Image Processing using concepts of Linear Algebra (Transformation Matrix) and Convolution

Group-06 (Machine Design)

By :

- **Bharat Anant (224103410)**
- **Digboloy Borah (224103412)**
- **Himanshu Lahare (224103414)**
- **Rishabh Saluja (224103426)**
- **Vidisha Singh (224103438)**

Course Instructors :

- Dr. Sajan Kapil
- Dr. Nelson Muthu

Department of Mechanical Engineering
Indian Institute of Technology Guwahati

Introduction

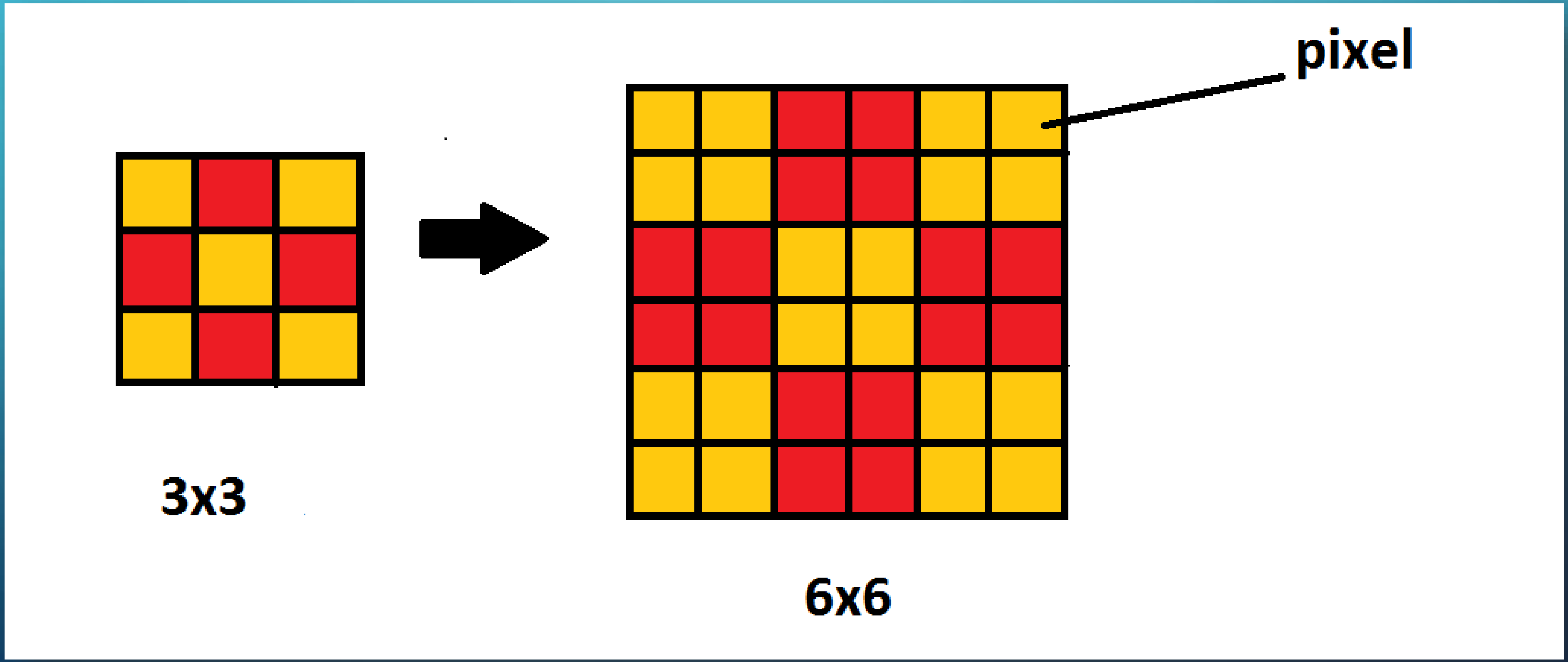
- In this project we will learn how to manipulate images in order to obtain different colour effects.

- We will use matrix multiplication in order to manipulate image colour and achieve different effects.

- We will look at several different ways to manipulate the image colours to obtain effects similar to those of ***Instagram*** and other ***Graphic Softwares***.

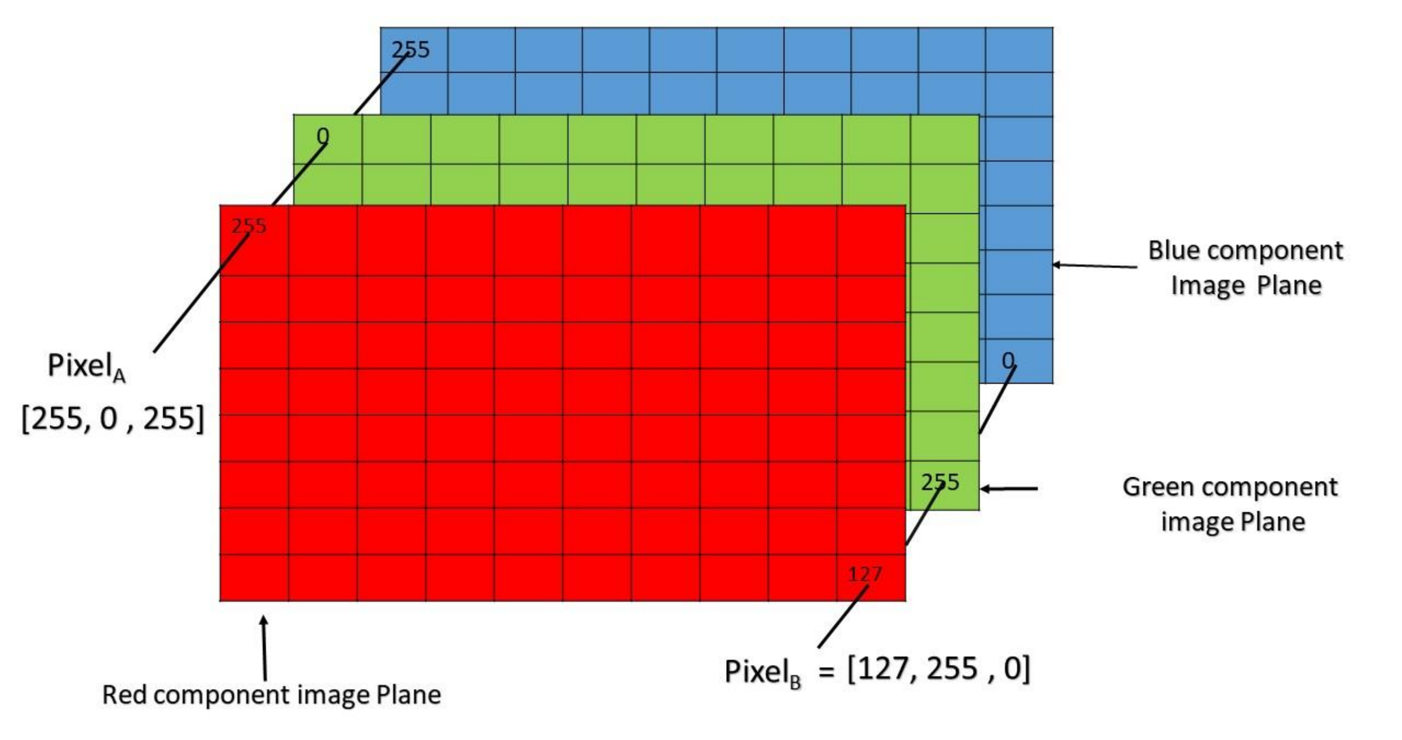
What is an Image?

❖ An image can be represented by a matrix with the dimensions corresponding to the dimensions of the image.



- ❖ Each of the elements of this matrix contains the number (or numbers) representing the colour of the corresponding pixel.

- ❖ If the image is a colour image, then the colour of the pixel is represented by three numbers {R, G, B} (Red, Green and Blue) with each number ranging from 0 to 255 (RGB system).



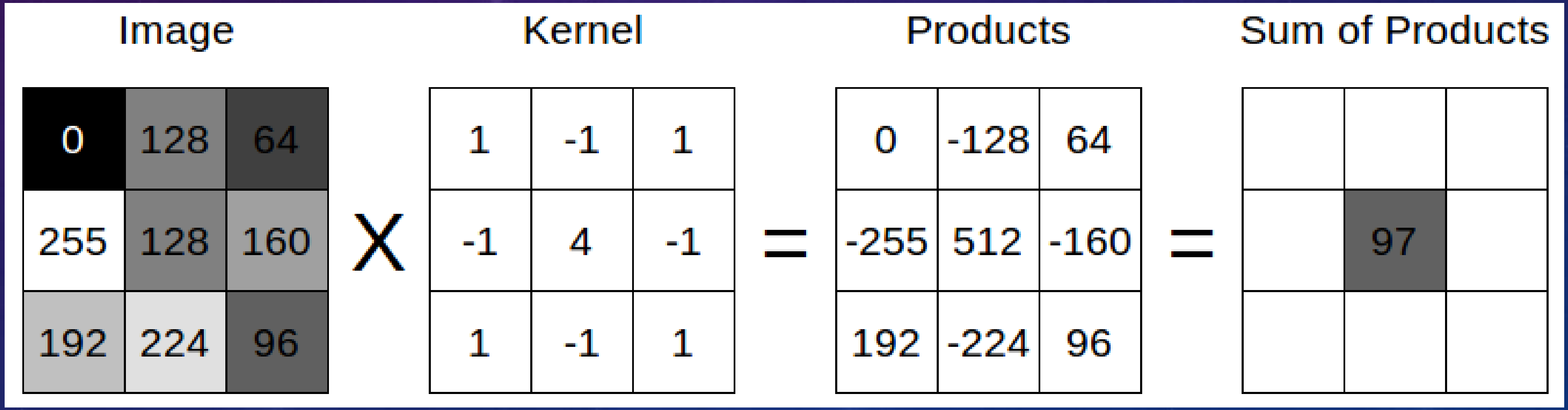
Pixel of an RGB image are formed from the corresponding pixel of the three component images

Relevant Topic from ME501

- Linear Algebra
  - Transformation Matrix (Kernel)
- Convolution

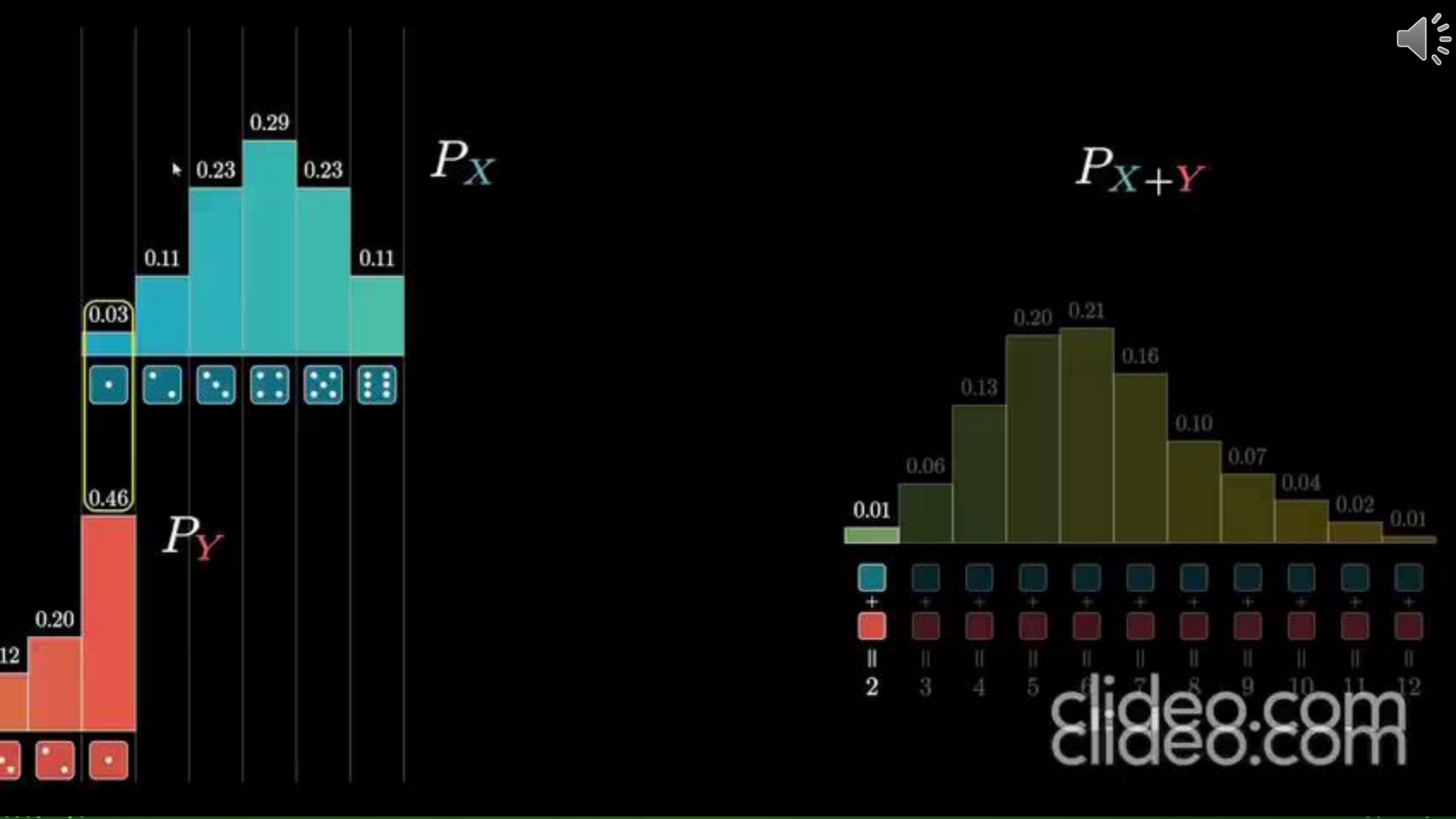
What is Convolution ?

- Convolution is the process of adding each element of the image to its local neighbors, weighted by the kernel.
- This is related to a form of mathematical convolution.



What is

$$(1, 2, 3) * (4, 5, 6)$$



0.29
0.23
0.23
P_X
0.11
0.11
0.03
0.46
P_Y
0.20
12
P_{X+Y}
0.20
0.21
0.16
0.13
0.10
0.06
0.07
0.04
0.01
0.02
0.01
2
3
4
5
9
10
11
12
clideo.com

Methodology

- Convolution Concept : Python
  - Blurring of an Image
  - Sharpening of an Image
- Transformation Matrix Concept : MATLAB
  - RGB Channel
  - Concept of Identity Matrix
  - Grayscale & Sepia Effects

Python Code

```
In [1]: import cv2
        import matplotlib.pyplot as plt
        import numpy as np
```

```
In [2]: path="E:\\IITG\\ME 501 Maths\\Project\\Python\\"
        imgpath = path + "1.jpg"
        # cv2.cvtColor()
        img= cv2.imread(imgpath, 1)
        img= cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

        k= np.array(np.ones((10,10),np.float32))/100 #For Blurring of Image

        output=cv2.filter2D(img,-1,k)
        plt.subplot(1,2,1)
        plt.imshow(img)
        plt.title('Original Image')
        plt.subplot(1,2,2)
        plt.imshow(output)
        plt.title('Blurred Image')
        plt.show()
```

```
In [3]: k= np.array(([0,-1,0],[-1,5,-1],
                     [0,-1,0]),np.float32)     #For sharpening of image
        output=cv2.filter2D(img,-1,k)
        plt.subplot(1,2,1)
        plt.imshow(img)
        plt.title('Original Image')
        plt.subplot(1,2,2)
        plt.imshow(output)
        plt.title('Sharpened Image')
        plt.show()
```

Blurring of an Image

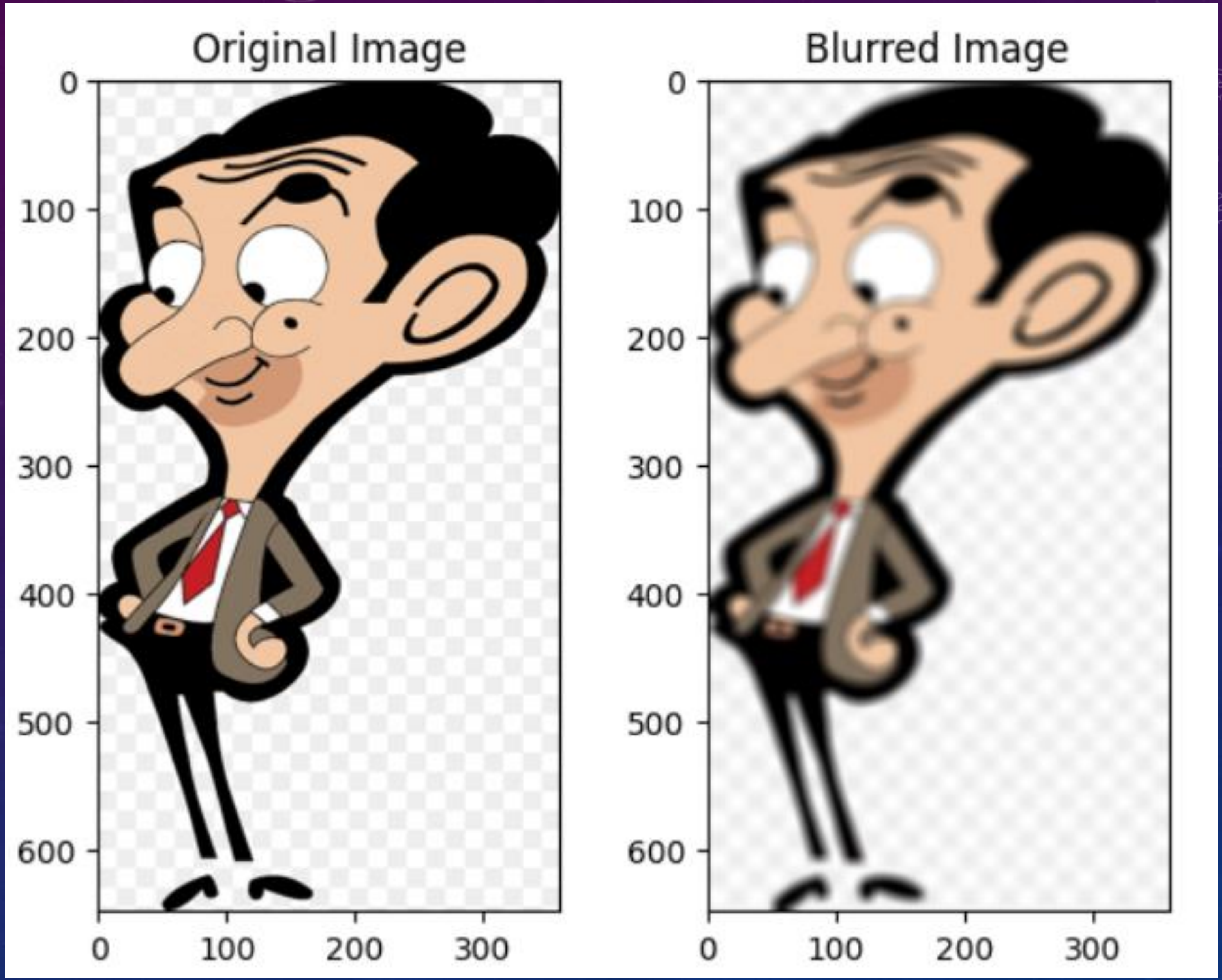
Kernel :

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Box Blur (Normalized)

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

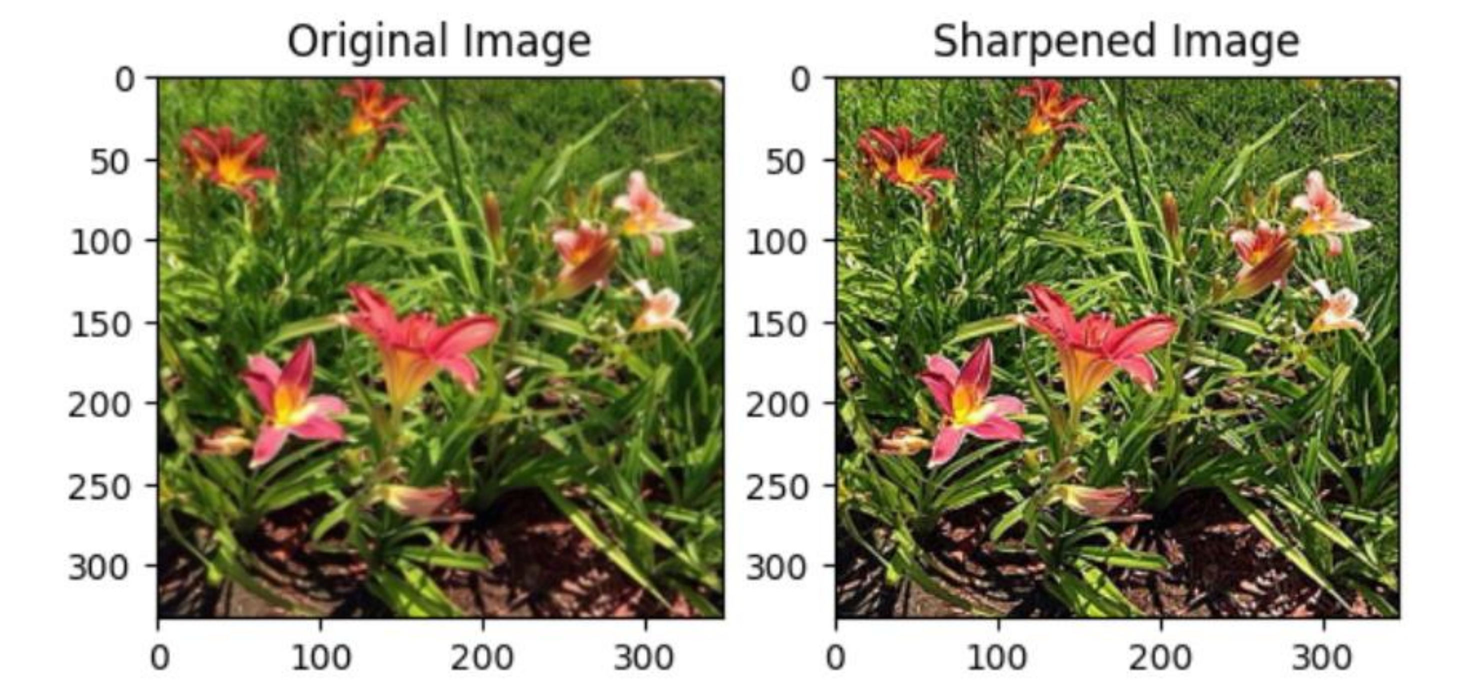
Gaussian Blur (Weighted Averaging)

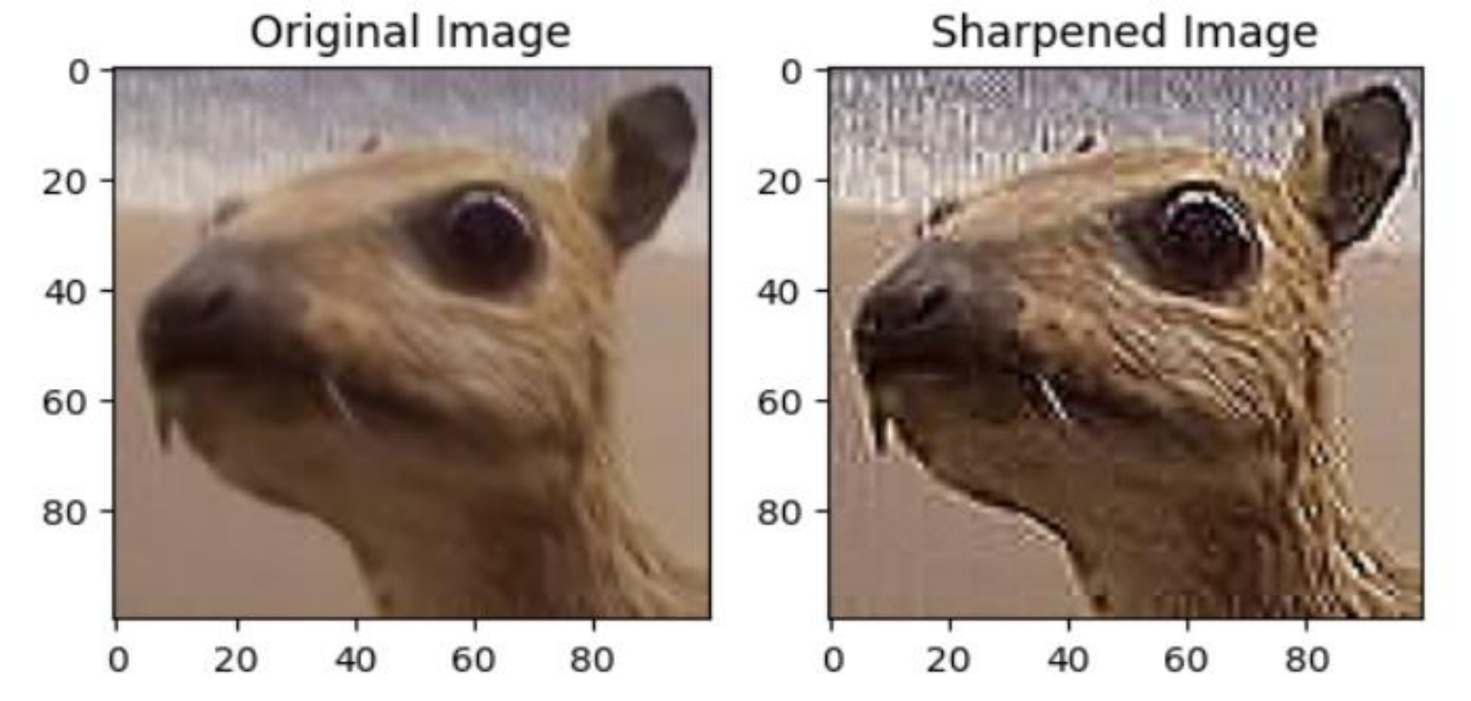

Original Image
Blurred Image
0
100
200
300
400
500
600
0
100
200
300
0
100
200
300
400
500
600
0
100
200
300

Sharpening of an Image

Kernel :

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


Original Image
Sharpened Image
0
50
100
150
200
250
300
0
100
200
300


Original Image
Sharpened Image
0
20
40
60
80
0
20
40
60
80

MATLAB Code

First Part : Image Input and Initialization of the Transformation Matrices

```
clear all
clc
ImJPG=imread('1.jpg');
[m,n,l]=size(ImJPG);

GrayMatrix=[1/3 1/3 1/3; 1/3 1/3 1/3; 1/3 1/3 1/3];
SepiaMatrix=[0.393 0.769 0.189;0.349 0.686 0.168;0.272 0.534 0.131];
IdentityMatrix=[1 0 0; 0 1 0; 0 0 1];
RedChannel=[1 0 0; 0 0 0; 0 0 0];
GreenChannel=[0 0 0; 0 1 0; 0 0 0];
BlueChannel=[0 0 0; 0 0 0; 0 0 1];
```

Second Part : Changes done in the Image at Pixel Level

```
for i=1:m
for j=1:n
PixelColor=reshape(double(ImJPG(i,j,:)),3,1);
ImJPG_Gray(i,j,:)=uint8(GrayMatrix*PixelColor);
ImJPG_Sepia(i,j,:)=uint8(SepiaMatrix*PixelColor);
ImJPG_Identity(i,j,:)=uint8(IdentityMatrix*PixelColor);
Red(i,j,:)=uint8(RedChannel*PixelColor);
Green(i,j,:)=uint8(GreenChannel*PixelColor);
Blue(i,j,:)=uint8(BlueChannel*PixelColor);
end;
end;
```

Third Part : Seeing the Magic happen!

```
figure;
subplot(1,2,1),imshow(ImJPG),subplot(1,2,2),imshow(ImJPG_Identity)
figure;
subplot(1,3,1),imshow(Red),subplot(1,3,2),imshow(Green),subplot(1,3,3),imshow(Blue)
figure;
subplot(1,2,1),imshow(ImJPG),subplot(1,2,2),imshow(ImJPG_Gray)
figure;
subplot(1,2,1),imshow(ImJPG),subplot(1,2,2),imshow(ImJPG_Sepia)
```

RGB Channels of an Image

Kernel :

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Red Channel | Green Channel | Blue Channel


Original Image


Red Channel


Green Channel


Blue Channel

Unchanged Original Image

Kernel : Combination of the Individual RGB Channels

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

Original Image

Filtered Image (No Effect)

Grayscale Effect

Kernel :

$$\begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}$$

Original Image

Filtered Image

Sepia Effect

Kernel

$$\begin{bmatrix} 0.393 & 0.769 & 0.189 \\ 0.349 & 0.686 & 0.168 \\ 0.272 & 0.534 & 0.131 \end{bmatrix}$$

Original Image

Filtered Image

Individual Contribution

- Bharat Anant : MATLAB Code + Documentation
- Digboloy Borah : Matrix Application research + Documentation
- Himanshu Lahare : Python Code + Documentation
- Rishabh Saluja : Convolution Theorem and its application + Documentation
- Vidisha Singh : Research on Various Transformation Kernels + Documentation


THANK YOU!!