Exp 4: Image Enhancement - Understanding Color spaces, color space conversion, Histogram equalization, Convolution, Image smoothing, Gradients, Edge Detection

Aim:

To write a Python program to implement the following Image Enhancement operations

1. Understanding Color Spaces.
2. Color Space Conversion.
3. Histogram Equalization.
4. Convolution.
5. Image Smoothing.
6. Gradients.
7. Edge Detection.

Tools required:

1. Computer with 32 bit or 64 bit Windows Operating system and 4GB RAM
2. Python3
3. OpenCV computer vision Library for Open CV in Python

Algorithm:

Step 1: Open a python Script file. Step 2: Load the image.
Step 3: Perform Understanding Color Spaces, Color Space Conversion, Histogram Equialization, Convolution, Image Smoothing, Gradients and Edge Detection over the input image.
Step 4: visualize the image using open CV.

PROGRAM:

```
import cv2

import numpy as np

from matplotlib import pyplot as plt

#To load the image and display

image=cv2.imread("image.jpeg")

cv2.imshow("original_image",image)

#The following command waits till we press any

key cv2.waitKey(0)


#To convert RGB to GRAY image

image_gray=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY

) cv2.imwrite('image_gray.jpeg',image_gray)

cv2.imshow("image_gray",image_gray)

cv2.waitKey(0)


#To get histogram equalised image

equalised_image=cv2.equalizeHist(image_gray)

cv2.imwrite('equalised_image.jpeg',equalised_image

) cv2.imshow("equalised_image",equalised_image)

cv2.waitKey(0)


#To plot histogram equalisation graph

histr = cv2.calcHist([equalised_image],[0],None,[256],[0,256])

plt.plot(histr)

plt.show()

#To convolve the given image
```

```
kernel=np.ones((3,3),np.float32)/2.0
convolved_image=cv2.filter2D(image,-1,kernel)
cv2.imwrite('convolved_image.jpeg',convolved_image)

cv2.imshow("convolved_image",convolved_image)
cv2.waitKey(0)

#To blurr the given image using guassianblur
#the (5,5) is the kernel size and should be odd num always
#FORMAT cv2.GaussianBlur(image_name,kernal size(height and width),standard
deviation of x and
as well as y
gaussianblurred_image=cv2.GaussianBlur(image,(5,5),0)
cv2.imwrite('gaussianblurred_image.jpeg',gaussianblurred_image)
cv2.imshow("gaussianblurred_image",gaussianblurred_image)
cv2.waitKey(0)

#To blurr the given image using medianblur
#FORMAT cv2.medianBlur(image_name,kernal size(single value not like
coordinates)) medianblurred_image=cv2.medianBlur(image,5)
cv2.imwrite('medianblurred_image.jpg',medianblurred_image)
cv2.imshow("medianblurred_image",medianblurred_image)
cv2.waitKey(0)

#To blurr the given image using blur
#FORMAT cv2.blur(image_name,kernal size(height and width),standard deviation of x
and as well as
y)
```

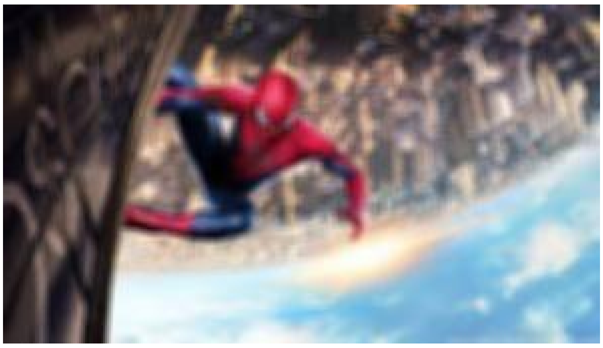
```
blurred_image=cv2.blur(image,(5,5),0)
cv2.imwrite('blurred_image.jpg',blurred_image)
cv2.imshow("blurred_image",blurred_image)
cv2.waitKey(0)
#To find the gradient of the given image using sobel operator for x,y
gradient_x=cv2.Sobel(image_gray,cv2.CV_64F,1,0,ksize=3)
cv2.imwrite('gradient_x.jpeg',gradient_x)
cv2.imshow("gradient_x",gradient_x)
cv2.waitKey(0)

gradient_y=cv2.Sobel(image_gray,cv2.CV_64F,0,1,ksize=3)
cv2.imwrite('gradient_y.jpeg',gradient_y)
cv2.imshow("gradient_y",gradient_y)
cv2.waitKey(0)

#To find the edges of the given image using canny operator
#FORMAT cv2.Canny(image_name, strongedge,weakedge) where
strongedge>weakedge edges=cv2.Canny(image,200,100)
cv2.imwrite('edges.jpeg',edges)
cv2.imshow("edges",edges)
cv2.waitKey(0)

#To close all windows created till now
cv2.destroyAllWindows()
```

Result:

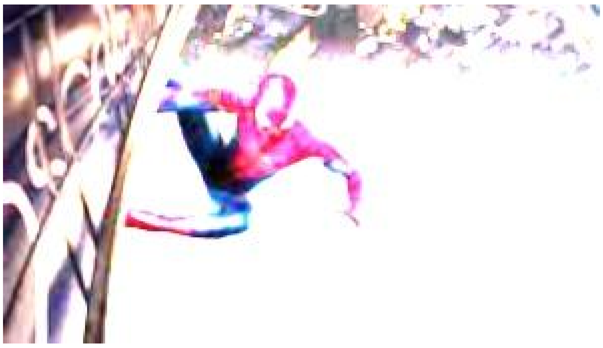
Thus, the Python program to implement the Understanding of Color Spaces, Color Space Conversion, Histogram Equalization, Convolution, Image Smoothing, Gradients, and Edge Detection is executed successfully.

Output:

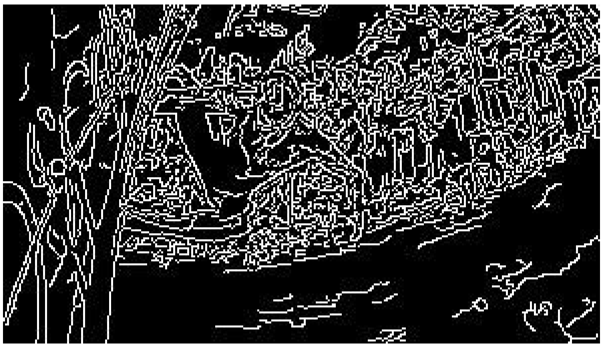
BLURRED



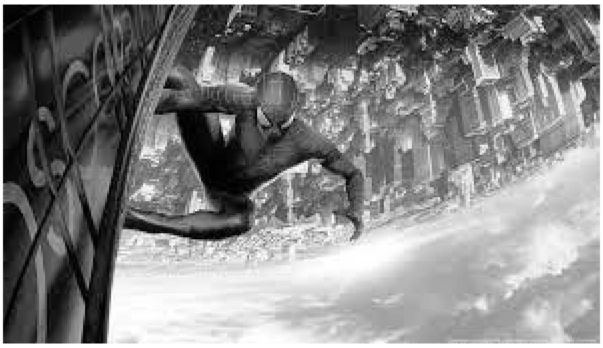
CONVOLVED



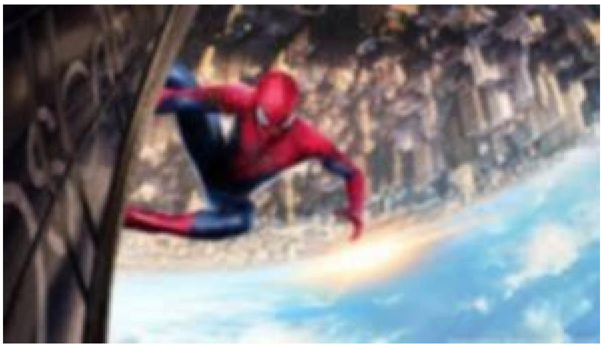
EDGES



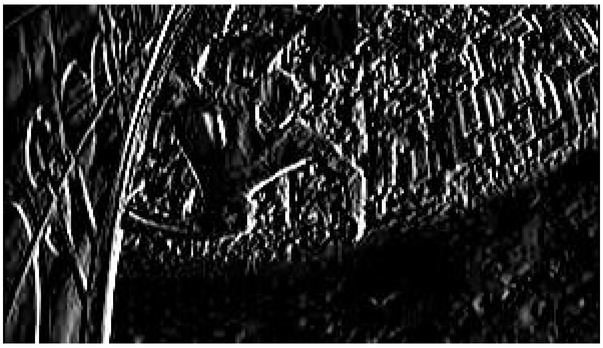
EQUALIZED IMAGE



GAUSSIAN BLURRED



GRADIENT X



GRADIENT Y

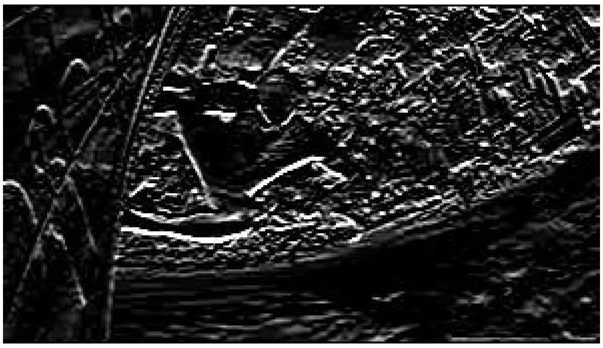


IMAGE GRAY

MEDIAN BLURRED