LNMIIT
The LNM Institute of Information Technology

PROJECT
Image Enhancer

NAME : DEVANSH SRIVASTAVA (21UCS061)
DHRUV PATEL (21UCS068)

SUBJECT : DIP

MENTOR : ANUKRITI BANSAL

LNMIIT
The LNM Institute of Information Technology

CONTENTS

LNMIIT
The LNM Institute of Information Technology

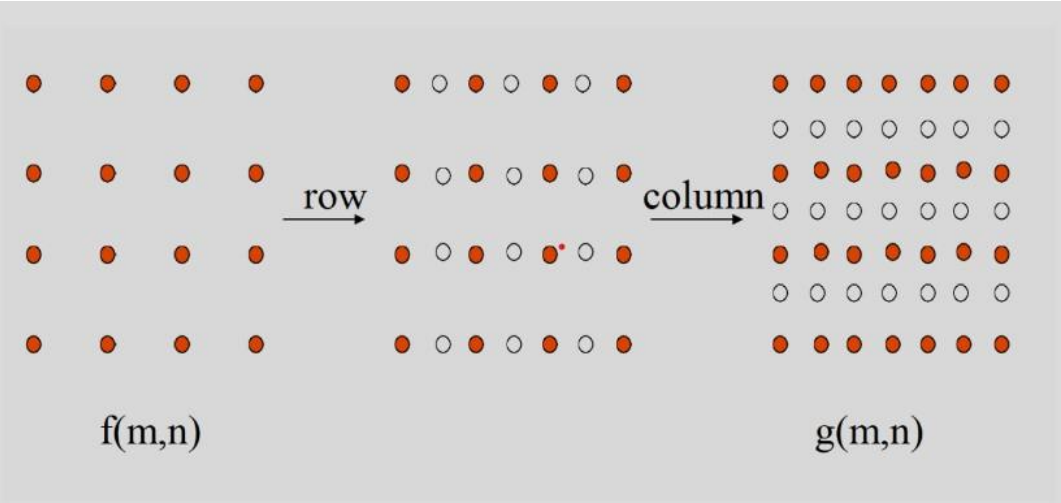
TASK 1 :

Scaling up of a Image

Ques : What do we mean by scaling up the image?
Ans : Here scaling up of an image is understood as increasing the size of image and hence also the number of pixels. Now the new pixels are given the intensity values by two methods namely:
1) Replication 2) Interpolation

Replication

Here the new pixels are given the same value as the previous neighbouring one. First we replicate the pixels row wise and then we replicate it columnwise.



LNMIIT
The LNM Institute of Information Technology

As we can see in the above image we can replicate the new pixels same as their neighbouring image and form a new image

Code for replication of a image :

```
import cv2 as cv
import numpy as np



# callback function to implement
replicated image
def click_event(event, x, y, flags,
param):
    if event == cv.EVENT_LBUTTONDOWN:
        print()
        print('Select coordinates of
the top left corner and bottom right
corner of the desired window size')
```

```
        print()
        print(x, ',', y)
        points.append([y, x])
        if (len(points) == 2):
            # Create window to
capture the desired window size

cv.namedWindow('replicated_image',
cv.WINDOW_AUTOSIZE)
            copy_img =
img[points[0][0]:points[1]
                                [0],
points[0][1]:points[1][1]]

            width =
2*int(copy_img.shape[1])
            height =
2*int(copy_img.shape[0])
            frame = np.zeros((height,
width, 3), np.uint8)
```

```
            for i in range(height):
# Looping through complete image
                for j in
range(width):
                    if i % 2 == 0:
                        if j % 2 !=
0:
                            frame[i,
j] = frame[i, j-1]
                        else:
                            frame[i,
j] = copy_img[i//2, j//2]
                    else:
                        frame[i, j] =
frame[i-1, j]

cv.imshow("replicated_image", frame)
```

```
points = []
img = cv.imread("lena_color.tif", 3)
# Reading original image file
if img is None:
    print("Error:Image cannot be
loaded")
    exit(0)
else:
    print("Image loaded succesfully")
    # Display original image to
compare with the selected window
    cv.imshow('original_image', img)
    # This function will track the
coordinates of the top left and
bottom right limit of the desired
window size to display the replicated
image

cv.setMouseCallback('original_image',
click_event)
```

```
cv.waitKey(0)
cv.destroyAllWindows()
```

LNMIIT
The LNM Institute of Information Technology

BILINEAR INTERPOLATION

In scaling up an image by the interpolation method we give the new pixels value average of the neighbouring pixels. Thus scaling up by this method usually gives a better image than the former one.

1. First we give the average intensity values row wise and then we can take the average value and give it to the pixels in between the two rows.
2. Now for the border ones as thor is only one neighbouring pixel so we will assume the other one as 0 and thus half the value of the neighbouring pixels would be given to them.

Code for scaling up by interpolation :

```
import numpy as np
import cv2 as cv

# This function will do select the desired window to be interpolated


def click_event(event, x, y, flags, param):
    if event == cv.EVENT_LBUTTONDOWN:
        print("Printing the Selected Coordinates")
        print(x, ',', y)
        points.append([y, x])
        if (len(points) == 2):
            # Now scaling to 2x
```

```
            cv.namedWindow('interpolated_display', cv.WINDOW_AUTOSIZE)
            copy_img = img[points[0][0]:points[1]
                           [0], points[0][1]:points[1][1]]
            width = 2*int(copy_img.shape[1])
            height = 2*int(copy_img.shape[0])
            frame = np.zeros((height, width, 3), np.uint8)
            # This loop will copy the intensities from the original image
to new image
            for i in range(0, height, 2):
                for j in range(0, width, 2):
                    frame[i][j] = copy_img[i//2][j//2]
            # This loop will fill the middle elements present in the even
index of each row
            for i in range(0, height, 2):
                for j in range(1, width, 2):
                    if j != width-1:
                        frame[i][j] = (frame[i][j-1]+frame[i][j+1])//2
                    else:
                        frame[i][j] = frame[i][j-1]//2
            # This loop will take the average values rowwise and thus
final image is ready
            for i in range(1, height, 2):
                for j in range(width):
                    if i != height-1:
                        frame[i][j] = (frame[i-1][j]+frame[i+1][j])//2
                    else:
                        frame[i][j] = frame[i-1][j]//2

            cv.imshow("interpolated_display", frame)


points = []
# Reading the image
img = cv.imread("lena_color.tif", 3)
if img is None:
    print("Wrong image")
    exit(0)
else:
    print("Image read successfully")
    # This will show the original image
```

```
    print("Select the desired Image to be interpolated :")
    print()
    print("NOTE : Select the first point by clicking on top left corner
and the second point as bottom-right corner of your desired image")

    cv.imshow('image', img)
    cv.setMouseCallback('image', click_event)
    # This will note the coordinates of your clicks and finally
interpolate your image
# Finally waiting for key press and then destroying the windows
cv.waitKey(0)
cv.destroyAllWindows()
```

LNMIIT
The LNM Institute of Information Technology

Scaling down of the image

Here we want our image to be small and thus we delete some of the data(some the pixels).
Example : Suppose we want the 4x4 image to be converted to the 2x2 image then we delete the alternate rows and columns and finally we get the 2x2 image.

In our case if we will again get back the original image from the scaled up image by removing the new pixels i.e. by deleting the alternate rows and columns.

Code for scaling down of the Image :

```
import cv2 as cv
import numpy as np


def scale_down(new_img):  # This function will remove the alternate rows
and columns
    height = new_img.shape[1]//2
    width = new_img.shape[0]//2

    frame = np.zeros((height, width, 3), np.uint8)

    for i in range(height):
        for j in range(width):
            frame[i][j] = new_img[i*2][j*2]

    cv.imshow("scaled_down_image", frame)  # Shows the scaled down image
```

```
img = cv.imread("lena_color.tif", 3)
if img is None:
    print("No image detected")
    exit(0)
else:
    print("Image read successfully")
    cv.imshow("original_image", img)  # Reads the color image
    scale_down(img)
cv.waitKey(0)
cv.destroyAllWindows()
```

LNMIIT
The LNM Institute of Information Technology

Task : 2

BIT PLANE SLICING

Bit plane slicing is a method of representing an image with one or more bits of the byte used for each pixel. Here we will convert the image into the grayscale first and thus we will have the pixels represented with 8 bits. Now we can derive all the eight bits individually and thus we can get 8 images based on the bits.

Now we will perform the operations on the obtained images in the following way :
Case 1 : We will add the 8th bit plane and 7th bit plane and form an image.
Case 2 : We will add the 8th,7th and 6th bit plane and form an image.
Case 3 : We will add the 8th,7th, 6th and 5th bit plane and form an image.

```
import cv2
import numpy as np
#Only if you using the google colab
```

```
#from google.colab.patches import
cv2_imshow #importing the imshow
function to display the image.

print()
print("-------->Welcome<---------- ")
print()
#Reading the image in the grayscale
and saving it in img variable

img = cv2.imread('lena_color.tif',0)
print("Showing the original image in
Grayscale")
print()


cv2.namedWindow('Display')
cv2.imshow('Display',img)
print(" Note : Close the window to
see the next image ")
```

```
cv2.waitKey(0)

#Setting the height and width of the
image into variables

height=int(img.shape[0])
width=int(img.shape[1])

#Making a list of 8 images such that
all the pixel values are set to zero
bit_plane = list()
for i in range(8):

bit_plane.append(np.zeros((height,wid
th,1),np.uint8))

# This section of code will not set
the proper values of 8 images created
beforehand
for k in range(8):
```

```
    for i in range(height):
        for j in range(width):
            #Converting the pixel
value into the binary and saving it
as a form of string in temp variable
            temp =
np.binary_repr(img[i][j] ,width=8)

            # Now if the bit value is
0 then that bit plane pixel value
will be set to 0 otherwise
appropriate value will be given
            if int(temp[7-k]) ==0 :
              bit_plane[k][i][j] = 0
            else:
              bit_plane[k][i][j] =
int(pow(2,k))

# Here showing all the bit_plane
images individually
```

```
for i in range(8):
    print()
    print("Opened image is the image
of bit plane " + str(i+1))
    #cv2.namedWindow('Display')

cv2.imshow('Display',bit_plane[i])
    print(" Note : Close the window
to see the next image ")

    cv2.waitKey(0)
    print()

#Now creating another image which
will be used to print the Final
result.
final =
np.zeros((height,width,1),np.uint8)
```

```
#Now casewise showing the image in
the output
print()
print("Case 1 : Resultant image
obtained by the addition of 8th and
7th Plane : ")

for i in range(height):
    for j in range(width):
        final[i][j] =
bit_plane[7][i][j] +
bit_plane[6][i][j]
#cv2.namedWindow('Display')
cv2.imshow('Display',final)
print(" Note : Close the window to
see the next image ")
cv2.waitKey(0)
print()
```

```
print("Case 2 : Resultant image
obtained by the addition of 8th,7th
and 6th Plane : ")

for i in range(height):
    for j in range(width):
        final[i][j] =
bit_plane[7][i][j] +
bit_plane[6][i][j] +
bit_plane[5][i][j]
#cv2.namedWindow('Display')
cv2.imshow('Display',final)
print(" Note : Close the window to
see the next image ")
cv2.waitKey(0)

print()
print("Case 3 : Resultant image
obtained by the addition of
8th,7th,6th and 5th Plane : ")
```

```
for i in range(height):
    for j in range(width):
        final[i][j] =
bit_plane[7][i][j] +
bit_plane[6][i][j] +
bit_plane[5][i][j] +
bit_plane[4][i][j]

#cv2.namedWindow('Display')
cv2.imshow('Display',final)
print(" Note : Close the window to
see the next image ")

cv2.waitKey(0)
cv2.destroyAllWindows
```

LNMIIT
The LNM Institute of Information Technology

Result and Observations:

From Question 1 :

We learnt that we can resize the original image into a new bigger image by Replication or by interpolation.

This can be used to get the clear zoomed image(interpolation better as we are taking average of the pixel may be some time replication can also be fruitful if all pixels have same intensity values)

From Question 2 :

We observed that most of the information is saved in the MSB bits of the image pixel values.

Thus we can use these bit planes when we want to transmit the image from one device to another with limited size and then finally we can reconstruct the image.

LNMIIT
The LNM Institute of Information Technology

Learnings and Conclusions

- We explored many new things as it was the first assignment.
- We got a little familiar with the python language and also we got a glimpse of the magic of Image Processing.
- Also working in a team was a great experience and looking for still better experiences