[1]

Image Processing

SC 205 - Discrete Math Project

Raj Shah - 202201403
Ramya Shah - 202201409
Bhoomish Patel - 202201414
Manthan Parmar - 202201416
Vats Shah - 202201417
Rakshit Pandhi - 202201426

Contents

1 Introduction

- Discrete mathematics is a branch of mathematics that deals with mathematical structures and objects that are fundamentally discrete and not continuous.
- It focuses on studying mathematical concepts and techniques that are primarily used in computer science, information theory, cryptography, and other areas of computer-related fields.
- Unlike continuous mathematics, which deals with concepts like calculus and real numbers, discrete mathematics deals with finite or countable sets, such as integers, graphs, and logical statements.
- It provides tools and methods for analyzing and solving problems in a precise and rigorous manner.
- Key topics in discrete mathematics include set theory, logic, combinatorics, graph theory, number theory, and discrete probability.
- Set theory establishes the foundation for much of discrete mathematics by defining basic operations on sets, such as union, intersection, and complement.
- Overall, discrete mathematics provides a formal framework for analyzing and solving problems in various fields by utilizing mathematical structures and techniques tailored to discrete and countable objects.
- Its applications span computer science, cryptography, information theory, operations research, and many other disciplines.
- In our project, we are going to be exploring and implementing an efficient application of 'Image Processing' by applying discrete mathematics.

2 Application

- We chose the topic of ”Image Processing” in the field of discrete mathematics for our project due to its strong practical significance and real-world applications. Image processing techniques, such as Blurring, Sharpening, Grayscale conversion, and RGB manipulation, play a vital role in domains including photography, computer vision, and digital media.
- The motivation behind our choice lies in the widespread demand for efficient image manipulation techniques. By leveraging the principles of discrete mathematics, we can develop optimized algorithms that enhance image quality, achieve desired visual effects, and provide precise control over image appearance.
- Additionally, the project combines the realms of discrete mathematics and image processing, demonstrating the powerful synergy between mathematical concepts and practical applications. By exploring the integration of discrete mathematics in image processing, we can unlock new possibilities for enhancing digital imagery.
- In conclusion, the practical importance of image processing methods and the chance to use discrete mathematics' strengths to improve these operations serve as our driving forces. We hope to contribute to the creation of improved image processing algorithms through this initiative that have practical uses.

3 Motivation

Our project aims to leverage discrete mathematics to address challenges in image processing, specifically Blurring, Sharpening, Grayscale conversion, and RGB conversion. In today's image-driven world, efficient and accurate manipulation of visual content is essential. By incorporating discrete mathematics principles, we strive to develop advanced algorithms that enhance the efficiency, accuracy, and effectiveness of these operations. Our mission is to provide practical solutions that empower professionals in various domains, such as photography, social media, and computer vision, to achieve superior image processing results, meeting the growing demand for high-quality visual content in today's digital landscape.

4 Formulating the Math

4.1 Theory

4.1.1 What is an Image?

- In simple terms, an image refers to a visual representation or picture of something. It can be a photograph, painting, drawing, or even a digital graphic. Images are often used to convey information, capture moments, or express ideas and emotions.
- In mathematics, an image can be represented as a huge M x N matrix which stores information regarding colors of a particular position.
- A high number represents a bright color that is near to white pixels in an image can also be said to have weight. The darker the color the lower the weight and the lighter the color the greater the weight.

4.1.2 What is Image Processing?

- Image processing is a technique for applying various procedures to an image to improve it or extract some relevant information from it.
- It is a kind of signal processing where the input is an image and the output can either be another image or features or characteristics related to that image.
- Image processing is one of the technologies that is currently expanding quickly it is a primary subject of research in both engineering and computer science fields.

4.1.3 What is a Kernel?

- In image processing, a kernel refers to a small matrix that is used for various operations such as filtering, convolutions, and feature extraction.
- A kernel is typically a square matrix with odd dimensions (e.g., 3x3, 5x5, etc.) that defines a specific operation to be performed on an image. Each element of the kernel holds a weight or a coefficient that determines the contribution of neighboring pixels to the output result.

4.2 Image Sharpening

- Image sharpening is a technique used to enhance the clarity and detail of an image by emphasizing the high-frequency components. There are various methods and algorithms available for image sharpening.
- We have used a technique known as ”Laplacian Sharpening”/”Laplacian Enhancement” to enhance the image.

  Following are the steps used in the technique:
- Split Color Channels: Split the color image into its individual color channels (Red, Green, and Blue) to access each channel separately.
- Convolution: Convolution is the process of adding each element of the image to its local neighbors, weighed by a kernel.
- For example, if we have two three-by-three matrices, the first a kernel, and the second an image piece, convolution is the process of flipping both the rows and columns of the kernel and multiplying locally similar entries and summing.
- The element at coordinates [2, 2] (that is, the central element) of the resulting image would be a weighted combination of all the entries of the image matrix, with weights given by the kernel:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} [2,2] = (i.1)+(h.2)+(g.3)+(f.4)+(e.5)+(d.6)+(c.7)+(b.8)+(a.9)$$

- Applying Laplacian Filter: The Laplacian operator can be given by:

$$\nabla^2 f = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y}$$

- Partial derivative in x direction can be given as:

$$\frac{\partial^2 f}{\partial^2 x} = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

- Similarly, partial derivative in y direction can be given as:

$$\frac{\partial^2 f}{\partial^2 x} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

- And combining these equations for partial derivatives, we can obtain:

$$\frac{\partial^2 f}{\partial^2 x} = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

- Now Lets consider what a filter looks like 3*3 filter for a image can be represented for co-ordinate (x,y) can be represented as:

$$\begin{bmatrix} x-1, y-1 & x, y-1 & x+1, y-1 \\ x-1, y & x, y & x, y+1 \\ x-1, y+1 & x, y+1 & x+1, y+1 \end{bmatrix}$$

- If we put the coefficients of the equation in the image we can obtain following matrix:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

- Mask of Laplacian + addition:

$$g(x) = f(x) - \nabla^2 f(x)$$

(when center coefficient of Laplacian mask is negative)

$$g(x) = f(x) + \nabla^2 f(x)$$

(when center coefficient of Laplacian mask is positive)

- Resultant kernel with coefficients:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

- Adjust the Sharpening Strength: Multiply each Laplacian-filtered image (for each color channel) by a constant factor to control the strength of the sharpening effect. Experiment with different values to achieve the desired level of sharpening.

- Combine Adjusted Color Channels: Add the adjusted Laplacian-filtered images (for each color channel) back together to obtain the sharpened color image. Combine the respective color channel values for each pixel to reconstruct the color image.

- Clip the Values: After combining the color channels, clip the pixel values to ensure they remain within the valid range (0 to 255 for each channel).

- Save the Sharpened Image: Once you're satisfied with the sharpening effect, save the sharpened color image as a separate file to preserve the original image.

4.3 Image Blurring

- Gaussian blur is a popular image blurring technique used in various image processing tasks. It applies a Gaussian filter to an image, which helps reduce noise and smooth out fine details. The blur effect is achieved by convolving the image with a Gaussian kernel, resulting in weighted averaging of pixel values.

- Gaussian Distribution: The Gaussian distribution, also known as the normal distribution, is a mathematical function that describes a bell-shaped curve. It is characterized by a mean (center) and a standard deviation (spread). The standard deviation determines the amount of blurring applied to the image. A larger standard deviation results in more extensive blurring.

- Gaussian Kernel: A Gaussian kernel is a matrix that represents the shape of the Gaussian distribution. It is generated based on the desired standard deviation. The kernel is usually square and has an odd size, such as 3x3, 5x5, or 7x7, to have a defined center.

We have used a 5*5 Gaussian kernel:

$$\frac{1}{256} * \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

- Convolution: It involves overlaying the Gaussian kernel onto each pixel of the image and calculating the weighted average of the pixel values.
- Weighted Averaging: During convolution, each pixel in the image is multiplied by the corresponding value in the Gaussian kernel and then summed up to obtain the new value for that pixel.
  The values in the Gaussian kernel represent weights, where pixels closer to the center of the kernel have higher weights, while those farther away have lower weights. This weighted averaging process smooths out the image by reducing the influence of noisy or high-frequency details.
- Border Handling: When applying Gaussian blur to the edges of an image, there might not be enough neighboring pixels to perform the convolution. Various methods can be used to handle this situation, such as padding the image with zeros or extending the image by replicating its border pixels.
- Repeated Convolution: To achieve stronger blurring, the Gaussian blur operation can be repeated multiple times. Each iteration applies the Gaussian filter again to the already blurred image. This cumulative effect results in increased blurring.

4.4 Image to Grayscale

- For converting an image to grayscale involves transforming a colored image into a grayscale representation, where each pixel's intensity value represents the level of brightness or grayness.
- The theory behind this method is based on the perception of human vision and the properties of color images.
- Color images are typically represented in the RGB (Red, Green, Blue) color space. Each pixel is composed of three color channels: red, green, and blue.
- Each color's intensity contributes to the overall color appearance of the image. In grayscale images, there is only one channel representing the level of brightness.
- We have used the method of weighted average of the RGB values. Since human vision is more sensitive to green light, the green channel usually has the highest weight, followed by the red channel and then the blue channel. The weights used in the conversion can vary, but a common formula is:
- Grayscale value = (0.2989 * Red) + (0.5870 * Green) + (0.1140 * Blue)
- By applying this formula to each pixel in the image, the RGB values are combined to produce a single grayscale value.
- When displaying or printing the grayscale image, each pixel's intensity value is mapped to a corresponding shade of gray, ranging from black (minimum intensity) to white (maximum intensity).
- This grayscale representation simplifies the image and can be useful in various applications, such as image analysis, computer vision, and printing.

4.5 Image to Monochromatic

- In digital images, colors are typically represented using the RGB (Red, Green, Blue) color model.
- Red, along with green and blue, is one of the primary colors in this model.
- Each color channel in RGB has a range of 0 to 255, where 0 signifies no intensity and 255 denotes the highest intensity.
- Red can be isolated by setting the green and blue channels to 0, resulting in a monochromatic image with shades of red.
- Similarly, green can be isolated by setting the red and blue channels to 0, and blue can be isolated by setting the red and green channels to 0.
- This process creates monochromatic images with varying intensities of the respective colors.

4.5.1 Monochromatic Red

- To convert an image to monochromatic red, you would need to set the green and blue color channels to 0 while keeping the red channel unchanged.
- This process effectively removes all colors except for shades of red, resulting in a monochromatic image with various intensities of red.
- In other words, the resulting image will be grayscale but with different shades of red instead of gray.
- Monochromatic Red value = (1 * Red) + (0 * Green) + (0 * Blue).

4.5.2 Monochromatic Green

- Similarly to the red channel, to convert an image to monochromatic green, you would set the red and blue color channels to 0 while retaining the green channel.
- This operation isolates the green color information and removes all other colors from the image.
- The resulting monochromatic image will consist of different shades of green, creating a grayscale appearance.
- Monochromatic Green value = (0 * Red) + (1 * Green) + (0 * Blue).

4.5.3 Monochromatic Blue

- Converting an image to monochromatic blue involves setting the red and green color channels to 0 while keeping the blue channel intact.
- By eliminating the red and green components, the resulting image will only contain shades of blue.
- This monochromatic representation appears grayscale but with varying intensities of blue instead of gray.
- Monochromatic Blue value = (0 * Red) + (0 * Green) + (1 * Blue)

5 Solving the Math

5.1 Image Sharpening

- To sharpen an image, we use a sample 6x6 matrix to explain how the process works :

$$\begin{bmatrix} 7 & 6 & 5 & 5 & 6 & 7 \\ 6 & 4 & 3 & 3 & 4 & 6 \\ 5 & 3 & 2 & 2 & 3 & 5 \\ 5 & 3 & 2 & 2 & 3 & 5 \\ 6 & 4 & 3 & 3 & 4 & 6 \\ 7 & 6 & 5 & 5 & 6 & 7 \end{bmatrix}$$

- By multiplying with kernel obtained previously in theory :

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

- For example the top left most element '7', we do the following math:

$$\begin{aligned} \text{Sharpened value} = & \\ & (0*7)+(-1*7)+(0*6) \\ & +(-1*7)+(5*7)+(-1*6) \\ & +(0*6)+(-1*6)+(0*4) \\ = & \ 9 \end{aligned}$$

- Following this step for every element of matrix we get the following sharpened matrix:

$$\begin{bmatrix} 9 & 8 & 6 & 6 & 8 & 9 \\ 8 & 2 & 1 & 1 & 2 & 8 \\ 6 & 1 & 0 & 0 & 1 & 6 \\ 6 & 1 & 0 & 0 & 1 & 6 \\ 8 & 2 & 1 & 1 & 2 & 8 \\ 9 & 8 & 6 & 6 & 8 & 9 \end{bmatrix}$$

5.2 Image Blurring

- To blur an image, we use a sample 6x6 matrix to explain how the process works :

$$\begin{bmatrix} 4 & 3 & 3 & 4 \\ 3 & 2 & 2 & 3 \\ 3 & 2 & 2 & 3 \\ 4 & 3 & 3 & 4 \end{bmatrix}$$

- By multiplying with kernel obtained previously in theory :We have used a 5*5 Gaussian kernel:

$$\frac{1}{256} * \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

- For example the top left most element '4', we do the following math:

  Blurred value =

$$\begin{aligned} &[(4*1)+(4*4)+(4*6)+(3*4)+(3*1)\\ &+(4*4)+(4*16)+(4*24)+(3*16)+(3*4)\\ &+(4*6)+(4*24)+(4*36)+(3*24)+(3*6)\\ &+(3*4)+(3*16)+(3*24)+(2*16)+(2*4)\\ &+(3*1)+(3*4)+(3*6)+(2*4)+(2*1)]/256 \quad = 3.375 \end{aligned}$$

- Following this step for every element of matrix we get the following blurred matrix:

$$\begin{bmatrix} 3.375 & 3.0625 & 3.0625 & 3.375 \\ 3.0625 & 2.75 & 2.75 & 3.0625 \\ 3.0625 & 2.75 & 2.75 & 3.0625 \\ 3.375 & 3.0625 & 3.0625 & 3.375 \end{bmatrix}$$

5.3 Image to Grayscale

- To convert an image to grayscale, we use a sample 4x4 matrix to explain how the process works:

$$\begin{bmatrix} [255,0,0] & [0,255,0] & [0,0,255] & [255,255,0] \\ [0,255,255] & [255,0,255] & [128,128,128] & [0,0,0] \\ [255,255,255] & [128,0,0] & [0,128,0] & [0,0,128] \\ [255,128,0] & [128,255,0] & [0,128,255] & [255,0,128] \end{bmatrix}$$

- We apply the formula to get a grayscale value :

  Grayscale value = [(1 * Red) + (1 * Green) +(1 * Blue)]/3

- For example we take the top left element [255,0,0], we do the following math :

  Grayscale value = [(1 * 255) + (1 * 0) +(1 * 0)]/3 = 85

- Following this step for every element of matrix we get the following grayscale matrix:

$$\begin{bmatrix} 85 & 85 & 85 & 170 \\ 170 & 170 & 128 & 0 \\ 255 & 42.67 & 42.67 & 42.67 \\ 127.67 & 127.67 & 127.67 & 127.67 \end{bmatrix}$$

5.4 Image to Monochromatic Red, Green and Blue

- To convert an image to monochrome, we use a sample 4x4 matrix to explain how the process works:

$$\begin{bmatrix} [255,0,0] & [0,255,0] & [0,0,255] & [255,255,0] \\ [0,255,255] & [255,0,255] & [128,128,128] & [0,0,0] \\ [255,255,255] & [128,0,0] & [0,128,0] & [0,0,128] \\ [255,128,0] & [128,255,0] & [0,128,255] & [255,0,128] \end{bmatrix}$$

- We apply the following formulae to get monochromatic element values :

  Red value = (1 * Red) + (0 * Green) +(0 * Blue)

  Green value = (0 * Red) + (1 * Green) +(0 * Blue)

  Blue value = (0 * Red) + (0 * Green) +(1 * Blue)

- For example we take the top left element [255,0,0], we do the following math :

- In the case of monochromatic red Monochromatic red value =(1 * 255) + (0 * 0) +(0 * 0) = 255

- Following this step for every element of matrix we get the following monochromatic red matrix:

$$\begin{bmatrix} 255 & 0 & 0 & 255 \\ 0 & 255 & 128 & 0 \\ 255 & 128 & 0 & 0 \\ 255 & 128 & 0 & 255 \end{bmatrix}$$

- For example we take the top left element [255,0,0], we do the following math :

- In the case of monochromatic green Monochromatic green value =(0 * 255) + (1 * 0) +(0 * 0) = 0

- Following this step for every element of matrix we get the following monochromatic green matrix:

$$\begin{bmatrix} 0 & 255 & 0 & 255 \\ 255 & 0 & 128 & 0 \\ 255 & 0 & 128 & 0 \\ 128 & 255 & 128 & 0 \end{bmatrix}$$

- For example we take the top left element [255,0,0], we do the following math :

- In the case of monochromatic blue Monochromatic blue value =(0 * 255) + (0 * 0) +(1 * 0) = 255

- Following this step for every element of matrix we get the following monochromatic blue matrix:

$$\begin{bmatrix} 0 & 0 & 255 & 0 \\ 255 & 255 & 128 & 0 \\ 255 & 0 & 0 & 128 \\ 0 & 0 & 255 & 128 \end{bmatrix}$$

6 Implementation

6.1 Image Sharpening

- Following is the code used in applying image sharpening. This converts image into a sharpened version of itself.

```
let src = cv.imread(imgElement);
let dst = new cv.Mat();
//Matrix for image sharpening
let M1 = cv.matFromArray(3, 3, cv.CV_32S, [0,-1, 0,-1, 5, -1,
    0, -1, 0]);
let anchor1 = new cv.Point(-1, -1);
cv.filter2D(src, dst3, cv.CV_8U, M1, anchor1, 0, cv.
   BORDER_DEFAULT);
cv.imshow('sharpOutput', dst);
src.delete();
dst.delete();
```

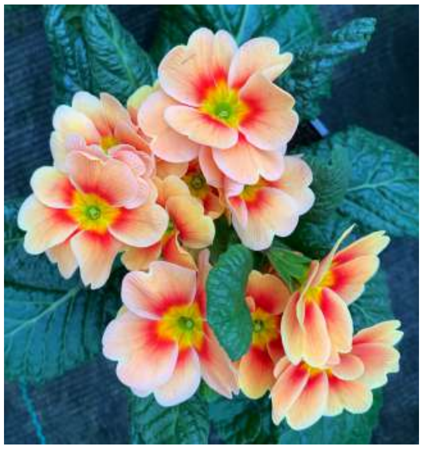

Figure 1: Original Image

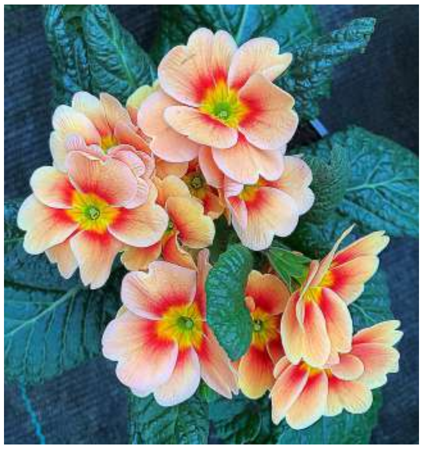

Figure 2: Sharpened Image

6.2 Image Blurring

- Following is the code used in applying image blurring. This converts image into a blurred version of itself.

```
let src = cv.imread(imgElement);
let dst = new cv.Mat();
//Matrix for blurring
let M = cv.matFromArray(5, 5, cv.CV_32F, [1/256, 4/256,
    6/256, 4/256, 1/256, 4/256, 16/256, 24/256, 16/256, 4/256,
     6/256, 24/256, 36/256, 24/256, 6/256, 4/256, 16/256,
    24/256, 16/256, 4/256, 1/256, 4/256, 6/256, 4/256, 1/256])
    ;
let anchor = new cv.Point(-1, -1);
cv.filter2D(src, dst, cv.CV_8U, M, anchor, 0, cv.
    BORDER_DEFAULT);
cv.imshow('blurOutput', dst);
src.delete();
dst.delete();
```

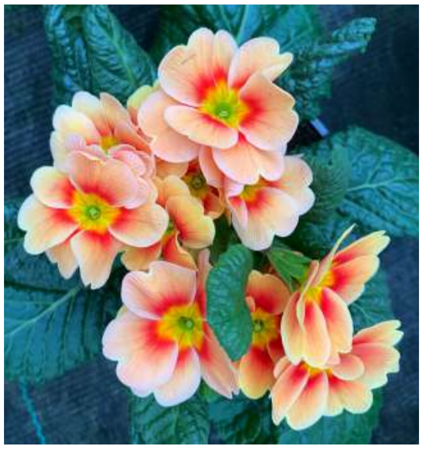
Figure 3: Original Image

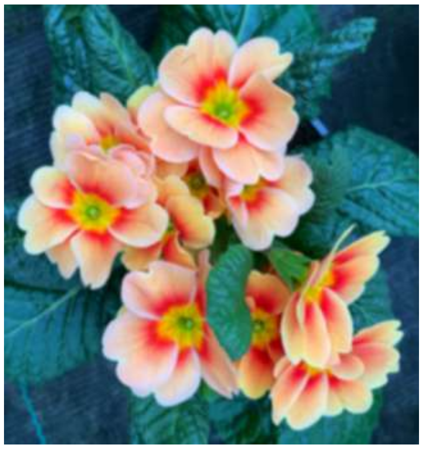
Figure 4: Blurred Image

6.3 Image to Grayscale

- Following is the code used in applying image to Grayscale convertor. This converts image into a grayscale version of itself.

```
const canvas = document.getElementById('blur-content');
const ctx = canvas.getContext('2d');
const img = imgElement;canvas.width = img.width;canvas.height
     = img.height;
ctx.drawImage(img, 0, 0, 420, 393);
const imageData = ctx.getImageData(0, 0, 420, 393);
const data = imageData.data;
for (let i = 0; i < data.length; i += 4) {
     const red = data[i];     const green = data[i + 1];
    const blue = data[i + 2];
     const grayscale = Math.round(0.2989 * red + 0.587 * green
     + 0.114 * blue);
     data[i] = grayscale;    data[i + 1] = grayscale;    data[
    i + 2] = grayscale; }
```

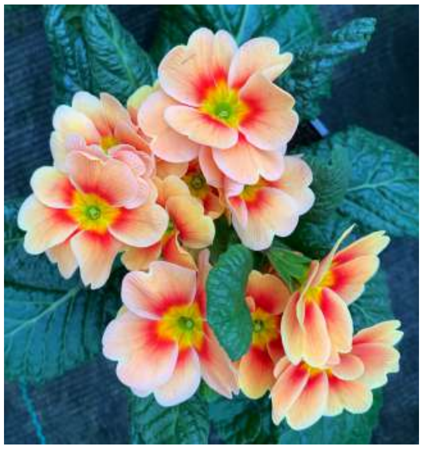
Figure 5: Original Image

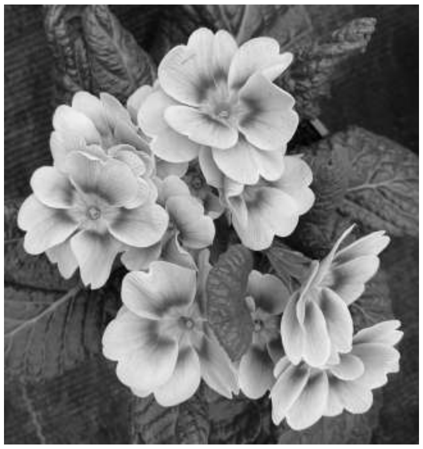
Figure 6: grayscale Image

6.4 Image to Monochromatic

6.4.1 Monochromatic Red

- Following is the code for a monochromatic red image convertor. This converts image into a monochromatic red version of itself.

```
const canvas = document.getElementById('blur-content');
const ctx = canvas.getContext('2d');
const img = imgElement;
canvas.width = img.width;
canvas.height = img.height;
ctx.drawImage(img, 0, 0,420,393);
const imageData = ctx.getImageData(0, 0, 420, 393);
const data = imageData.data;
for (let i = 0; i < data.length; i += 4) {
    data[i + 1] = 0; // green channel
    data[i + 2] = 0; // blue channel
}
ctx.putImageData(imageData, 0, 0);
```

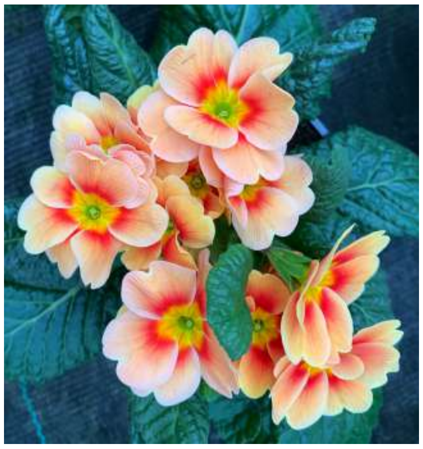
Figure 7: Original Image

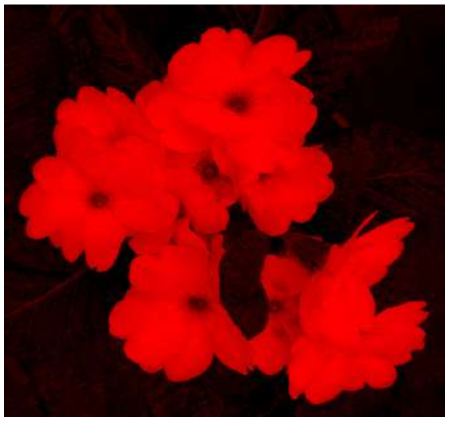
Figure 8: Monochromatic Red Image

6.4.2 Monochromatic Green

- Following is the code used in applying image to monochromatic green convertor. This converts image into a monochromatic green version of itself.

```
const canvas = document.getElementById('blur-content');
const ctx = canvas.getContext('2d');
const img = imgElement;
canvas.width = img.width;
canvas.height = img.height;
ctx.drawImage(img, 0, 0,420,393);
const imageData = ctx.getImageData(0, 0, 420, 393);
const data = imageData.data;
for (let i = 0; i < data.length; i += 4) {
    data[i] = 0; // red channel
    data[i + 2] = 0; // blue channel
}
ctx.putImageData(imageData, 0, 0);
```

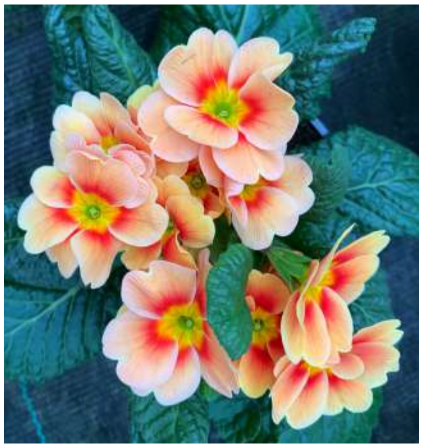
Figure 9: Original Image

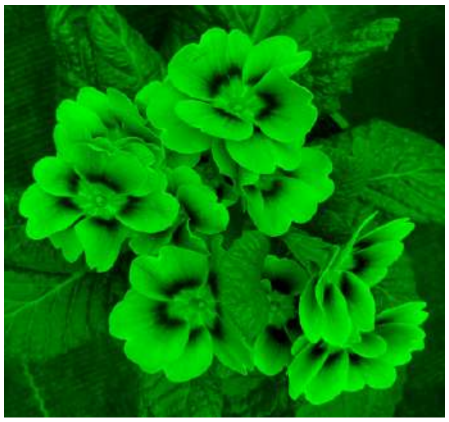
Figure 10: Monochromatic Green Image

6.4.3 Monochromatic Blue

- Following is the code used in applying image to monochromatic blue convertor. This converts image into a monochromatic blue version of itself.

```
const canvas = document.getElementById('blur-content');
const ctx = canvas.getContext('2d');
const img = imgElement;
canvas.width = img.width;
canvas.height = img.height;
ctx.drawImage(img, 0, 0,420,393);
const imageData = ctx.getImageData(0, 0, 420, 393);
const data = imageData.data;
for (let i = 0; i < data.length; i += 4) {
    data[i ] = 0; // red channel
    data[i + 1 ] = 0; // green channel
}
ctx.putImageData(imageData, 0, 0);
```

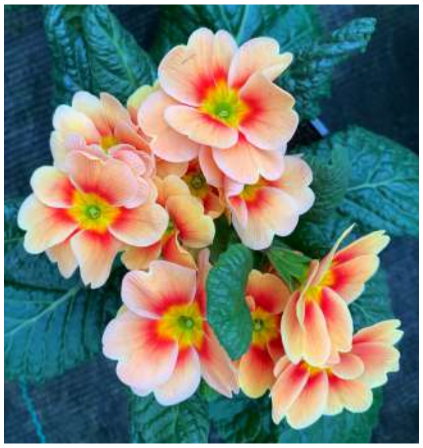

Figure 11: Original Image

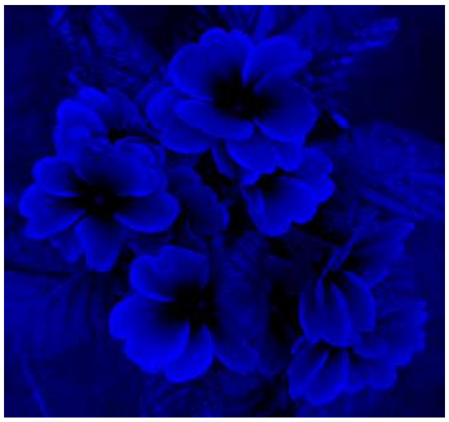

Figure 12: Monochromatic Blue Image

7 Commercialization

- This strategy, combined with further refinements and more polishing can be used to construct a website or a fully-fledged application that can be used to address various market needs and cater to different industries.
- One potential avenue is to target the photography industry, where professionals and photo enthusiasts constantly are seeking ways to tinker with images by manipulate and enhance them.
- By making our solution provide more high-quality sharpening, advanced blurring and precise grayscale and RGB conversions for accurate color reproduction, we can enter a market where we stand out as a better solution than the rest of the crowd.
- By offering an intuitive and easy to use user interface, compatibility with all of most popular image file formats, and the ability to process images quickly and efficiently, our project can attract a wide range of costumers eager to get hands on to use the solution for their own.

8 Contribution

- Raj Shah - 202201403 (16.67%) :-
  PowerPoint presentation, Image processing code
- Ramya Shah - 202201409 (16.67%) :-
  Website Designing, Video.
- Bhoomish Patel - 202201414 (16.67%) :-
  Coding implementation, google sites.
- Manthan Parmar - 202201416 (16.67%) :-
  LaTeX, Video editing.
- Vats Shah - 202201417 (16.67%) :-
  LaTeX, Info collection.
- Rakshit Pandhi - 202201426 (16.67%) :-
  Google sites, Image processing code.

9 Learning Outcome

- Throughout this project, we gained valuable insights into image processing techniques and their mathematical foundations. By working with HTML, CSS, and JavaScript, we successfully implemented various algorithms, including image sharpening, blurring, grayscale conversion, and RGB processing.
- We would also like to thank our professors, Prof. Manish K. Gupta, Prof. Prosenjit Kundu and Prof. Manoj Raut for providing us with such an opportunity to be able to work on real life applications of discrete mathematics.
- One of the key lessons we learned was the importance of discrete mathematics in image processing. The understanding and application of mathematical concepts, such as convolution, filters, and color spaces, proved crucial in achieving the desired results. We realized how discrete mathematics provides a solid framework for solving complex problems in image manipulation.
- Moreover, collaboration played a significant role in our project's success. By working together as a team, we shared ideas, troubleshooted issues, and leveraged each other's strengths. This collaborative environment fostered effective communication and allowed us to learn from one another's perspectives and experiences.
- In conclusiom, this project provided us with a practical hands-on experience in applying discrete mathematics to image processing. We developed problem-solving and colloborative skills, which are invaluable in the field on image processing and beyond.
- Overall this has been an enriching learning experience, equipping us with a solid foundation in discrete mathematics, image processing and real world problem solving.

References

[1] Dhirubhai Ambani Institute of Information and Communication Technology Logo. https://ecampus.daiict.ac.in/webapp/intranet/index.jsp .

[2] Roger N. Clark. Exploring the mnist digits dataset. https://dyclassroom.com/image-processing-project/how-to-convert-a-color-image-into-red-green-blue-image .

[3] Roger N. Clark. Image Sharpening intro. https://clarkvision.com/articles/image-sharpening-intro/ .

[4] Greg Cope. Photography image processing kernel. https://www.naturefocused.com/articles/photography-image-processing-kernel.html .

[5] R.C. Gonzalez and R.E. Woods. *Digital Image Processing*. Pearson, 2018.

[6] Demi Lindy. Pink and White Flower on Gray Textile . https://unsplash.com/photos/RBIQqvkeRwc .

[7] MathWorks. RGB to Gray. https://www.mathworks.com/help/matlab/ref/rgb2gray.html.

[8] W.K. Pratt. *Introduction to Digital Image Processing*. CRC Press, 2013.

[9] Simplilearn. Image Processing Article. https://www.simplilearn.com/image-processing-article .

[10] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision*. Cengage Learning, 2014.

[11] Tutorialspoint. Concept of blurring. https://www.tutorialspoint.com/dip/concept_of_blurring.htm .

[12] Andrew Zola. What is definition of Image. https://www.techtarget.com/whatis/definition/image .