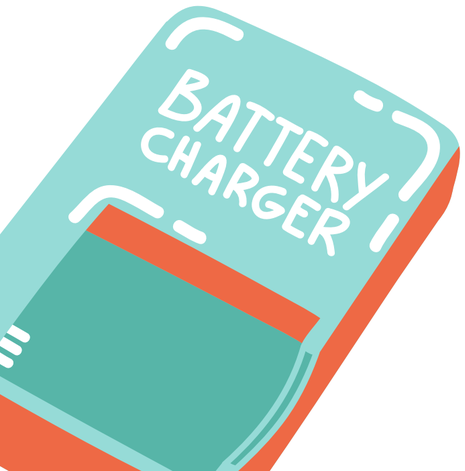
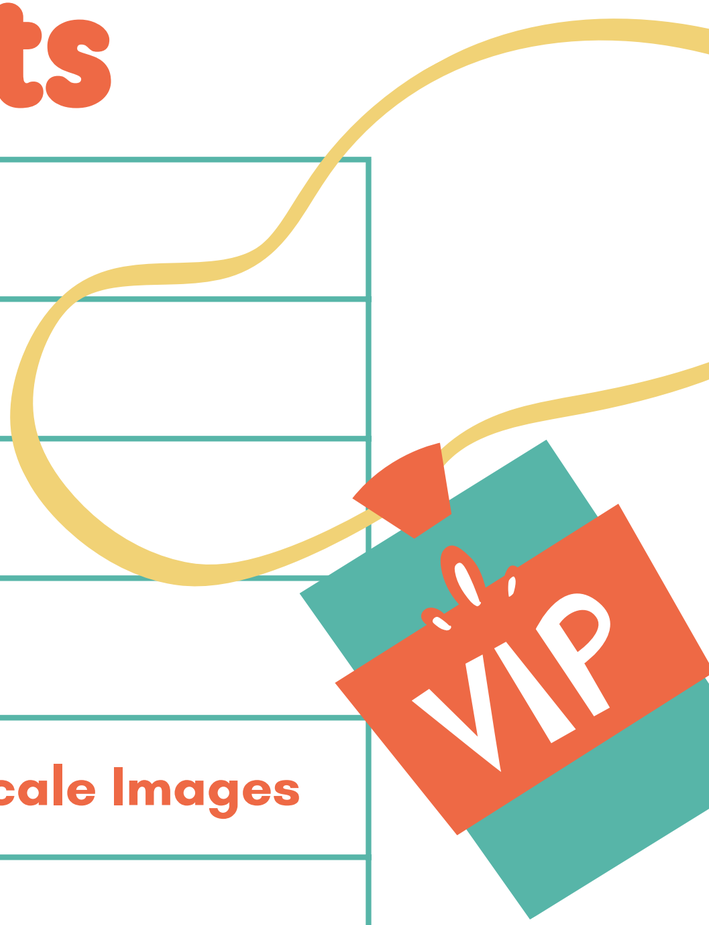
Digital Image Formation and Enhancement

Manalang, Johnenn R.
Applied Physics 157 WFY-FX2

Table of Contents

1	Objectives
2	1.1. Image DIY
7	1.2. Color Image
13	1.3. Altering the Input-Output Curve
17	1.4. Histogram Backprojection on Grayscale Images
22	1.5. Contrast Enhancement
25	1.6. Restoring Faded Color Photographs
31	Reflection





Objectives

01 Mathematically create images.

02 Save an image in an appropriate file format.

03 Open and capture images using Python or Matlab.

04 Improve the appearance of graylevel and color images.

05 Use the backprojection technique to transform the histogram of an image to a desired distribution.

Activity 1.1. Image DIY

Background

The pictures that we see online, which are created using computer graphics, simulation methods, and Artificial Intelligence (AI), are called **synthetic images** [1]. There are several research studies that work with synthetic images, so it is important to be accurate in setting the parameters of these objects when performing simulations [2]. They are used for semantic segmentation [3], machine learning [4], mapping [5], and a lot more applications. In this part, we learned how to create synthetic images using Matlab.

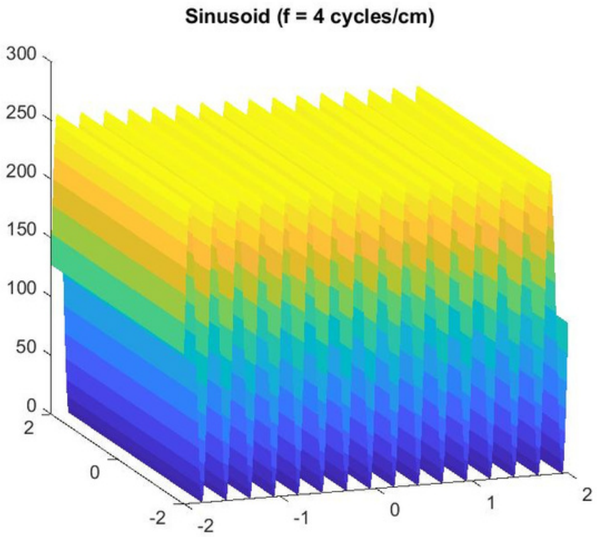
Sinusoid along the x – direction

We are asked to make a sinusoid along the x-direction which has a frequency of 4 cycles/cm, amplitude of 2 and a range from -2cm to 2cm. To mathematically make this sinusoid, I used the equation shown below.

$$y = Asin(\omega t) \quad (1)$$

Here, *A* is the amplitude, ω = 2π*f* where *f* is the frequency, and *t* is time[6]. Since physical images do not have negative values[2], I used the rescale( ) function to change the range of the intensity of the sinusoid between 0 and 255.

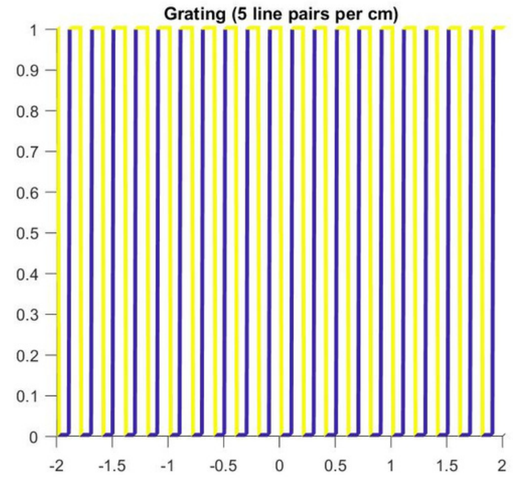
```
%Sinusoid
N = 500;
x = linspace(2,-2,N);
y = x;
[X,~] = meshgrid(x,y);
A = 2;
f = 4;
omega = 2*pi*f;
R = A*sin(omega.*X);
A = rescale(R,0,255);
mesh(x,y,A, 'FaceAlpha', '0.5');
title("Sinusoid (f = 4 cycles/cm)");
view([-19 15]);
grid off;
saveas(gcf, "Sinusoid.tif");
```

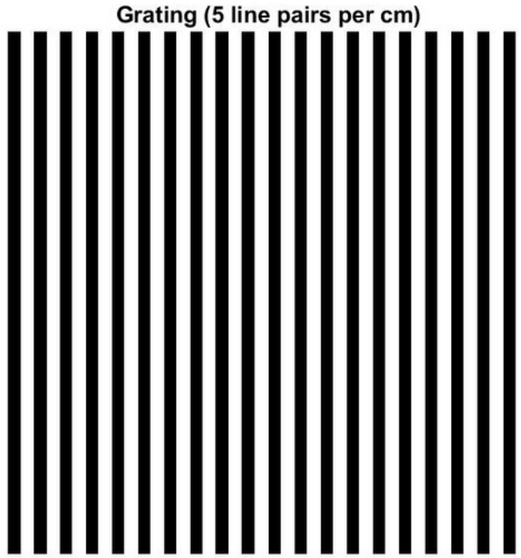


Grating (f = 5 line pairs/cm)

The next task is to make a grating, a lattice or a fixed frame of bars, with frequency of 5 line pairs per cm. Each line pair contains a strip of black and white. Here, I also used Equation (1) to create a sinusoid stored in the variable *R*. Then, I created an array of zeros *A*, the same size as *R* and set the values to be 1 for values of *R* less than 0.1 which is then stored in *A*. This creates a square wave with amplitude of 1. Since the range is from -2cm to 2cm, the image shows 20 line pairs.

```
N = 500;
x = linspace(-2,2,N);
y = x;
[X,~] = meshgrid(x,y);
a = 5;
omega = 2*pi*a;
R = sin(omega.*X);
A = zeros(size(R));
A(R<0.1) = 1;
imshow(A);
title("Grating (5 line pairs per cm)");
saveas(gcf,'grating.tif');
```





Grating is usually used in diffraction simulations. When light encounters a diffraction grating, it produces lines that are very sharp and bright [7].

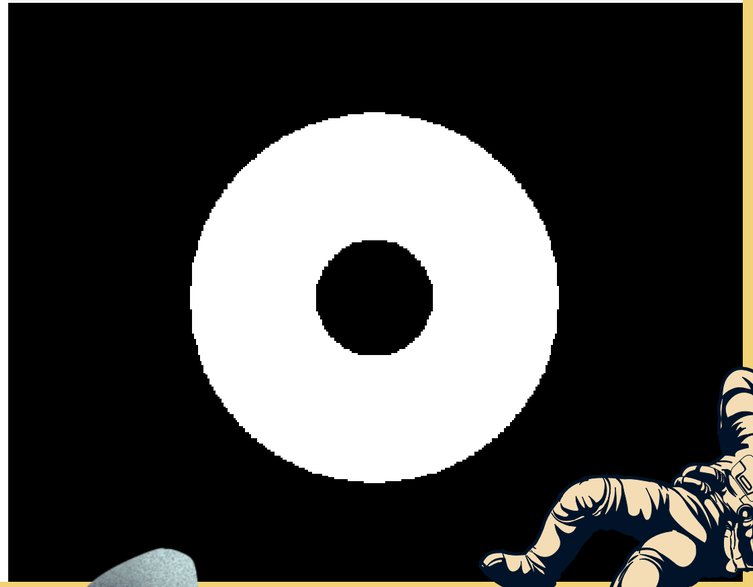
Hubble's Primary Mirror

Hubble's primary mirror is an annulus, a circle with a hole in the middle. To mathematically make this, I used the equation of a circle centered at the origin,

$$r^2 = x^2 + y^2 \qquad (2)$$

where *r* is the radius. I stored this in the variable *R*. To create a hole in the middle, I set the values of *R* less than 0.1 to zero. The image shown here is the cross-section of the annulus.

```
%annulus
N = 500;
x = linspace(-2, 2, N);
y =x;
[X,Y] = meshgrid(x,y);
R = (X.^2 + Y.^2);
A = zeros(size(R));
A(R<1) = 1;
A(R<0.1) = 0;
imshow(A);
title("Hubble's Primary Mirror");
```

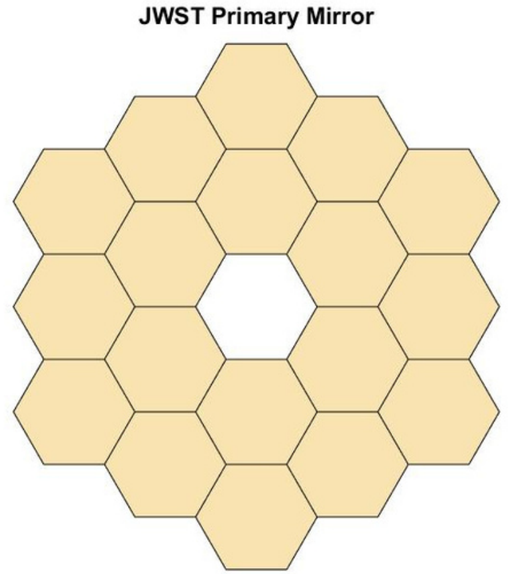


TRIVIA

When the Hubble Space Telescope was launched in 1990, the images it produced were blurred as it cannot bring all the light into a single focus. The problem is due to its primary mirror that is too shallow at the edge by 1/50 the width of a hair.

James Webb Space Telescope Mirrors

The primary mirror of the James Webb Space Telescope is like a honeycomb or a hexagon array [9]. For this part, I used the nsidedpoly( ) function in Matlab that "returns a regular polygon with n equal-length sides. Its name-value arguments include the *Center* to specify the center of the polygon and the *SideLength* to specify the length of each sides. However, there is no argument for color in this function, it chooses a random color for each polygon.



So, I plot all the 18 hexagons and used a for loop to change the *FaceColor* to "#EDB120," which is close to the color of the original gold-coated mirrors.

```
%hexagon array
hex1 = nsidedpoly(6, "Center", [0,0], "SideLength", 1);
hex2 = nsidedpoly(6, "Center", [1.5,0.866025], "SideLength", 1);
hex3 = nsidedpoly(6, "Center", [-1.5,0.866025], "SideLength", 1);
hex4 = nsidedpoly(6, "Center", [-1.5,0.866025+1.73205], "SideLength", 1);
hex5 = nsidedpoly(6, "Center", [1.5,0.866025+1.73205], "SideLength", 1);
hex6 = nsidedpoly(6, "Center", [0,3.4641], "SideLength", 1);
hex7 = nsidedpoly(6, "Center", [3,1.73205], "SideLength", 1);
hex8 = nsidedpoly(6, "Center", [-3,1.73205],"SideLength", 1);
hex9 = nsidedpoly(6, "Center", [-3,1.73205*2],"SideLength", 1);
hex10 = nsidedpoly(6, "Center", [3,1.73205*2],"SideLength", 1);
hex11 = nsidedpoly(6, "Center", [-3,1.73205-0.866025*2],"SideLength", 1);
hex12 = nsidedpoly(6, "Center", [3,1.73205-0.866025*2],"SideLength", 1);
hex13 = nsidedpoly(6, "Center", [1.5,-0.866025], "SideLength", 1);
hex14 = nsidedpoly(6, "Center", [-1.5,-0.866025], "SideLength", 1);
hex15 = nsidedpoly(6, "Center", [-1.5,4.33013], "SideLength", 1);
hex16 = nsidedpoly(6, "Center", [1.5,4.33013], "SideLength", 1);
hex17 = nsidedpoly(6, "Center", [0,5.19616], "SideLength", 1);
hex18 = nsidedpoly(6, "Center", [0,-1.73205], "SideLength", 1);
r = plot([hex1, hex2, hex3, hex4,hex5, hex6, hex7, hex8,hex9, hex10, hex11, hex12, hex13, hex14, hex15, hex16, hex17, hex18]);
for i = 1:1:18
    r(i).FaceColor ="#EDB120";
    axis off;
    axis equal;
    title('JWST Primary Mirror');
end
saveas(gcf, 'hexagon.tif');
```

DID YOU KNOW???

"The Hexagonal shape [of the mirrors] allows for a roughly circular, segmented mirror with high filling factor and six-fold symmetry. High filling factor means the segments fit together without gaps." The mirror is also made of beryllium to make it strong and light (lighter than Hubble's by one-tenth) and the gold coat improves the mirrors' reflection of infrared light [10].

Activity 1.2. Color Image

Background

A person with a trichromatic vision has the ability to match all the colors in the spectrum with three colors: red, green, and blue. The *trichromatic theory of Lomonosov-Young-Helholtz* states the idea that the "retina area of the human eye consists of three primary colors: red, green, and blue." Various colors can then be produced by the fusion of these primary colors, but when all three colors are fused together the result is white [11]. In this activity, we'll learn more about color images and the different image modes and file formats.

Color Images

Here we have an example of a synthetic colored image that consists of three circles colored using the primary colors: red(R), green(G), and blue(B). The circles are plotted such that there are parts where they overlap, like a venn diagram. The image perfectly shows the idea of the *trichromatic theory*. When green and red overlap, the result is yellow. When green and blue overlap, the result is cyan. Moreover, when red and blue overlap, the result is magenta. Yellow(Y), cyan(C), and magenta(M) are the secondary colors. And finally, when all of the circles overlap, the result is white. This is the process of **color addition**.

The secondary colors CMY are the primary colors of pigments. This is because when white light interacts with CMY, the primary colors of light RGB are removed (**color subtraction**).

```
clear; close all;
N = 500;
x = linspace(-10,10,N);
y = x;
[X,Y] = meshgrid(x,y);
Rd = zeros(N,N);
Gn = Rd; Bl = Rd;
%draw colored circles
Rt = 3; RC = 4; deg = 30;
xt = Rt*cosd(deg); yt=Rt*sind(deg);
R = sqrt((X.^2) + (Y+Rt).^2);
Rd(R<RC) = 1;
R = sqrt((X-xt).^2 + (Y-yt).^2);
Gn(R<RC) = 1;
R = sqrt((X+xt).^2 + (Y-yt).^2);
Bl(R<RC) = 1;
I(:,:,1) = Rd;
I(:,:,2) = Gn;
I(:,:,3) = Bl;
figure;image(I);
axis equal;
title('Synthetic Colored Image');
```

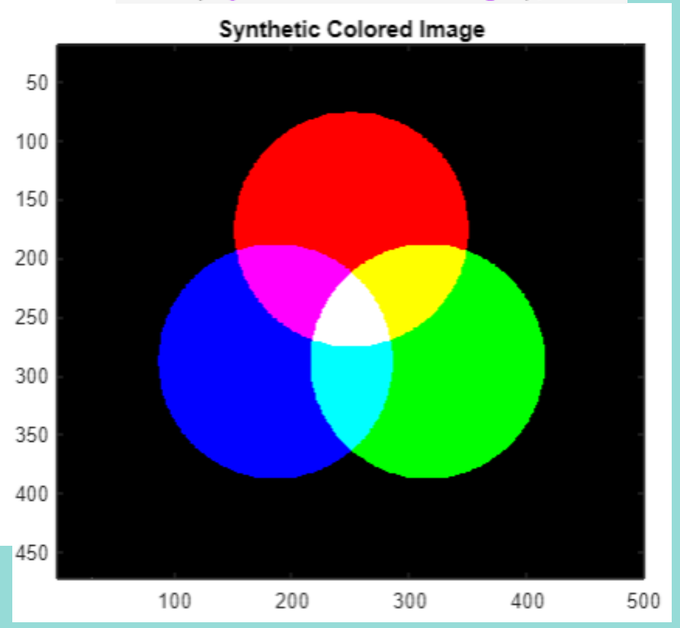


IMAGE MODES

BINARY

There are only 2 possible intensity values for the pixels: 0 for black, 1 or 255 for white. These images are useful for image processing because "they allow easy separation of an object from the background. [13]"

Every form of color information is eliminated in this image, leaving only different shades of gray. Pixel values range from 0 to 255. Grayscale images help in simplifying algorithms and eliminates the complexities related to computational requirements [14].

GRAYSCALE

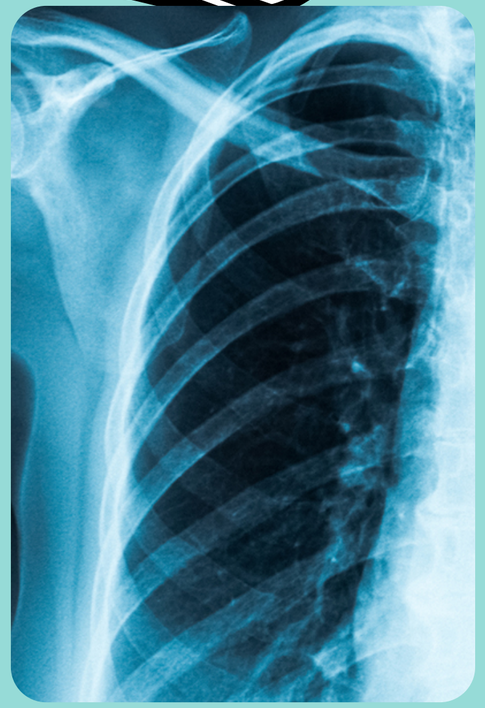
TRUECOLOR

Each pixel is specified by 3 values, one each for red, blue, and green components. The combination of these 3 colors determines the color of each pixel [15]. NASA uses True Color Imagery on satellite images to identify vegetation, water bodies, smogs, clouds, etc [16].

These images consist of a data matrix and a colormap matrix [17]. Unlike truecolor images, changing the colormap of an indexed image affects the display of the image[18]. Indexed images are generally smaller in size[2].

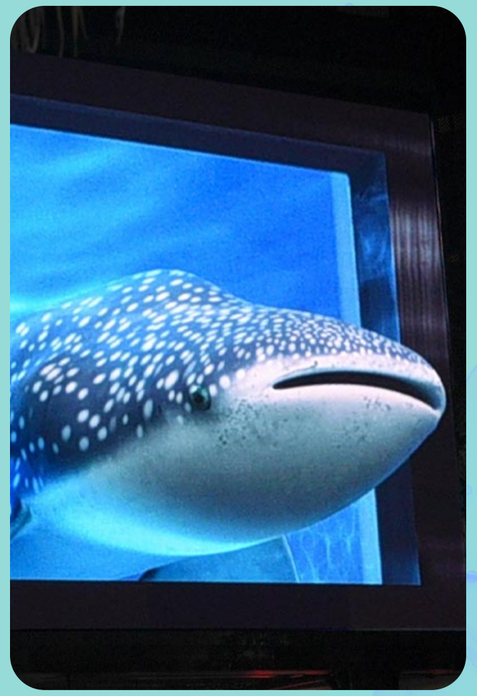
INDEXED

ADVANCE IMAGE TYPES



HIGH DYNAMIC RANGE

HYPERSPECTRAL

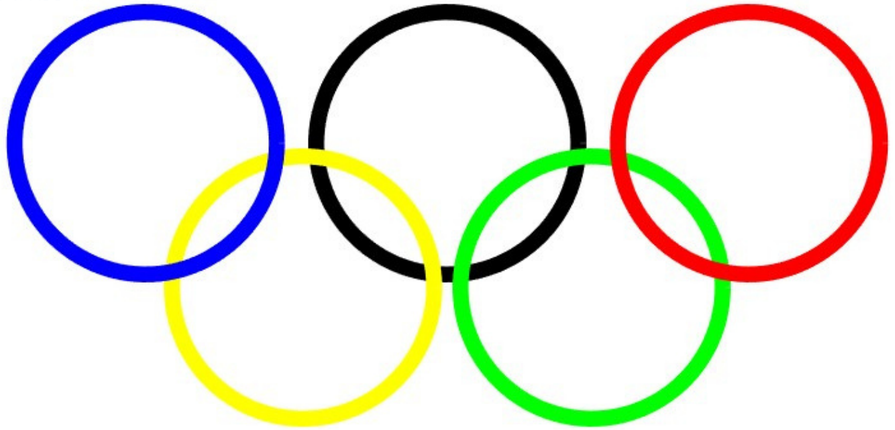


3D

VIDEOS

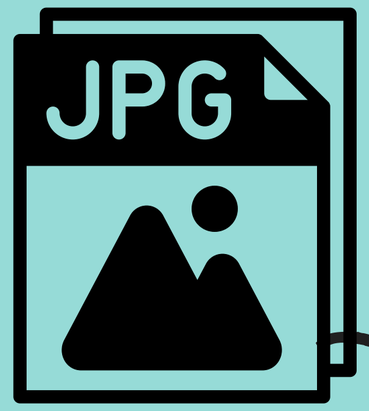
Olympic Logo

An interesting task that I did is to mathematically recreate the Olympics logo as an image. Here, I made 5 circles with the same radius and linewidth. The only difference is their color: black(k), blue(b), red(r), yellow(y), and green (g). To make the circles overlap, I manipulated the equation for their center. The black centered is centered at the origin (0,0) which I used as my reference point. Though the result is not as accurate as the original olympics logo, this one is already a good start.



```
clear
a = linspace(0,2*pi);
%black ring
x = cos(a);
y = sin(a);
%yellow ring
x1 = cos(a) - 1.1;
y1 = sin(a) - 1.1;
%green ring
x2 = cos(a) + 1.1;
y2 = sin(a) - 1.1;
%red ring
x3 = cos(a) + 2.3;
y3 = sin(a);
%blue ring
x4 = cos(a) - 2.3;
y4 = sin(a);
%plotting
plot(x,y,"LineWidth",6, "Color", "k");
hold on;
plot(x1,y1, "LineWidth", 6, "Color", "y");
plot(x2,y2, "LineWidth", 6, "Color", "g");
plot(x3,y3, "LineWidth", 6, "Color", "r");
plot(x4,y4, "LineWidth", 6, "Color", "b");
axis equal;
axis off;
hold off;
```

IMAGE FILE FORMATS

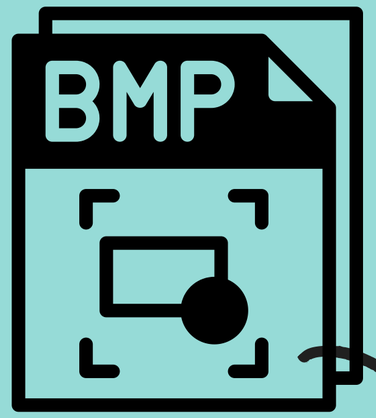


JPEG

Named after the Joint Photographic Experts Group (JPEG) which developed the file format. When an image is saved as a jpeg file, some of the data is lost to reduce the file size. This is the method of **lossy compression.** This file format is not advisable to use in image processing as it results in pixelation around the edges. Use it if you need to save space or share your image online[19].

Unlike JPEG, the Portable Network Graphics (PNG) useless **lossless compression.** This means that saving an image as PNG will not make it lose any of the important details even for a small file size. It is good for saving graphics and supports transparency. If you want your image to be transparent, this is the way to go[19].

PNG

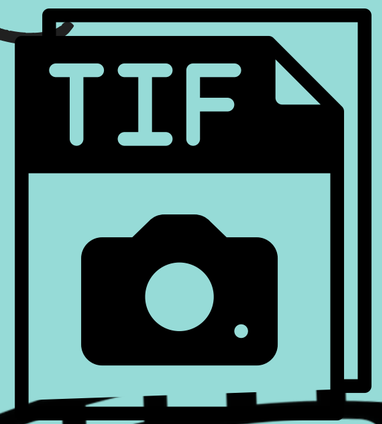


BITMAP

This file format also uses the method of lossless compression. However, unlike PNG which has a smaller file size, the bmp format has a relatively large file size. This is because all the pixels are stored with their respective color values. Therefore, this file format is not advisable to use if you want to send an image through social media sites or email. It will also take up most of your memory's space[20].

Though not advisable to use for displaying images on the web as most web browsers do not support this format, the Tagged Image File Format (TIFF) is significantly useful in image processing. This format supports RGB and CMYK color models and produces high-quality images. However, it can also require more storage space [20].

TIFF



Activity 1.3. Altering the Input-Output Curve

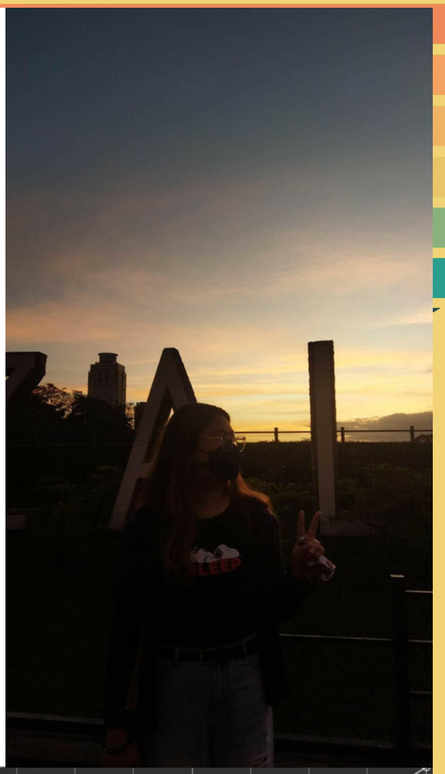
Background

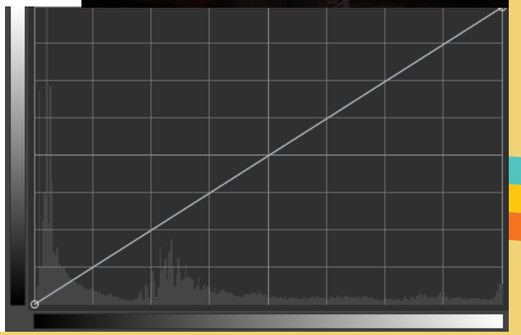
The human eye's 20/20 vision is said to resolve the equivalent of a 52 megapixel camera. However, only the central vision is 20/20 and as you move away from the center, the visual ability decreases. Moreover, at night where there is extremely no light, the human eyes begin to see in monochrome and the central vision begins to register less detail [21]. The human eyes and cameras have different sensitivity to light. Our eyes is said to have a higher sensitivity in dark settings than a typical camera [22]. In this section, we learned about the techniques in enhancing dark-looking images using the software GIMP.

Image Enhancement using GIMP

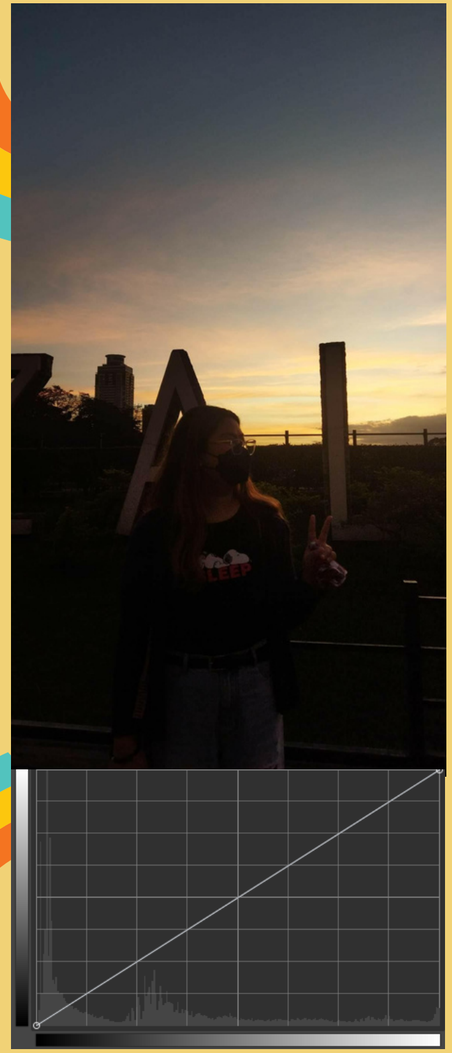
Our task is to pick out a dark image and try to alter its input-output curve using the software GIMP. Here, I selected a photo that my friend took of me when we watched the sunset in Rizal Park. The image appeared dark because the natural source of light, the sun, is slowly fading in the background. Therefore, the details on the lower part of this image is not visible.

I opened the image in GIMP and took a look at its histogram. The histogram of an image is "a plot of grayscale value or digital numbers (DN) versus the number of pixels having those values." A poor contrast image has a skewed histogram towards the brighter or darker graylevels [2]. In this particular image that I used, it is skewed towards the darker graylevels. The goal is for the histogram to be evenly distributed. This means that the image has a good contrast.





ORIGINAL



1

2

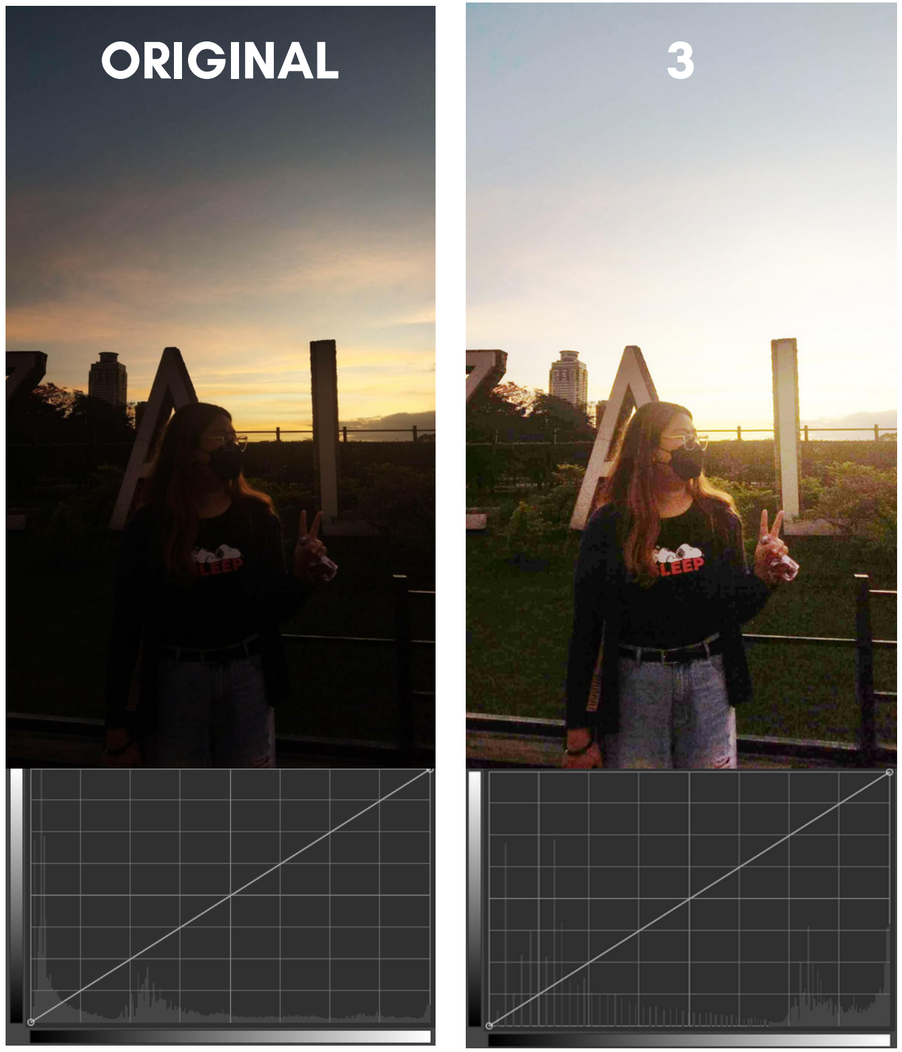
3

Image Enhancement using GIMP

From the previous slide, we saw that as the image gets brighter, its histogram gets distributed evenly.

In the last image, as shown here, the histogram is relatively more even in distribution than the original image. This means that it has a better contrast than the original one. We can also clearly see the details in the lower part of the image: the green grass, the fence, the building, and the big letters behind.

Through this we see that using the right tools, we can enhance the brightness and contrast of an image.



Activity 1.4. Histogram Backprojection on Grayscale Images

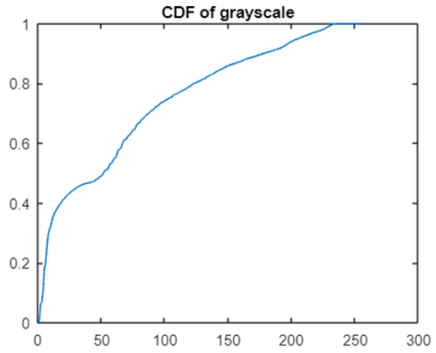
Background

Histogram manipulation is particularly useful in image processing as it can improve the quality of an image, enhance certain image features, and mimic the response of different imaging systems [2]. One way to manipulate the histogram of an image is through the method of **backprojection.** In this section, we will use histogram backprojection to mathematically enhance a grayscale image using linear and nonlinear CDFs.

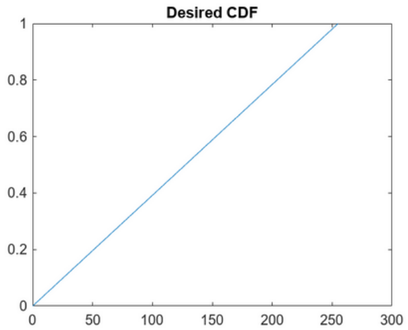
How does Backprojection work?

1 Choose an image that you want to edit. Open it in Matlab and convert to grayscale.

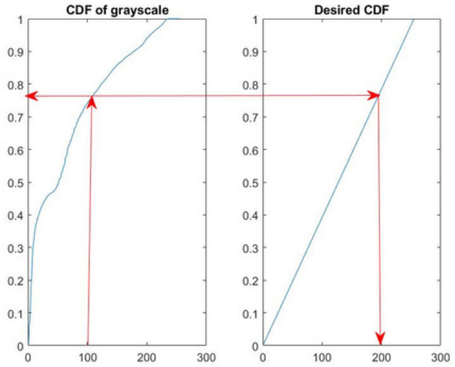
2 Obtain the grayscale historgram of the image and normalize by the number of pixels to get its PDF. Compute the CDF from PDF.



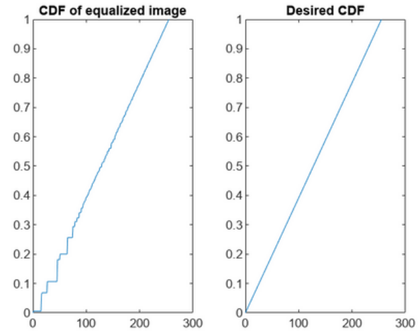
3 Create the desired CDF of the normal distribution with values of x from 0 to 255 and y from 0 to 1.



4 Perform an inverse mapping by using a lookup table and an interpolation routine. Use the syntax: newX = interp1(oldY, oldX, newY).



5 Get the CDF of the equalized image and compare to the desired CDF.



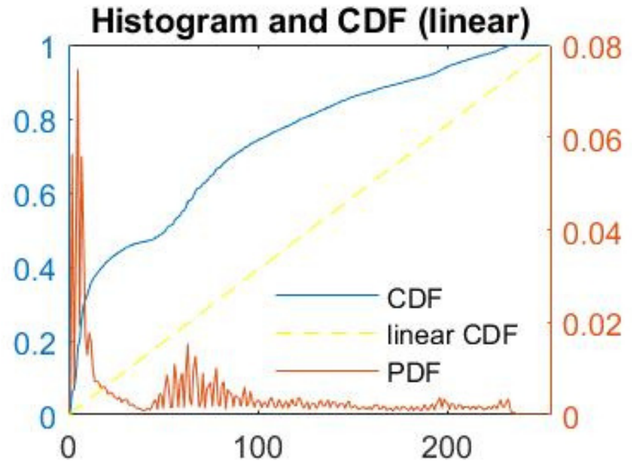
This is the method of **backprojection.** It starts with a grayscale value x which has a corresponding CDF value of the image. It then finds the corresponding y-value in the desired CDF for the given CDF value of the image. Then, it is mapped to the corresponding new x value in the desired CDF. The new CDF of the grayscale image should achieve the desired CDF.

Linear CDF

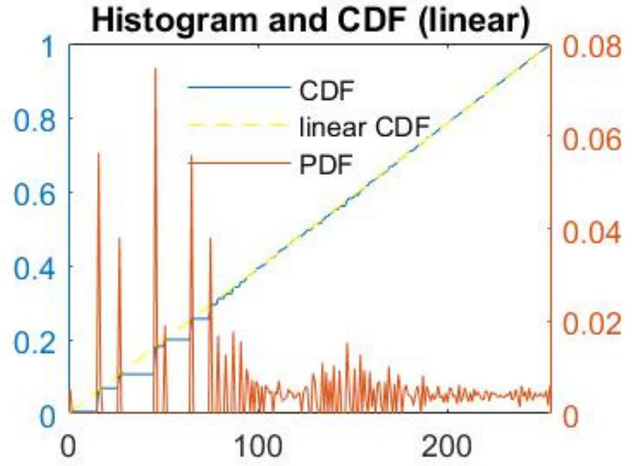
This is the result of the histogram backprojection using a linear CDF. As we can observe, the edited image is relatively brighter than the original grayscale image. Moreover, its histogram is more evenly distributed compared to the original image's skewed histogram to the dark graylevel. It also achieved the desired CDF, though not perfectly.

Hence, the edited image is brighter and has a better contrast than the original.

Original Image



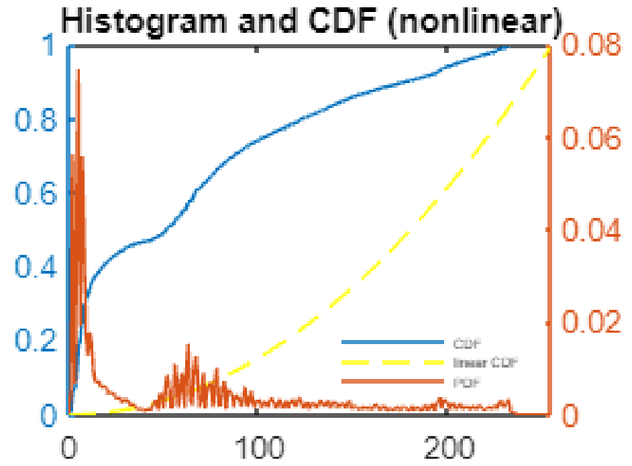
Edited Image



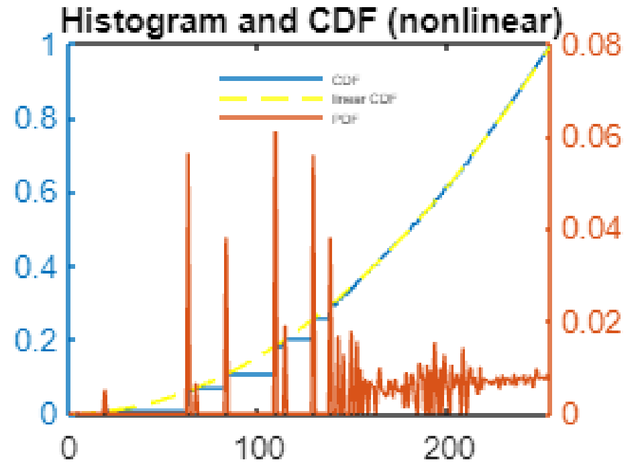
Nonlinear CDF

I tried using a nonlinear CDF, this is like a half of a parabola curved up. The resulting image is brighter. Its histogram is now distributed, but is leaning to the bright graylevel. The CDF of the edited image also achieved the desired CDF.

Original Image



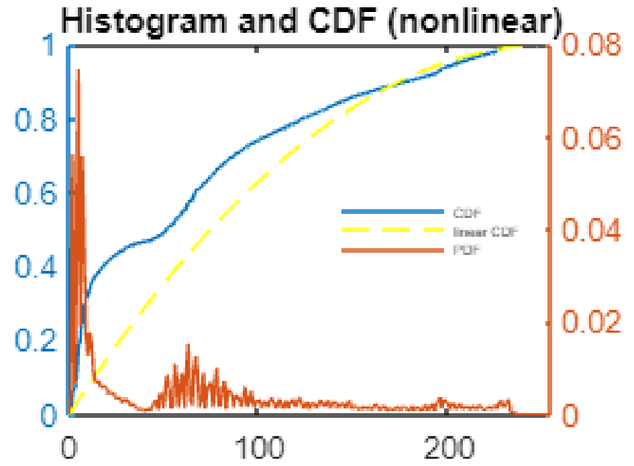
Edited Image



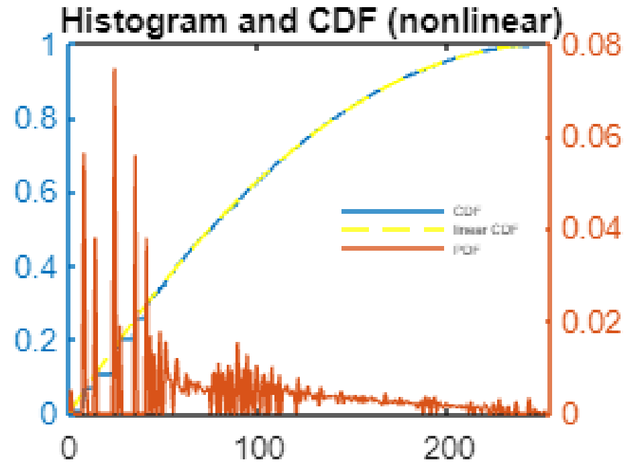
Nonlinear CDF

This is another nonlinear CDF that I used. This time, the resulting image appears to be somewhat brighter, but is relatively darker than the first two resulting images from the previous slides. Its histogram is more distributed than the original, however, it is still leaning to the dark grayscale. Nevertheless, it achieved the desired CDF.

Original Image



Edited Image



Activity 1.5. Contrast Enhancement

Background

Contrast is one of the important factors in image evaluation and processing. Contrast is the difference between the tones, colors, and texture of an image [23-24]. Another type of image enhancing technique is **contrast stretching.** It tries to improve the contrast of an image by stretching its intensity values to fill the desired range. Unlike other methods, this only uses a linear transformation function [24]. In this section, we learned how to enhance the image through contrast stretching.

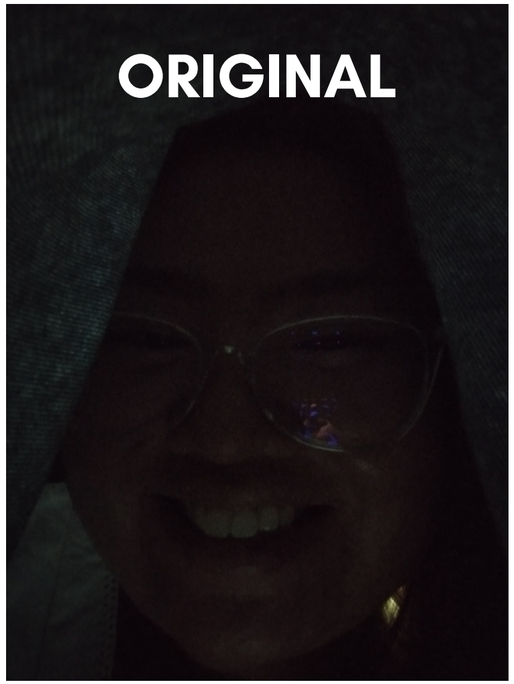
Contrast Stretching

For this task, I took a selfie under my jacket and used that image for the simulation. Here is the original image that I used. It is obvious that some details are not visible in this image.

Similar with the previous section, I also turned this image into grayscale and computed for its PDF and CDF. Then, I used the equation

$$I_{new} = \frac{I_{old} - I_{min}}{I_{max} - I_{min}} \quad (3)$$

But instead of just getting the minimum and maximum value of the of the array, I selected the min and max values at different percentiles. I used the 10th, 50th, and 90th percentile.

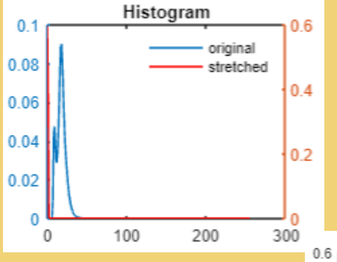


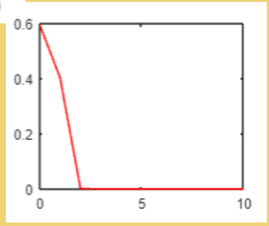
Original Image

These are the resulting images after contrast stretching. As we can observe, as we use a higher percentile, the image becomes darker (higher contrast). If we compare the histogram of the original and stretched images, we can see that the histogram is also "stretched" after applying the method.

10th

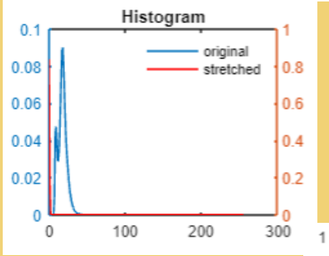
Stretched Image

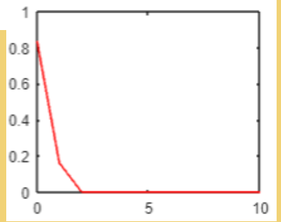




50th

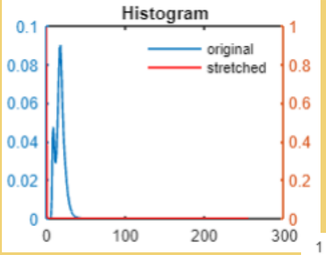
Stretched Image

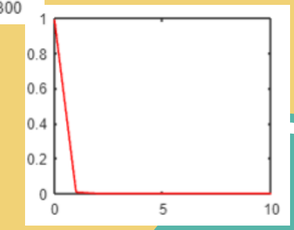




90th

Stretched Image





Activity 1.6. Restoring Faded Color Photographs

Background

In camera settings, you can select the "White Balance (WB)," which allows you to select white balancing constants appropriate for the capturing conditions or setting. In image processing, there are 3 popular White Balancing Algorithms namely **contrast stretching**, **white patch algorithm, and gray world algorithm.** These algorithms can be used to restore faded color photographs or correct unbalanced images [2]. In this section, we learned how to use the White Balancing Algorithms in restoring faded or unbalanced photographs, and described how these algorithms differ.

The "Cute" Unbalanced Image

For this section, we are asked to choose a faded or unbalanced image from our collection. So I dug in my "baul" and I saw this throwback picture of a 2-year old Johnenn showing her then only 5 sets of teeth to the camera. This is an example of an unbalanced image because my skin looks redish and my white and pink striped clothes and my white socks do not look white. I used this image and applied the 3 White Balancing Algorithms.

Contrast Stretching in RGB

Unlike the method of contrast stretching in grayscale image, contrast stretching in RGB requires separating the red, green, and blue channels of the image. As we can see, the RGB channels appear to be gray. This is because an RGB image, the original one, has 3 channels while the separated R, G, and B channels now only has 1 channel each, which results to a grayscale image. The regions which appear to be lighter in the channels contribute more to the color that they represent, while the regions that contribute less are darker [26].

Each of these channels are stretched using the method of contrast stretching.

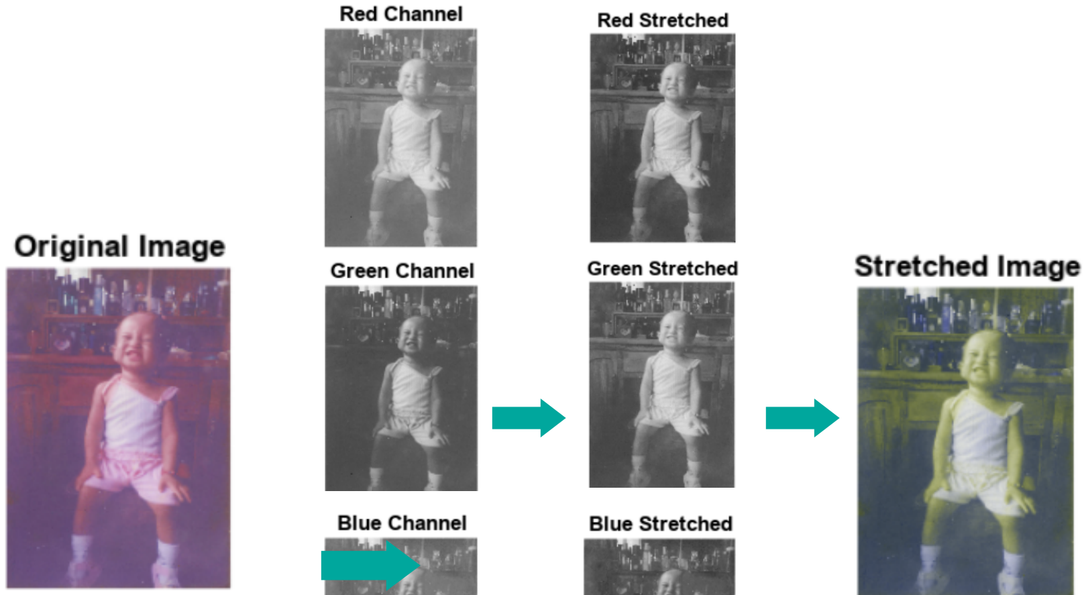
Original Image

Red Channel

Green Channel

Blue Channel

Contrast Stretching in RGB



After performing contrast stretching in each of the RGB channels, they are combined again to form the stretched image.

As we can see, the stretched image has a different appearance. The skin looks more realistic in color and the white regions appear to be brighter.

Gray World Algorithm

In the gray world algorithm, it is assumed that the average color of the world is gray. The way to perform this is to get the average of the RGB channels of an image. Then, to get the white balanced image, the RGB channels will be divided by their respective averages [2].

Original Image

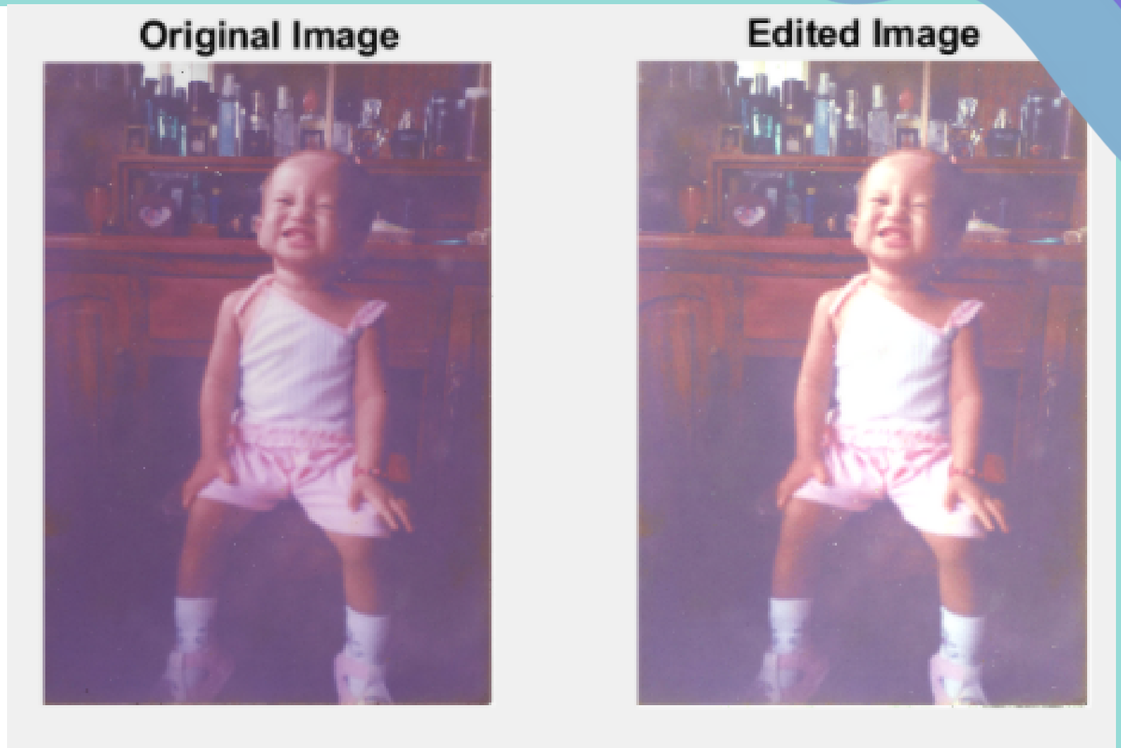
Edited Image

Edited Image (contrast=0.5)

Here, the resulting image has become so bright that some of the details are not visible already. So, I multiplied 0.5 to increase the contrast of the resulting image. This problem arises when the original image used has a dominant color patch [27]. In the original image that I used, the dominant color patch is red. But by increasing the contrast of the resulting image, we are left with a well-balanced image.

White Patch Algorithm

In the White Patch Algorithm, instead of getting the average of the RGB channels of the whole image, what we only need is the average of the RGB channels of the white regions. Then, the RGB channels are divided by the respective white averages from each channel [2]. To choose the particular white region, I used the imcrop( ) function from the Image Processing Toolbox in Matlab.



Here we can see the comparison between the original and resulting image. The white regions in the original image now appears to be white in the resulting image. This is how the white patch algorithm works, it locates the regions of the objects that are truly white by assuming that these regions are the brightest [28].

Reflection

It is my first time using Matlab in this activity. As a Python user, from AP 155, it is not difficult to learn how to use Matlab. The activity is really interesting because the results are not just some random numbers, they're images! And it's already visible whether your code was correct or you did something wrong along the process because the resulting image is not what you expect to see. What I particularly enjoyed the most in this activity is the white balancing algorithms. I get to use my old faded pictures and enhance them using these algorithms. We have a lot of old pictures stored in our house that I can't wait to use.

One of the challenges that I encountered in this activity is when the resulting image from the contrast stretching part always comes out binary; the array values are just 0 and 1. So with the help of my classmates, we tried to debug the code and we finally figured out what to do. Team work always work!

Overall, I enjoyed the whole activity. It's nice to play around with images and color channels. Looking forward to the next activities!

References

1. What is synthetic data? Synthetic Images. (2023). Retrieved March 8, 2023, from https://synthetic-images.com/synthetic-data/
2. Soriano, M. (2023). Digital Image Information and Enhancement.
3. Li, G., Kang, G., Liu, W., Wei, Y., Yang, Y. (2020). Content-Consistent Matching for Domain Adaptive Semantic Segmentation. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, JM. (eds) Computer Vision – ECCV 2020. ECCV 2020. Lecture Notes in Computer Science(), vol 12359. Springer, Cham. https://doi.org/10.1007/978-3-030-58568-6_26
4. Xiong, R. and Tang, P. (2021), "Machine learning using synthetic images for detecting dust emissions on construction sites", Smart and Sustainable Built Environment, Vol. 10 No. 3, pp. 487-503. https://doi.org/10.1108/SASBE-04-2021-0066
5. Xue, H., Shah, S., Greiser, A., Guetter, C., Littmann, A., Jolly, M.-P., ... Kellman, P. (2011). Motion correction for myocardial T1 mapping using image registration with synthetic image estimation. Magnetic Resonance in Medicine, 67(6), 1644–1655. doi:10.1002/mrm.23153
6. Bayin, Ş.S. (2020) Essentials of mathematical methods in science and engineering. Hoboken: Wiley.
7. Loewen, E.G. and Popov, E. (1997) Diffraction gratings and applications. New York: M. Dekker.
8. Pickering, E. C. (1881). Large telescopes. The Rosen Publishing Group, Inc.
9. Chimileski, S., Kolter, R., & Schaechter, M. (2017). Life at the edge of sight: A photographic exploration of the Microbial World. The Belknap Press of Harvard University Press.
10. NASA. (n.d.). Web's Mirrors. NASA. Retrieved March 8, 2023, from https://webb.nasa.gov/content/observatory/ote/mirrors/index.html#:~:text=meters%20in%20diameter

References

11. Rajamanickam, M. (2002). Modern psychophysical and scaling methods and experimentation. Concept Pub. Co.
12. The three little pigments. Exploratorium. (2020, October 2). Retrieved March 8, 2023, from https://www.exploratorium.edu/snacks/three-little-pigments#:~:text=Red%2C%20green%2C%20and%20blue%20are,cyan%2C%20magenta%2C%20and%20yellow
13. Vlerick, A. (2020, June 12). A visual introduction to binary image processing (part 1). Medium. Retrieved March 8, 2023, from https://towardsdatascience.com/a-visual-introduction-to-binary-image-processing-part-1-d2fba9f102a4
14. Isahit. (2022, August 4). Why to use grayscale conversion during image processing? Isahit. Retrieved March 8, 2023, from https://www.isahit.com/blog/why-to-use-grayscale-conversion-during-image-processing#:~:text=Why%20is%20grayscale%20needed%20for,to%20its%20barest%20minimum%20pixel
15. Matlab. (n.d.). True Color Images. Image Processing Toolbox User's Guide. Retrieved March 8, 2023, from http://matlab.izmiran.ru/help/toolbox/images/intro7.html#:~:text=A%20truecolor%20image%2C%20also%20known,components%20for%20each%20individual%20pixel
16. NASA. (n.d.). Sport viewer. NASA. Retrieved March 8, 2023, from https://weather.ndc.nasa.gov/sport/viewer/
17. Indexed Images. Introduction (image processing toolbox). (n.d.). Retrieved March 8, 2023, from http://www.ece.northwestern.edu/local-apps/matlabhelp/toolbox/images/intro5.html#:~:text=An%20indexed%20image%20uses%20direct,as%20an%20index%20into%20map%20

References

18. Eddins, S. (2016, February 22). Matlab image display – truecolor and indexed images. Steve on Image Processing with MATLAB. Retrieved March 8, 2023, from https://blogs.mathworks.com/steve/2016/02/22/matlab-image-display-truecolor-and-indexed-images/?doing_wp_cron=1678287285.9702069759368896484375
19. Lee, J. (2018). JPEG, PNG or GIF? what these image formats mean and when to use them. Young Post. Retrieved March 8, 2023, from https://www.scmp.com/yp/discover/entertainment/tech-gaming/article/3073568/jpeg-png-or-gif-what-these-image-formats-mean
20. Image file formats: An overview of 10 image file types. IONOS Digital Guide. (n.d.). Retrieved March 8, 2023, from https://www.ionos.com/digitalguide/websites/web-design/image-file-format-types/
21. Cameras vs. the human eye. Cameras vs. The Human Eye. (n.d.). Retrieved March 8, 2023, from https://www.cambridgeincolour.com/tutorials/cameras-vs-human-eye.htm#:~:text=2.-,RESOLUTION%20%26%20DETAIL,60%C2%B0%20angle%20of%20view
22. Pro's Choice. (2016, August 24). Using a night vision camera as a security camera. Pro's Choice. Retrieved March 8, 2023, from https://www.proschoice.com.au/blogs/main/using-a-night-vision-camera-as-a-security-camera#:~:text=Night%20vision%20cameras%20have%20infrared,help%20of%20these%20infrared%20lights.
23. Anbarjafari, G. (n.d.). 5. image enhancement: Contrast enhancement, part I. University of Tartu. Retrieved March 8, 2023, from https://sisu.ut.ee/imageprocessing/book/5#:~:text=Contrast%20is%20an%20important%20factor,other%20objects%20and%20the%20background

References

24. Understanding contract in photography. Skylum Blog. (n.d.). Retrieved March 8, 2023, from https://skylum.com/blog/understanding-contrast-in-photography
25. Atul, K. (2019, April 19). Contrast stretching. TheAILearner. Retrieved March 8, 2023, from https://theailearner.com/2019/01/30/contrast-stretching/
26. Maximinusjoshus. (2021, June 8). Understanding the concept of channels in an image. Medium. Retrieved March 8, 2023, from https://medium.com/featurepreneur/understanding-the-concept-of-channels-in-an-image-6d59d4dafaa9
27. Li, X., Wu, J. (2013). Improved Gray World Algorithm Based on Salient Detection. In: Tan, T., Ruan, Q., Chen, X., Ma, H., Wang, L. (eds) Advances in Image and Graphics Technologies. IGTA 2013. Communications in Computer and Information Science, vol 363. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-37149-3_38
28. Cepeda-Negrete, J., & Sanchez-Yanez, R. E. (2012). Experiments on the white patch Retinex in RGB and CIELAB color spaces. Academia. Retrieved March 8, 2023, from https://www.aacademica.org/jcepedanegrete/2.pdf

REPORT GRADE

Criteria	Score
Technical Correctness	35
Quality of Presentation	35
Self Reflection	30
Initiative	10
TOTAL	100

EVALUATION

Overall, I know that I have accomplished all the tasks included in this activity. Moreover, I did some extra experiments and made some comparisons in the resulting images. I also put a lot of effort in creating this presentation and cited a lot of sources for additional information in the report.