they are the same pixel belonging to the same edge.

To repeat the above procedure on an ideal problem, Figure 14 depicts a house image, which is composed of a series of vertical, horizontal, and slanted lines. It is expected that the 1x2 and Sobel kernel perform well on extracting the vertical and horizontal lines; however, the Sobel should perform better on extracting the edges that are slanted.

Figure 14: House Intensity Image

After applying the two kernels to the house image, the resultant vertical, horizontal, and combined gradient images are resulted below:

Figure 15: 1x2 Kernel Vertical, Horizontal, Combined Gradient Images, House

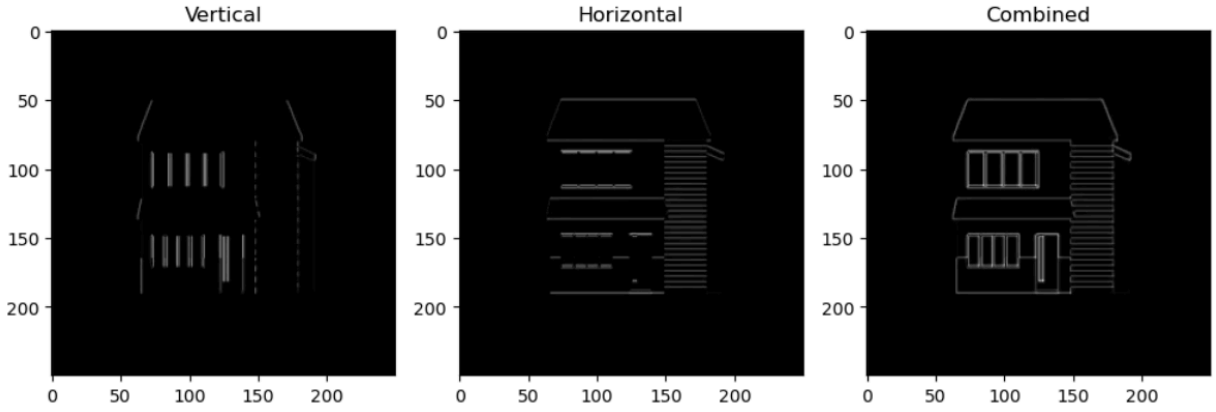


Figure 16: Sobel Kernel Vertical, Horizontal, Combined Gradient Images, House

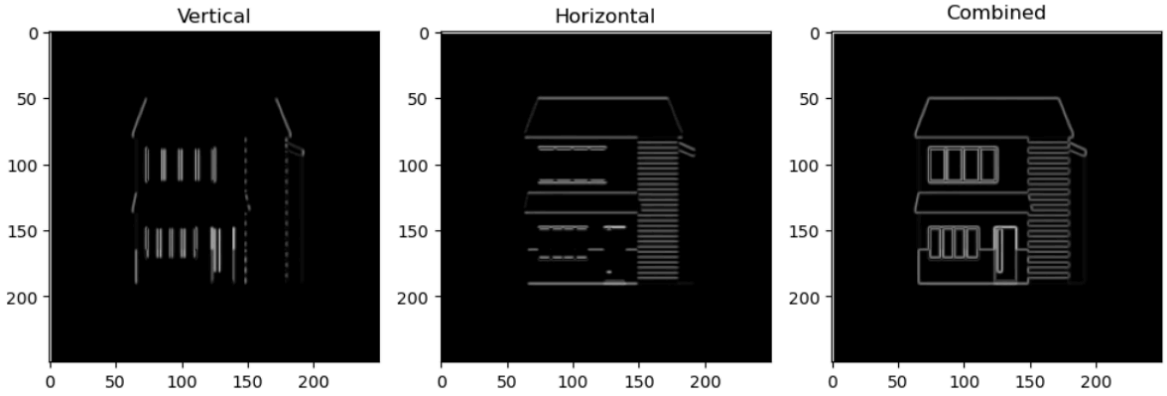


Figure 17: Combined Gradient Images Residuals between Sobel and 1x2 Kernels, House

It is observed that both kernels perform well identifying the vertical and horizontal edges. As a result, if the main goal is to extract these specific lines, then using 1x2 operator may be the best choice due to its simplicity and faster runtime. However, for edges that are slanted, the Sobel kernel performs better. This is apparent in the combined gradient images where the Sobel image's edges are clearer and more pronounced.

3. (20 points) Generate edge maps of the above two combined gradient maps (10 points). An edge image should be a binary image with 1s as edge points and 0s as non-edge points. You may first generate a histogram of each combined gradient map, and only keep certain percentage of pixels (e.g. 5% of the pixels with the highest gradient values) as edge pixels (edgels) . Please study what is the best percentage for a specific image, and why. Use the varying percentage to automatically find a corresponding threshold for the gradient magnitudes, and then pick up the one having the best visual performance. In your report, please write up the description and probably equations for finding the threshold, and discuss what percentage is a good value (5 points). You may also consider to use local, adaptive thresholds to different portions of the image so that all major edges will be shown up nicely (5 points). In the end, please try to generate a sketch of an image, such as the **ID image of Prof. Zhu**.

To aid with the process of generating an edge map, the histograms of the combined gradient maps are visualized. It is observed that most of the pixels in both gradient maps exist at the lower end of the possible intensity value range, which indicates both maps are dark. One difference that is illustrated is that the Sobel kernel results in a gradient map with higher contrast than the 1x2 kernel results. These distributions are used to compute the best threshold of gradient magnitude based on an input of the percentage of pixels kept.

Figure 18: Distribution of Intensity Values, Combined Gradient Map using 1x2 Kernel

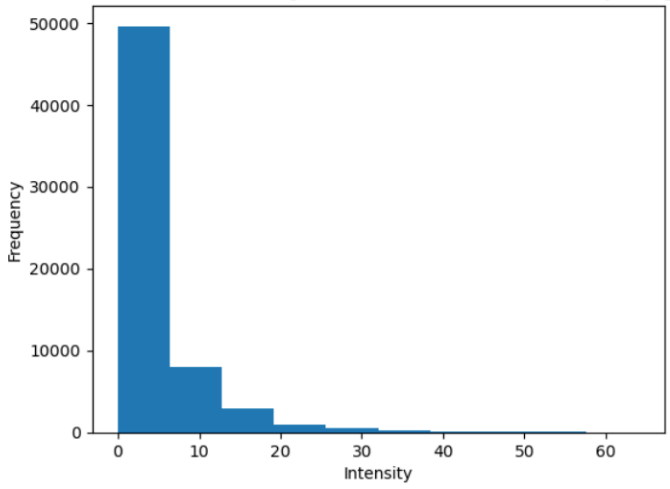
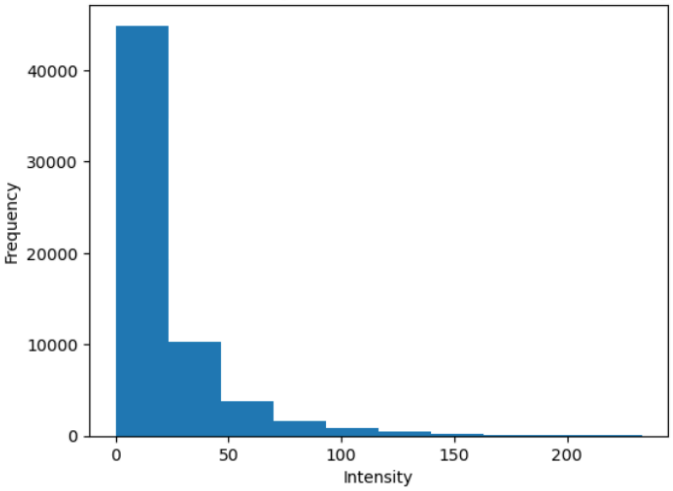


Figure 19: Distribution of Intensity Values, Combined Gradient Map using Sobel Kernel



The procedure for computing the threshold given an input of percentage of pixels kept is described below:

1. Compute the histogram of the image, which gives information about the frequency of pixel intensities.
2. Compute the sum of the total number of pixels within the image.
3. Starting from the max pixel intensity value bin in the histogram, iterate backwards towards the min pixel intensity value.
4. Keep a running sum of the number of pixels as you iterate backwards and compute the percentage of pixels for this running total.

$$let\ current\ sum = 0$$
$$Upate\ current\ sum\ for\ each\ iteration:$$
$$current\ sum = current\ sum + current\ count$$

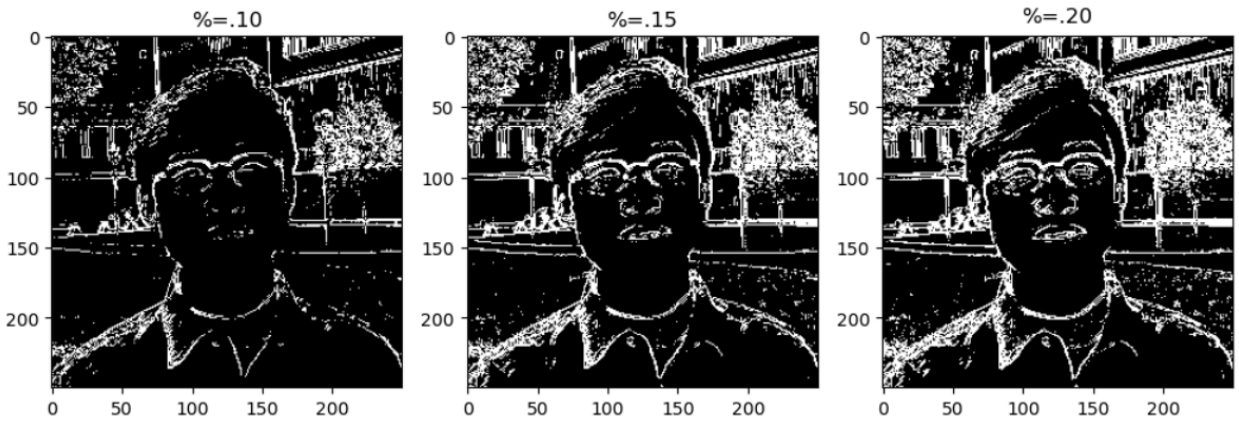
5. Return the pixel intensity value corresponding to the bin once the percentage of pixels for the running total equals or is greater than the input/target percentage.

$$percentage\ of\ running\ total = \frac{current\ sum}{total\ sum}$$

6. Experiment with various input percentages/thresholds to give the best visual performance.

Using the above procedure, the various input percentages are used to find the best visually performing edge map:

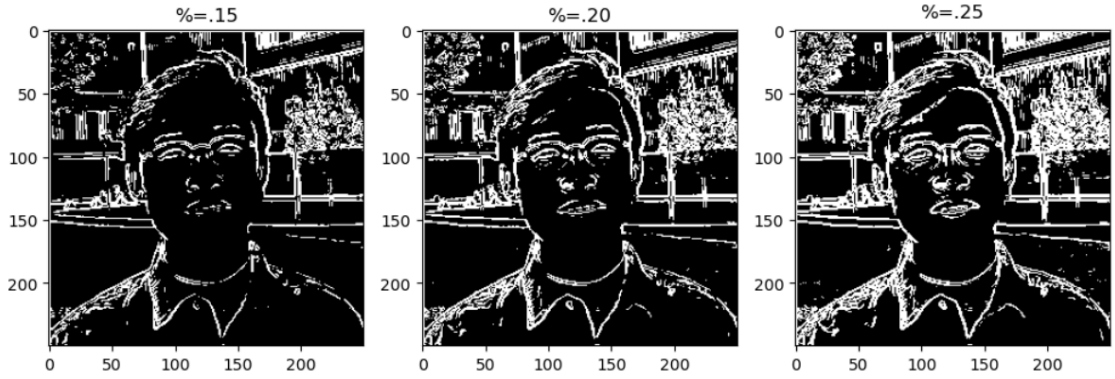
Figure 20: Edge Map of Combined Gradient Map using 1x2 Kernel



It is observed that the best performing edge map results from an input percentage of 15%. This is because at lower percentages, a loss of features occurs. For instance, in the 10% image, the nose, mouth, and hair edges are not captured. At higher percentages, the amount of noise increases, which would impact the ability to perform segmentation and object detection easily. This is apparent in the increased noise in the background of the 20% image.

The best performing edge map for the Sobel kernel gradient map resulted from taking the top 20% of pixels:

Figure 21: Edge Map of Combined Gradient Map using Sobel Kernel



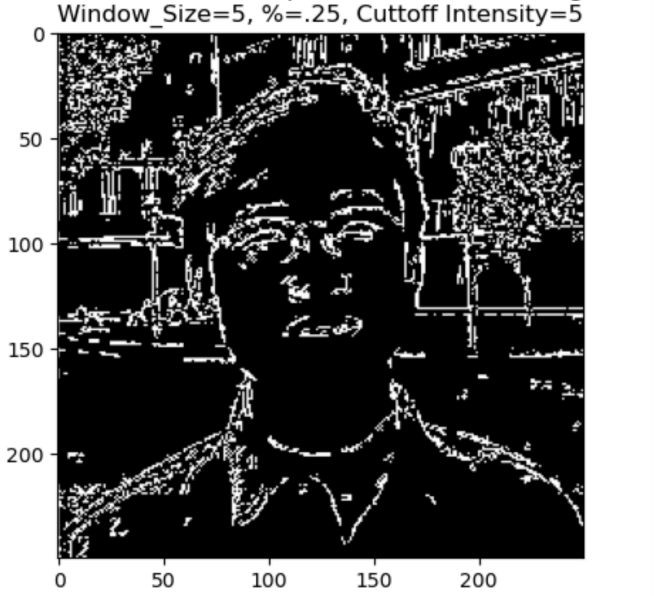
For similar reasons as discussed in the 1x2 edge map results, the best visually performing edge map for the Sobel gradient map is an input percentage of 20%. Again, we see that there is a balance between retaining important edge structures but reducing the amount of noise within the image.

Adaptive thresholding is a useful approach for determining good thresholds within local areas of the input image. This results in a potentially better results since a global threshold may not be the best threshold in all areas of the gradient image. In other words, when using a global threshold, we may be filtering out local edges that may have been assigned lower intensity values in the gradient image. To perform adaptive, local thresholding to produce an edge map, the following procedure is used:

1. Define a window size. The window size is a N x N local image within the larger image where we will perform operations on.
2. Use the window size to iterate through the original image.
3. For each iteration, compute the threshold given an input of the percentage of pixels to keep. If the threshold is below a certain defined cutoff intensity, we do not consider it a major edge. This is done to reduce the noise of the resultant edge map.
4. Compute the edge map for the local image, setting values greater than the threshold to 255 and values lower to 0.
5. Repeat the process to generate the edge map via adaptive, local thresholding.

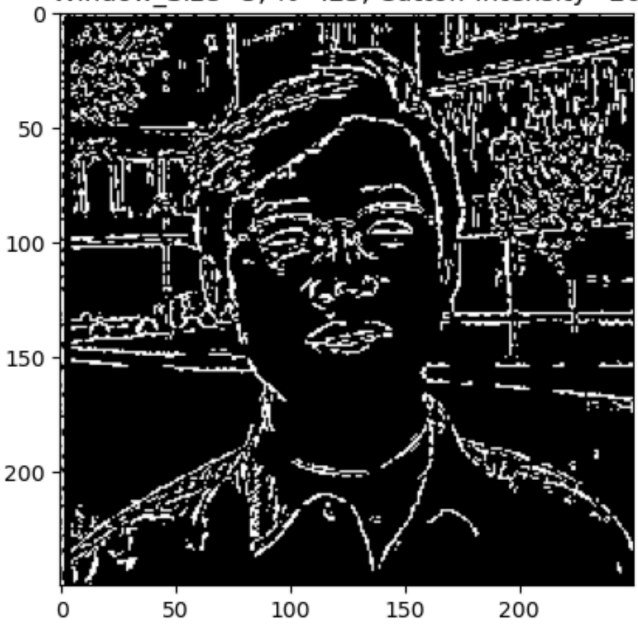
After experimenting with the hyperparameters of window size, percent pixels to retain, and cutoff intensity, the best performing edge map for the 1x2 kernel resulted from a window size of 5 pixels, percent pixels to retain of 25%, and a cutoff intensity of 5:

Figure 22: Edge Map of Combined Gradient Map using 1x2 Kernel and Adaptive, Local Thresholding Window_Size=5, %=.25, Cuttoff Intensity=5



The best performing edge map for the Sobel kernel resulted in a window size of 5, a percent of 25% and a cutoff intensity of 20.

Figure 23: Edge Map of Combined Gradient Map using Sobel Kernel and Adaptive, Local Thresholding Window_Size=5, %=.25, Cuttoff Intensity=20



Figures 22 and 23 were the best performing edge maps due to the balance of retaining features and reducing noise within the resultant image. It is observed that the edges on these images are clearer and the noise has been reduced dramatically using the local, adaptive thresholding approach. The hyperparameter that affected the result of the image was the window size. For example, will larger window sizes, the resultant edge map performed worse since the calculated threshold would be applied to a larger area of the image.

4. (20 points) What happens when you increase the size of the edge detection kernel from 1×2 to 3×3 and then to 5×5 , or 7×7? Discuss (1) computational cost (in terms of number of operations, and the real machine running times – 5 points); (2) edge detection results (5 points) and (3) sensitivity to noise, etc. (5 points). Note that your larger kernel should still be an edge detector. Please list your kernels as matrices in your report and tell us what they are good for (5 points).

The 1x2 kernel is increased to the sizes of 3x3, 5x5, and 7x7 to explore the differences in the resultant edge detection and computational costs. The below displays the kernels, where v describes the kernel that will find the vertical gradient and h will find the horizontal gradient:

1x2: v = [-1,1]
h = [-1,1]

3x3: v = [[-1,0,1],[-1,0,1],[-1,0,1]]
h = [[1,1,1],[0,0,0],[-1,-1,-1]]

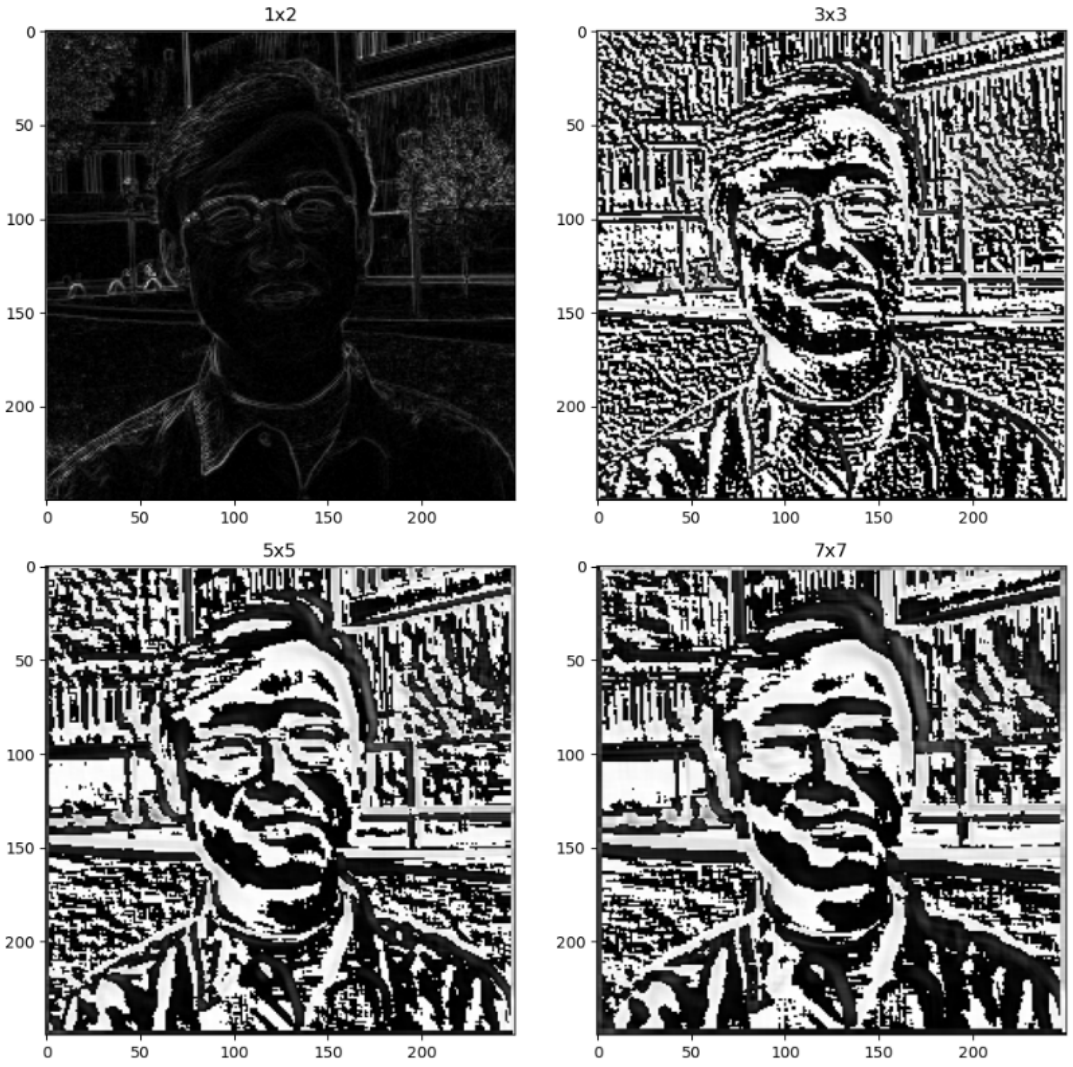
5x5: v = [[-1,0,0,0,1],[-1,0,0,0,1],[-1,0,0,0,1],[-1,0,0,0,1],[-1,0,0,0,1]]
h = [[1,1,1,1,1],[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0],[-1,-1,-1,-1,-1]]

7x7: v = [[-1,0,0,0,0,0,1],[-1,0,0,0,0,0,1],[-1,0,0,0,0,0,1],[-1,0,0,0,0,0,1],
[-1,0,0,0,0,0,1],[-1,0,0,0,0,0,1],[-1,0,0,0,0,0,1]]
h =[[1,1,1,1,1,1,1],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],
[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[-1,-1,-1,-1,-1,-1,-1]]

In terms of computational costs, the smaller the kernel, the less computationally expensive it is to compute the convolutional operation for each pixel. For instance, for the 1x2 kernel, it only requires 2 multiplication and 1 addition operation for one pixel. For the 7x7 kernel, it requires 49 multiplication operations and a summation of all resultant products for one pixel. This is exponentially more computationally expensive, which also results in longer runtimes for computing the gradient maps.

To illustrate the differences in edge detection and sensitivity to noise, Figure 24 below displays the resultant combined gradient images when the 1x2 kernel is increased to 3x3, 5x5, and 7x7.

Figure 24: Increasing 1x2 Kernel Size Combined Gradient Image



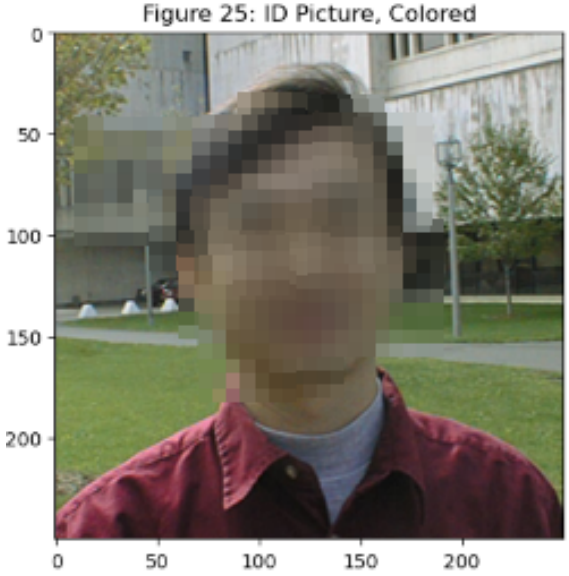
The above figure displays the difference in edge detection results when the kernel size is increased. For a smaller kernel, it will be more sensitive to small details and small changes in intensity. This allows it to detect small edges but be more affected by noise. In contrast, for a larger kernel, it will not be as affected by small details, edges, or noise since it looks at a larger neighborhood of pixels to determine the intensity value of the pixel. This is easily apparent in areas like the glasses and eyes, where in the smaller kernel sizes (1x2 and 3x3), we can see the edges of the glasses and eyes. However, at a larger kernel size of 5x5, the edges of the glasses and eyes are now fusing with the rest of the face since with larger kernels, the edge width increases. At 7x7, we no longer can make out the fine details/edges of those structures. Lastly, since the larger kernel results in wider edges, the ability to accurately determine the actual position of the edge is reduced.

In terms of when certain kernel sizes should be used, smaller kernels should be prioritized if the important features of an image are small or if there is not much noise in the scene. Additionally, if the geometry of the object being preserved is highly important, then using a smaller kernel will be better. Larger kernels should be used if there is a lot of noise within the image that needs to be removed from the resultant edge detection result or if the important features of the image are large and prominent.

5. (20 points) Suppose you apply the Sobel operator to each of the RGB color bands of a color image. How might you combine these results into a color edge detector (5 points)? Do the resulting edge differ from the gray scale results? How and why (5 points)? You may compare the edge maps of the intensity image (of the color image), the gray-scale edge map that are the combination of the three edge maps from three color bands, or a real color edge map that edge points have colors (5 points). Please discuss their similarities and differences, and how each of them can be used for image enhancement or feature extraction (5 points). Note that you want to first generate gradient maps and then using thresholding to generate edge maps. In the end, please try to generate a color sketch of an image, such as the **ID image of Prof. Zhu**. You may also consider local, adaptive thresholding in generating a color edge map.

To explore edge detection with colored images, the below colored ID picture will be used:

Figure 25: ID Picture, Colored



The procedure for performing edge detection on colored images is as follows:

1. Split the color bands of a colored image into 3 separate intensity images.
2. Using the defined Sobel operator, apply the kernel on each intensity image separately, for every pixel. This results in 3 combined gradient images for the red, green, and blue channels.

3. Compute the edge map of each intensity image by determining an appropriate threshold given an input percentage of how many of the top intensity pixels to keep.
4. Combine edge maps (or gradient images) together to create a real color edge map (or real color combined gradient image), such that the red edge map is the first channel of a NxNx3 image, the green edge map is the second, and the blue is the third.

Following the above procedure, Figure 26 below depicts the edge maps for each colored channel using a global thresholding value of 20.

Figure 26: Edge Maps of RBG Channels, Global Thresholding; Threshold=20

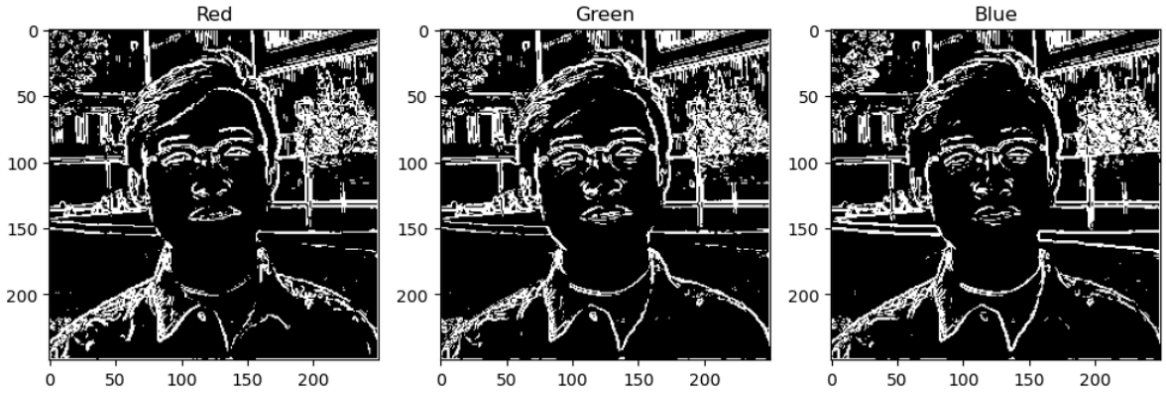
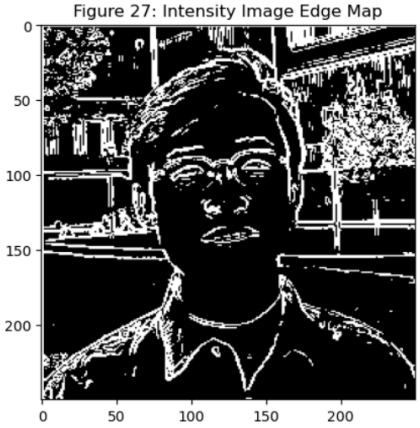


Figure 27: Intensity Image Edge Map

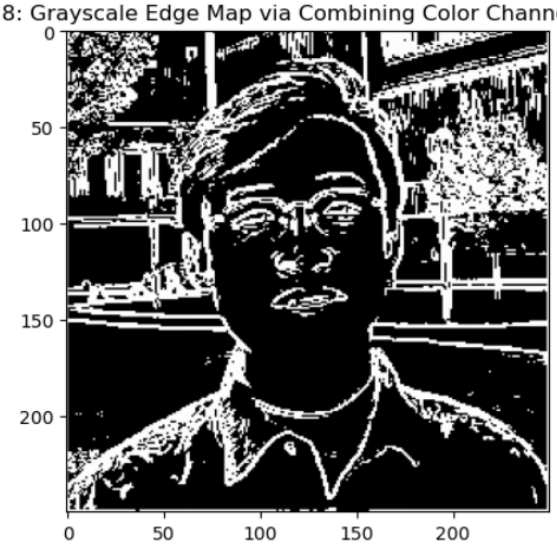


The above edge maps for each colored channel are like the intensity image edge map (seen in Figure 21, and Figure 27 for convenience) because certain colors utilize large components of the red, green, and blue values. For instance, white is produced by setting red = 255, green = 255, and blue = 255 so any edges that are white or white-like like the building or edges of certain parts of the shirt will show in the color channel edge maps. This holds true for black or black-like edges. The color channel edge maps differ from the grayscale one because each channel will emphasize where an edge is predominantly its respective color. For instance, in the red channel edge map, it is observed that there are more edges within the shirt. This is because the shirt is primarily

red, so these edges will be detected with greater detail and intensity in the red channel edge map. Because each channel is sensitive to a specific color, it is useful for focusing on certain objects and extracting them. For instance, if there is an object whose edges are primarily green, it would be best to use the green channel edge map to extract that feature.

The resultant grayscale edge map from combining the three channel edge maps together is seen in Figure 28 below:

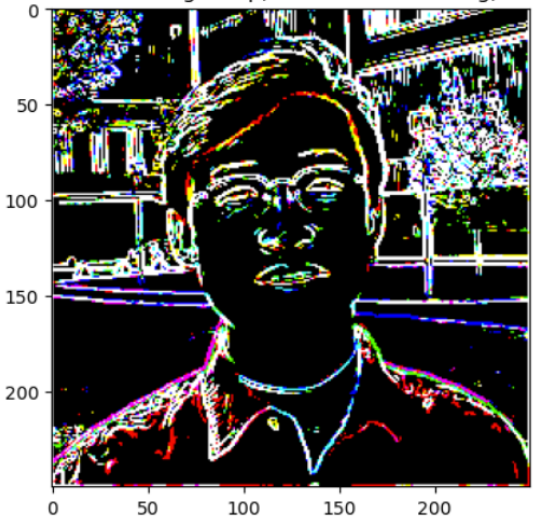
Figure 28: Grayscale Edge Map via Combining Color Channel Edge Maps



Using this approach of producing edge maps of each color channel and then combining them to produce a grayscale edge map resulted in more information being retained than performing edge detection on just the intensity image. For instance, in Figure 28, it is observed that there is a clearer edge for the hairline, shirt color, background path, and facial features. This is because producing an intensity image via simple averaging could result in edges that mostly red, green, or blue could be overshadowed by edges that use all three channels during thresholding. For instance, the hairline is primarily composed of red pixels as seen in Figure 26; but this information is lost in Figure 27 but is retained in Figure 28. As a result, a grayscale edge map generated from combining the edge maps of the three colored channels is a good approach for producing edge maps with more edges retained. Consequently, it likely will be easier to extract features or detect objects within the images due to the additional edges being retained.

Combining the three channel edge maps together will result in a real color edge map, as seen in Figure 27 below:

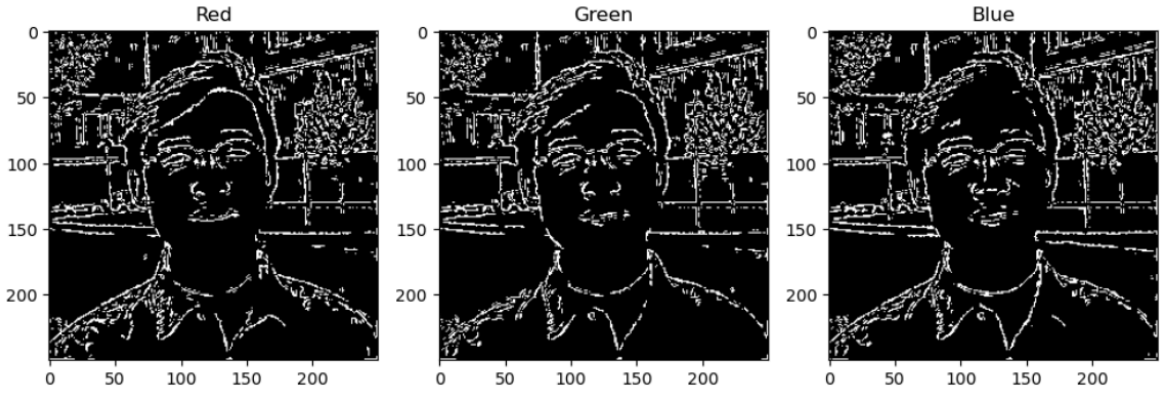
Figure 27: Real Color Edge Map, Global Thresholding; Threshold=20



This edge map is particularly useful for seeing where certain colored edges exist. For instance, if I want to understand where the red edges exist, the white and red edges in the above figure would be the answer. It is observed that there are many red edges located on the shirt and hairline, which matches the aforementioned conclusions from investigating the red channel edge map.

Using adaptive thresholding, like described in Question 3, can also be used for producing edge maps with colored images. This approach results in more retained edge information within each edge map since a local threshold is being calculated within each local window. This is observed when comparing the results of Figure 28 and Figure 26 where there exists additional edge information, particularly within the shirt and face structures.

Figure 28: Edge Maps of RBG Channels, Adaptive Thresholding
Window Size=5,%=.25,Cutoff Intensity=25



Next, we combine the three color channel edge maps into intensity and real-color edge maps as seen in Figure 29 and 30:

Figure 29: Grayscale Edge Map via Combining Color Channel Edge Maps, Adapative Thresholding

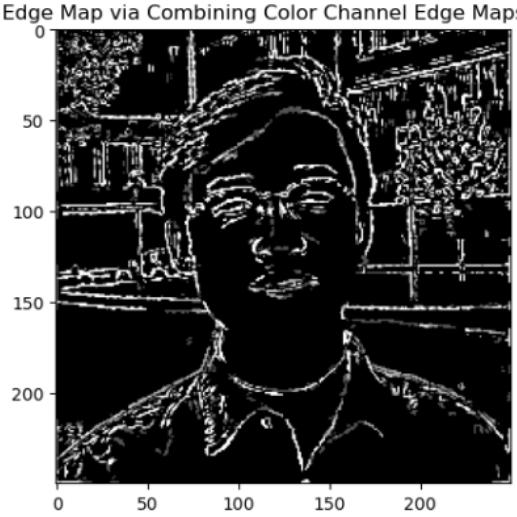
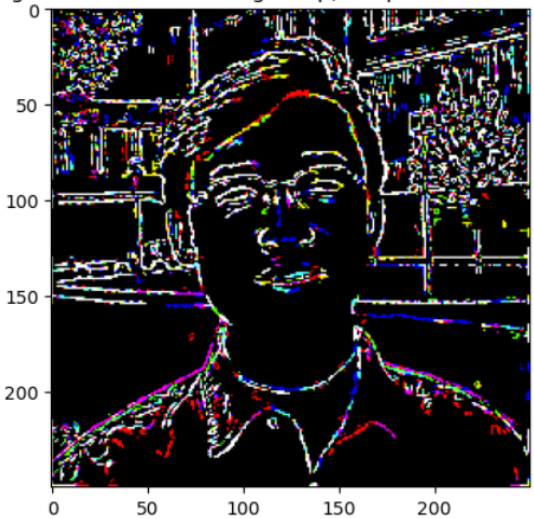


Figure 30: Real Color Edge Map, Adaptive Thresholding



The above results provide the same conclusions and observations as the maps produced via global thresholding, with the added benefit of retaining more edge information since a global threshold is not being applied to all pixels within the image. Additionally, for the real color edge map, retaining the 3 color channel edge information provides more information for training models or drawing conclusions during data exploration/analysis that otherwise may be lost.