IMCOST

MCA AX. 2022-2023

Journal of Image Processing

ASM's Institute of Management & Computer Studies, Thane

Candidate Full Name	**Omkar Bharat Khandare**
Roll No.	**MC2223044**
Course	**Masters of Computer Applications (MCA)**
Semester	**II**
Subject Faculty Incharge	**Prof. Mahesh Mahajan**

IMCOST

Audyogik Shikshan Mandal's

Institute of Management & Computer Studies, Thane

Certificate This is to certify that Mr. **Omkar Bharat Khandare** Student of **MCA** Course, **First year**, **Semester 2**, Roll No **MC2223044** has successfully complete the required number of practical in subject of **Image Processing** as prescribed by the University of Mumbai and our supervision during the academic-year 2022-2023.

Practical In-Charge **Internal Examiner**

Date: **Date:**

External Examiner. **Director**

Date: **IMCOST**

College Seal

INDEX

13	Program to implement Huffman coding technique for image compression.	

OpenCv

- OpenCV is a programming library/package that has been created especially for allowing programmers to enter the world of Computer Vision. The primary developer of the OpenCV package is Intel Corporation, and the package was released to the public during the year 1999-2000.
- OpenCV stands for Open-Source Computer Vision (Library). It is the most commonly used, popular, and well-documented Computer Vision library. It is open-source, which means that one does not require a license to utilize the software.

NumPy

- NumPy is a Python library used for working with arrays.
- It also has functions for working in domain of linear algebra, fourier transform, and matrices.
- NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. • NumPy stands for Numerical Python.

- In Python we have lists that serve the purpose of arrays, but they are slow to process.
  - NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
  - The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.
  - Arrays are very frequently used in data science, where speed and resources are very important.

Steps to install OpenCV

- **Open Anaconda Prompt**

  Start Menu / Anaconda3 / Anaconda Prompt

- **In Anaconda Prompt, type commands to install necessary libraries:**

pip install opencv-python

Practical 1

- **Title: Program to display image using read and write operation.**
- imread() helps us read an image
- imshow() displays an image in a window
- imwrite() writes an image into the file directory

Theory :

Program to display image using read and write operation.

Reading, displaying, and writing images are basic to image processing and computer vision. Even when cropping, resizing, rotating, or applying different filters to process images.

For reading an image, use the imread() function in OpenCV. Here's the syntax:

imread(filename, flags)

It takes two arguments:

The first argument is the image name, which requires a fully qualified pathname to the file.

The second argument is an optional flag that lets you specify how the image should be represented.

OpenCV offers several options for this flag, but those that are most common include:

cv2.IMREAD_UNCHANGED or -1 cv2.IMREAD_GRAYSCALE

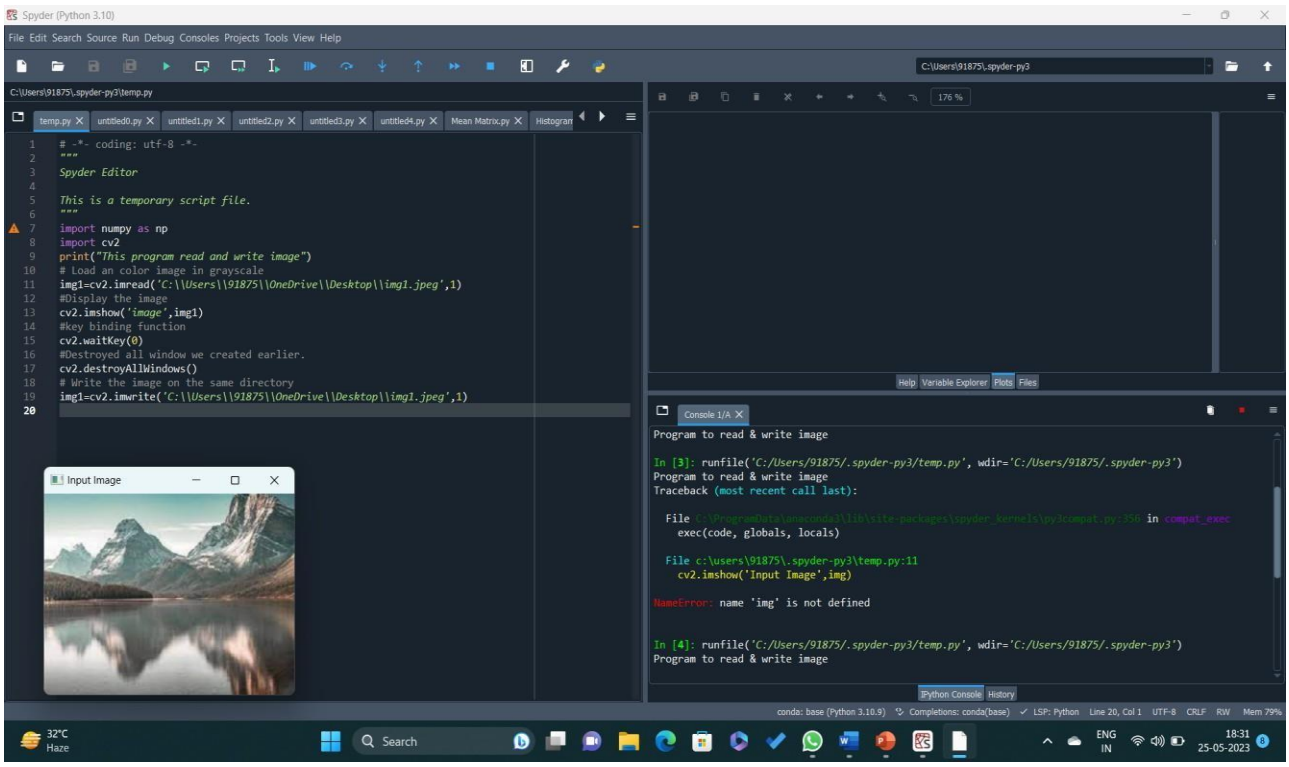
or 0

cv2.IMREAD_COLOR or 1

Source Code :

```
import numpy as np import cv2 print("This program read
and write image") # Load an color image
in grayscale
img = cv2.imread('C:\\Users\\IM160\\Desktop\\Python Practicals\\Practical
1\\bike.jpg',1)
#Display the image
cv2.imshow('image',img)
#key binding function
cv2.waitKey(0)
#Destroyed all window we created earlier. cv2.destroyAllWindows()
# Write the image on the same directory
cv2.imwrite("C:\\Users\\IM160\Desktop\\Python Practicals\\Practical
1\\grayscale.jpg", img);
```

Output :


Spyder (Python 3.10)
File Edit Search Source Run Debug Consoles Projects Tools View Help
-*- coding: utf-8 -*-
"""
Spyder Editor
This is a temporary script file.
"""
import numpy as np
import cv2
print("This program read and write image")
# Load an color image in grayscale
img1=cv2.imread('C:\\Users\\91875\\OneDrive\\Desktop\\img1.jpeg',1)
#Display the image
cv2.imshow('image',img1)
#key binding function
cv2.waitKey(0)
#Destroyed all window we created earlier.
cv2.destroyAllWindows()
# Write the image on the same directory
img1=cv2.imwrite('C:\\Users\\91875\\OneDrive\\Desktop\\img1.jpeg',1)
Input Image
Program to read & write image
In [3]: runfile('C:/Users/91875/.spyder-py3/temp.py', wdir='C:/Users/91875/.spyder-py3')
Program to read & write image
Traceback (most recent call last):
File c:\users\91875\.spyder-py3\temp.py:11
cv2.imshow('Input Image',img)
NameError: name 'img' is not defined
In [4]: runfile('C:/Users/91875/.spyder-py3/temp.py', wdir='C:/Users/91875/.spyder-py3')
Program to read & write image
32°C
Haze
Search
ENG
IN
18:31
25-05-2023

Practical 2

• Title : Program to enhance image arithmetic and logical operations.

- The input images can be subjected to arithmetic operations such as addition, subtraction, multiplication and bitwise operations (AND, OR, NOT, XOR). These operations can help to improve the qualities of the input photos. • Image Addition :

- Python OpenCV provides cv2.addWeighted() method to add two images. This method calculates the weighted sum of two arrays. It returns an image in the output, which is a combination of the corresponding pixel values of the input images • Image Subtraction:

- In Python OpenCV, the cv2.subtract() method is used to subtract two images. Both images should be of same size and type.

• Image Multiplication:

- In Python OpenCV, cv2.multiply() method is used to multiply two images.
- OpenCV Bitwise AND
- Python OpenCV provides cv2.bitwise_and() method to perform bitwise AND logical operation. It combines corresponding pixels of two image buffers by a bitwise AND operation. It is commonly used for detecting differences in the input images. It returns an image highlighting the target regions with a binary mask.
- OpenCV Bitwise OR
- Python OpenCV provides cv2.bitwise_or() method to perform bitwise OR logical operation. It combines corresponding pixels of two image buffers by a bitwise OR operation. The bitwise OR operator are useful for processing

binary-valued images (0 or 1) to detect objects which have moved between frames.

- OpenCV Bitwise XOR
- Python OpenCV provides cv2.bitwise_xor() method to perform bitwise XOR logical operation. It combines corresponding pixels of two image buffers by a bitwise XOR operation. It is also used for processing binary-valued images (0 or 1) to detect objects which have moved between frames.
- OpenCV Bitwise NOT
- Bitwise NOT or complement is a unary operation that performs logical negation on each bit. It is forming the ones' complement of the given binary value. It means the bits that are 1 become 0, and those that are 0 become 1. Python OpenCV provides cv2.bitwise_not() method to perform a bitwise NOT operation on each pixel of the input image. It inverts the image representation.

- **Title: Program to enhance image arithmetic and logical operations. •**

2.1 Program to perform addition operation on image.

Theory :

Program to enhance image using image arithmetic and logical operations.

Arithmetic Operations like Addition, Subtraction, and Bitwise Operations(AND, OR, NOT, XOR) can be applied to the input images. These operations can be helpful in enhancing the properties of the input images. The Image arithmetics are important for analyzing the input image properties. The operated images can be further used as an enhanced input image, and many more operations can be applied for clarifying, thresholding, dilating etc of the image.

Addition of Image:

We can add two images by using function **cv2.add()**. This directly adds up image pixels in the two images.

Syntax: cv2.add(img1, img2)

Subtraction of Image:

Just like addition, we can subtract the pixel values in two images and merge them with the help of cv2.subtract(). The images should be of equal size and depth.

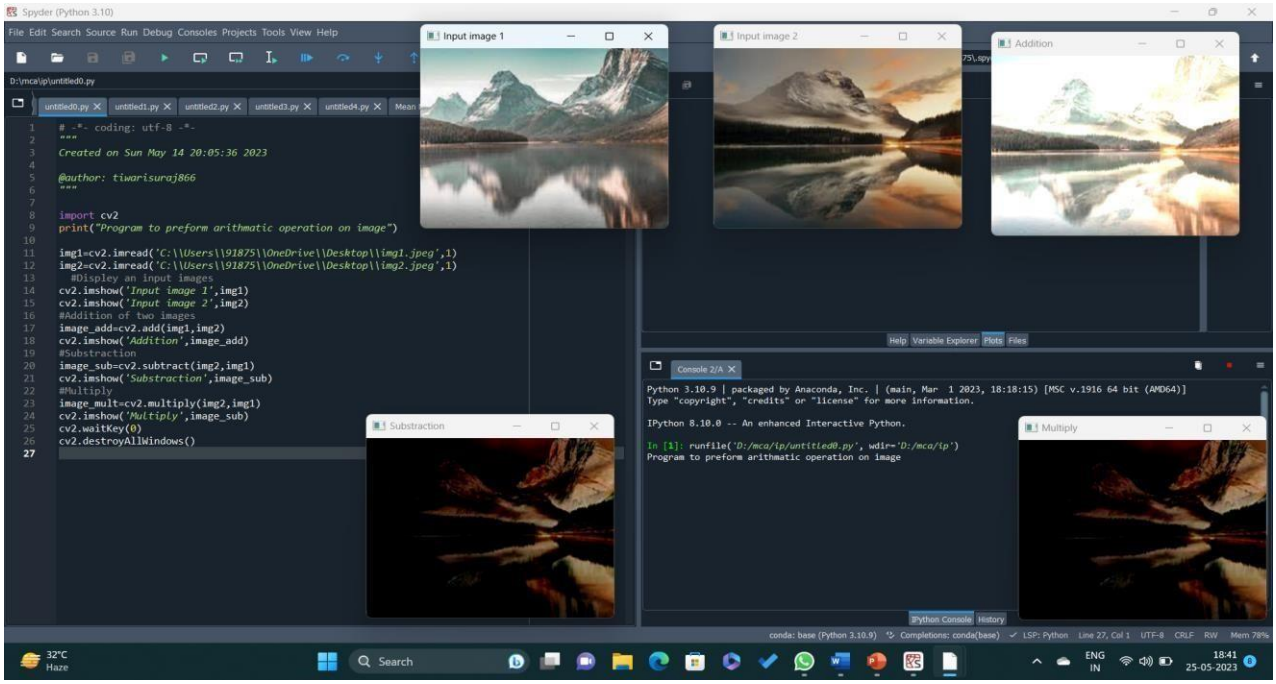
Syntax: cv2.subtract(src1, src2)

Source Code:

```
# -*- coding: utf-8 -*-

"""

Created on Sun May 14 20:05:36 2023

@author: OMKAR """ import cv2


print("Program to preform arithmatic operation on image")

img1=cv2.imread('C:\\Users\\91875\\OneDrive\\Desktop\\img1.jpeg',1)

img2=cv2.imread('C:\\Users\\91875\\OneDrive\\Desktop\\img2.jpeg',1)

#Displey an input images

cv2.imshow('Input image 1',img1)

cv2.imshow('Input image 2',img2)

#Addition of two images

image_add=cv2.add(img1,img2)

cv2.imshow('Addition',image_add)

#Substraction image_sub=cv2.subtract(img2,img1)

cv2.imshow('Substraction',image_sub)
```

```
#Multiply

image_mult=cv2.multiply(img2,img1)

cv2.imshow('Multiply',image_sub)

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

Output :



2.2 Program to perform logical operation on image. Source

Code:

```
# -*- coding: utf-8 -*-

"""
```

Created on Sun May 14 21:25:08 2023

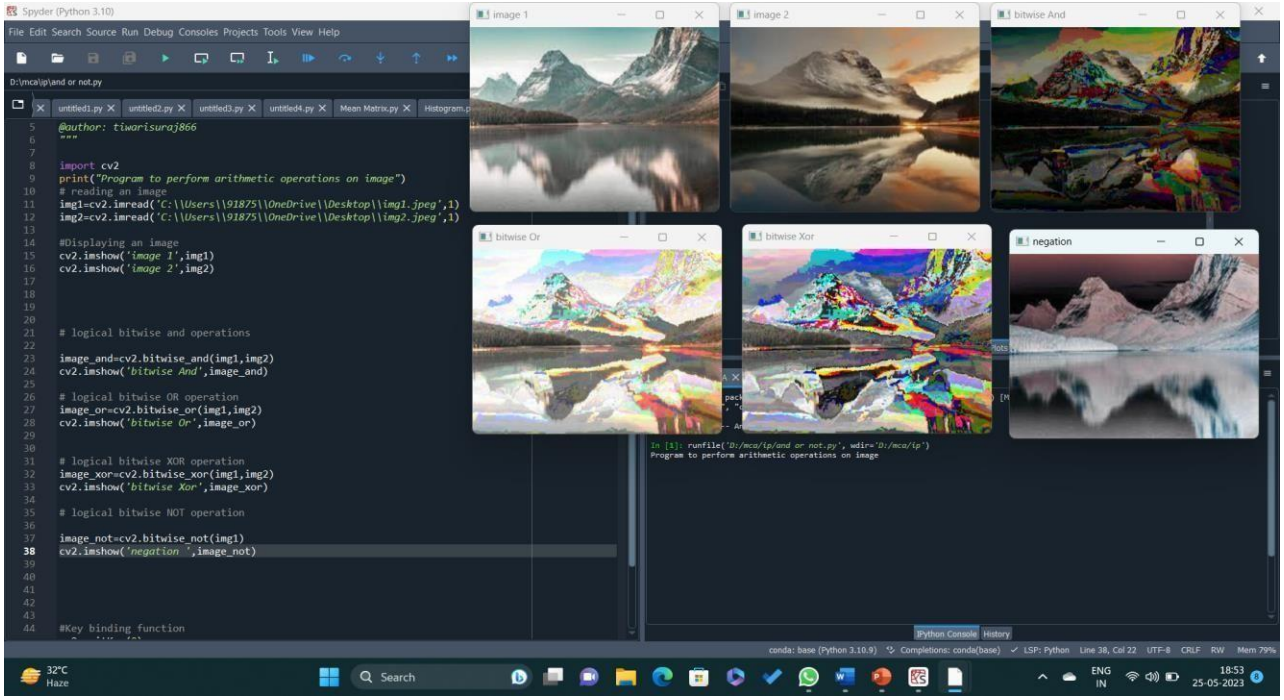
@author: OMKAR

"""

```
import cv2
print("Program to perform arithmetic operations on image")
# reading an image img1=cv2.imread('C:\\Users\\91875\\OneDrive\\Desktop\\img1.jpeg',1)
img2=cv2.imread('C:\\Users\\91875\\OneDrive\\Desktop\\img2.jpeg',1)
#Displaying an image
cv2.imshow('image 1',img1)
cv2.imshow('image 2',img2) # logical
bitwise and operations
image_and=cv2.bitwise_and(img1,img2
) cv2.imshow('bitwise And',image_and)
# logical bitwise OR operation
image_or=cv2.bitwise_or(img1,img2)
cv2.imshow('bitwise Or',image_or)

# logical bitwise XOR operation
image_xor=cv2.bitwise_xor(img1,img2)
cv2.imshow('bitwise Xor',image_xor) #
logical bitwise NOT operation
image_not=cv2.bitwise_not(img1)
```

```
cv2.imshow('negation ',image_not)

#Key binding function cv2.waitKey(0)

#Destroy all previous window

cv2.destroyAllWindows()
```

Output :


Spyder (Python 3.10)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\mca\ip\and or not.py
untitled1.py
untitled2.py
untitled3.py
untitled4.py
Mean Matrix.py
Histogram.p
@author: tiwarisuraj866
"""
import cv2
print("Program to perform arithmetic operations on image")
# reading an image
img1=cv2.imread('C:\\Users\\91875\\OneDrive\\Desktop\\img1.jpeg',1)
img2=cv2.imread('C:\\Users\\91875\\OneDrive\\Desktop\\img2.jpeg',1)
#Displaying an image
cv2.imshow('image 1',img1)
cv2.imshow('image 2',img2)
# logical bitwise and operations
image_and=cv2.bitwise_and(img1,img2)
cv2.imshow('bitwise And',image_and)
# logical bitwise OR operation
image_or=cv2.bitwise_or(img1,img2)
cv2.imshow('bitwise Or',image_or)
# logical bitwise XOR operation
image_xor=cv2.bitwise_xor(img1,img2)
cv2.imshow('bitwise Xor',image_xor)
# logical bitwise NOT operation
image_not=cv2.bitwise_not(img1)
cv2.imshow('negation ',image_not)
#Key binding function
image 1
image 2
bitwise And
bitwise Or
bitwise Xor
negation
In [1]: runfile('D:/mca/ip/and or not.py', wdir='D:/mca/ip')
Program to perform arithmetic operations on image
IPython Console
History
conda: base (Python 3.10.9)
Completions: conda(base)
LSP: Python
Line 38, Col 22
UTF-8
CRLF
RW
Mem 79%
32°C
Haze
Search
ENG
IN
18:53
25-05-2023

Practical 3

- Title : Program to Implement Image Negative.
- Image negation is produced by subtracting each pixel from the maximum intensity value.
- For e.g. for an 8-bit image ,the maximum intensity value is 28 -1 = 255.Thus each pixel is subtracted from 255 to produces the resulting image.
- Thus the transformation function used in image negative is :

  s =T(r) = L-1-r

  Where, L-1 is maximum intensity value

  s is output pixel value.

  r is input pixel value.

Theory :

For grayscale images, light areas appear dark and vice versa. For color images, colors are replaced by their complementary colors. Thus, red areas appear cyan, greens appear magenta, and blues appear yellow, and vice versa.

Image negative is produced by subtracting each pixel from the maximum intensity value. e.g. for an 8-bit image, the max intensity value is $2^8 – 1 = 255$, thus each pixel is subtracted from 255 to produce the output image.

Program to implement Image Negative.

$s = T(r) = L – 1 – r$

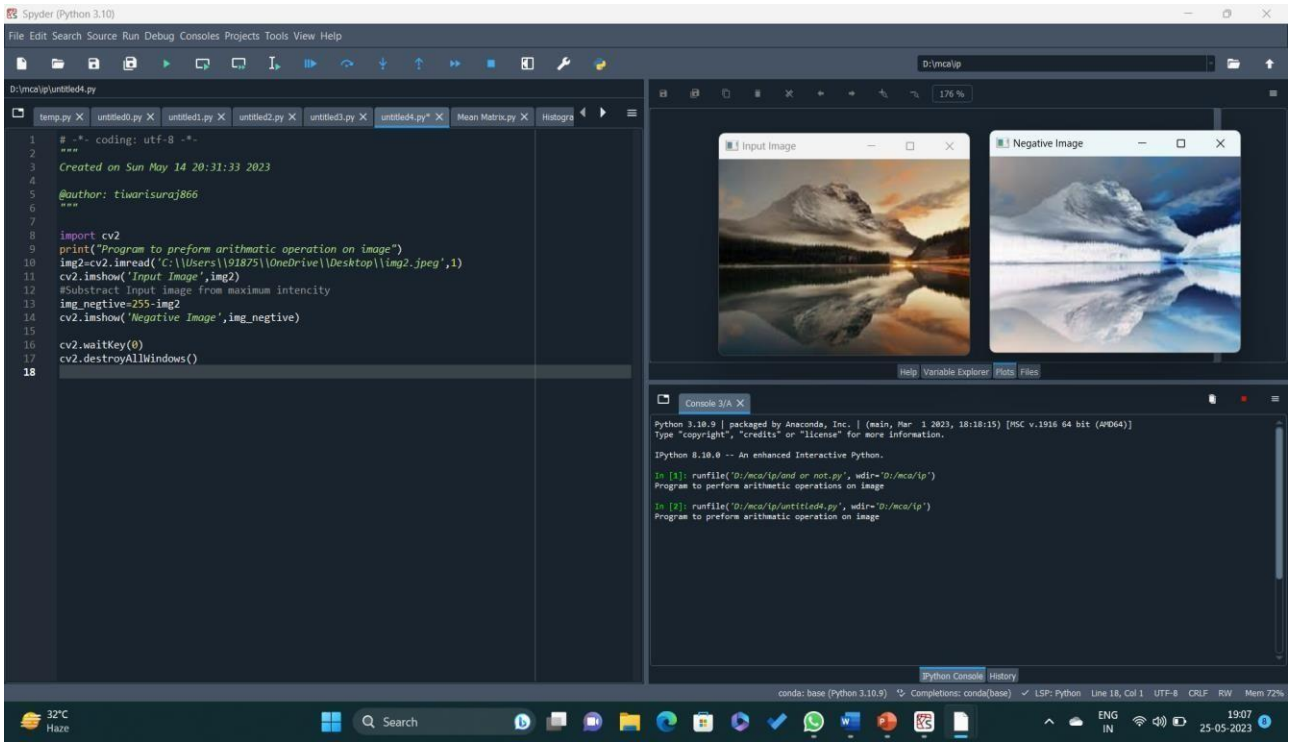
Thus, the transformation function used in image negative is

Where L-1 is the max intensity value and s, and r are the output and input pixel values respectively.

Source Code :

```
# -*- coding: utf-8 -*-
"""
Created on Sun May 14 20:31:33 2023
@author: OMKAR
"""
import cv2
print("Program to preform arithmatic operation on image")
img2=cv2.imread('C:\\Users\\91875\\OneDrive\\Desktop\\img2.jpeg',1
) cv2.imshow('Input Image',img2)
#Substract Input image from maximum intencity
img_negtive=255-img2 cv2.imshow('Negative
Image',img_negtive) cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:


Spyder (Python 3.10)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\mca\ip\untitled4.py
temp.py untitled0.py untitled1.py untitled2.py untitled3.py untitled4.py* Mean Matrix.py Histogra
-*- coding: utf-8 -*-
"""
Created on Sun May 14 20:31:33 2023
@author: tiwarisuraj866
"""
import cv2
print("Program to preform arithmatic operation on image")
img2=cv2.imread('C:\\Users\\91875\\OneDrive\\Desktop\\img2.jpeg',1)
cv2.imshow('Input Image',img2)
#Substract Input image from maximum intencity
img_negtive=255-img2
cv2.imshow('Negative Image',img_negtive)
cv2.waitKey(0)
cv2.destroyAllWindows()
Input Image
Negative Image
Help Variable Explorer Plots Files
Console 3/A
Python 3.10.9 | packaged by Anaconda, Inc. | (main, Mar 1 2023, 18:18:15) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
IPython 8.10.0 -- An enhanced Interactive Python.
In [1]: runfile('D:/mca/ip/and or not.py', wdir='D:/mca/ip')
Program to perform arithmetic operations on image
In [2]: runfile('D:/mca/ip/untitled4.py', wdir='D:/mca/ip')
Program to preform arithmatic operation on image
IPython Console History
conda: base (Python 3.10.9) Completions: conda(base) LSP: Python Line 18, Col 1 UTF-8 CRLF RW Mem 72%
32°C Haze
Search
ENG IN
19:07 25-05-2023

Practical 4

- **Title: Program to implement Thesholding of an image**

- **Thresholding** is a technique in OpenCV, which is the assignment of pixel values in relation to the threshold value provided. In thresholding, each pixel value is compared with the threshold value. If the pixel value is smaller than the threshold, it is set to 0, otherwise, it is set to a maximum value (generally 255). Thresholding is a very popular segmentation technique, used for separating an object considered as a foreground from its background. A threshold is a value which has two regions on its either side i.e. below the threshold or above the threshold.

In Computer Vision, this technique of thresholding is done on grayscale images. So initially, the image has to be converted in grayscale color space.

- If f (x, y) < T then f (x, y) = 0
- else f (x, y) = 255
- where f (x, y) = Coordinate Pixel Value
- T = Threshold Value.
- In OpenCV with Python, the function **cv2.threshold** is used for thresholding.
- ***Syntax:***
- *cv2.threshold(source, thresholdValue, maxVal, thresholdingTechnique)*
- ***Parameters:***

*-> **source**: Input Image array (must be in Grayscale).*
*-> **thresholdValue**: Value of Threshold below and above which pixel values will change accordingly.*
*-> **maxVal**: Maximum value that can be assigned to a pixel.*
*-> **thresholdingTechnique**: The type of thresholding to be applied.*

- The different Simple Thresholding Techniques are:

- **cv2.THRESH_BINARY**: If pixel intensity is greater than the set threshold, value set to 255, else set to 0 (black).
- **cv2.THRESH_BINARY_INV**: Inverted or Opposite case of cv2.THRESH_BINARY.
- **cv.THRESH_TRUNC**: If pixel intensity value is greater than threshold, it is truncated to the threshold. The pixel values are set to be the same as the threshold. All other values remain the same.
- **cv.THRESH_TOZERO**: Pixel intensity is set to 0, for all the pixels intensity, less than the threshold value. • **cv.THRESH_TOZERO_INV**: Inverted or Opposite case of cv2.THRESH_TOZERO.

Theory :

1. **Program to implement Thresholding of an Image.**

Thresholding is a technique in OpenCV, which is the assignment of pixel values in relation to the threshold value provided. In thresholding, each pixel value is compared with the threshold value. If the pixel value is smaller than the threshold, it is set to 0, otherwise, it is set to a maximum value (generally 255). Thresholding is a very popular segmentation technique, used for separating an object considered as a foreground from its background. A threshold is a value which has two regions on its either side i.e. below the threshold or above the threshold. In Computer Vision, this technique of thresholding is done on grayscale images. So initially, the image has to be converted in grayscale color space.

If f (x, y) < T then
  f (x, y) = 0
else
  f (x, y) = 255

where

f (x, y) = Coordinate Pixel Value T

= Threshold Value.

In OpenCV with Python, the function **cv2.threshold** is used for thresholding.

Syntax: *cv2.threshold(source, thresholdValue, maxVal, thresholdingTechnique)*

Parameters:

*-> **source**: Input Image array (must be in Grayscale).*

*-> **thresholdValue**: Value of Threshold below and above which pixel values will change accordingly.*

*-> **maxVal**: Maximum value that can be assigned to a pixel.*

*-> **thresholdingTechnique**: The type of thresholding to be applied.*

The different Simple Thresholding Techniques are:

cv2.THRESH_BINARY: If pixel intensity is greater than the set threshold, value set to 255, else set to 0 (black). **cv2.THRESH_BINARY_INV**: Inverted or Opposite case of cv2.THRESH_BINARY.

cv.THRESH_TRUNC: If pixel intensity value is greater than threshold, it is truncated to the threshold. The pixel values are set to be the same as the threshold. All other values remain the same.

cv.THRESH_TOZERO: Pixel intensity is set to 0, for all the pixels intensity, less than the threshold value.

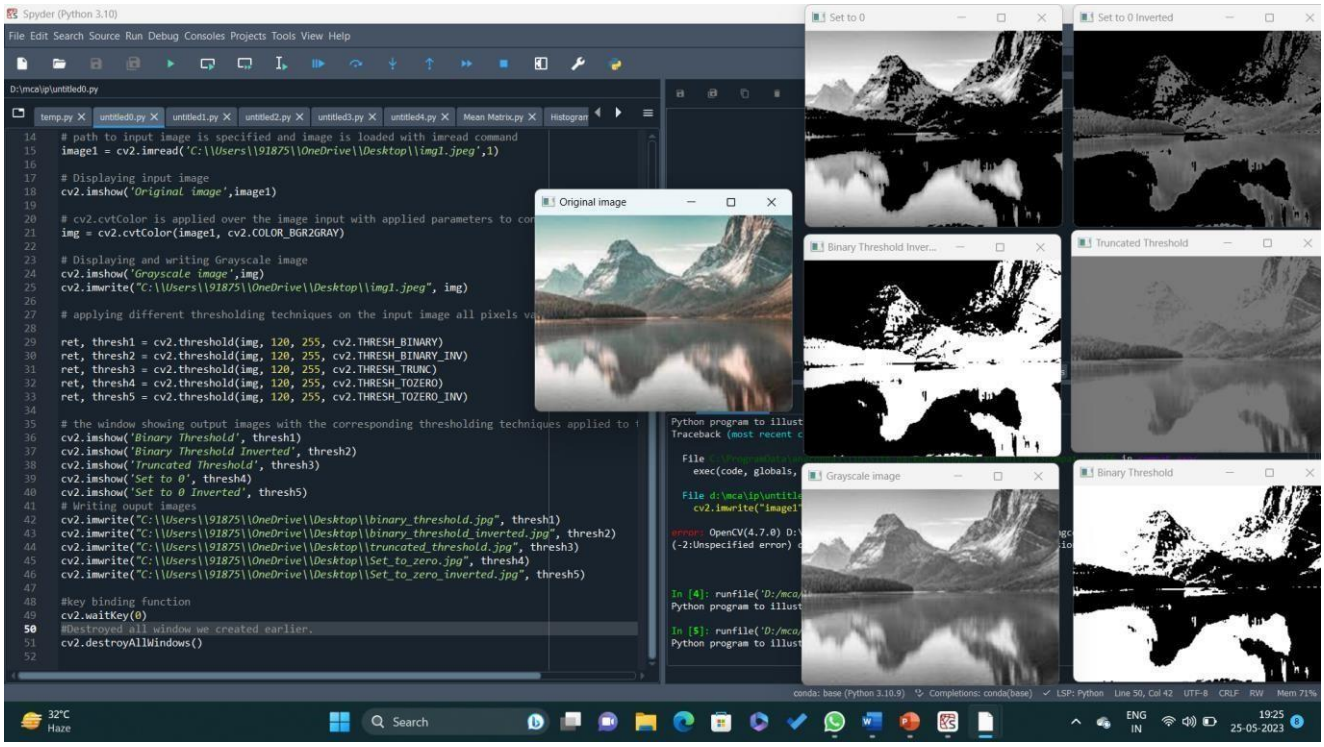
cv.THRESH_TOZERO_INV: Inverted or Opposite case of cv2.THRESH_TOZERO.

Source Code :

```
# -*- coding: utf-8 -*-
"""
Created on Sun May 14 20:05:36 2023

@author: OMKAR
"""

# organizing imports import
cv2
import numpy as np
print("Python program to illustrate simple thesholding type on image")

# path to input image is specified and image is loaded with imread command image1
= cv2.imread('C:\\Users\\91875\\OneDrive\\Desktop\\img1.jpeg',1)

# Displaying input image
cv2.imshow('Original image',image1)

# cv2.cvtColor is applied over the image input with applied parameters to convert the
image in grayscale img = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)
# Displaying and writing Grayscale image cv2.imshow('Grayscale
image',img)
cv2.imwrite("C:\\Users\\91875\\OneDrive\\Desktop\\img1.jpeg", img)
# applying different thresholding techniques on the input image all pixels value above
120 will be set to 255 ret, thresh1 = cv2.threshold(img, 120, 255,
cv2.THRESH_BINARY) ret, thresh2 = cv2.threshold(img, 120, 255,
cv2.THRESH_BINARY_INV) ret, thresh3 = cv2.threshold(img, 120, 255,
cv2.THRESH_TRUNC) ret, thresh4 = cv2.threshold(img, 120, 255,
cv2.THRESH_TOZERO) ret, thresh5 = cv2.threshold(img, 120, 255,
cv2.THRESH_TOZERO_INV) # the window showing output images with
the corresponding thresholding techniques applied to the input images
```

```
cv2.imshow('Binary Threshold', thresh1) cv2.imshow('Binary Threshold
Inverted', thresh2) cv2.imshow('Truncated Threshold', thresh3)
cv2.imshow('Set to 0', thresh4) cv2.imshow('Set to 0 Inverted', thresh5)
# Writing ouput images
cv2.imwrite("C:\\Users\\91875\\OneDrive\\Desktop\\binary_threshold.jpg", thresh1)
cv2.imwrite("C:\\Users\\91875\\OneDrive\\Desktop\\binary_threshold_inverted.jpg
", thresh2)
cv2.imwrite("C:\\Users\\91875\\OneDrive\\Desktop\\truncated_threshold.jpg",
thresh3) cv2.imwrite("C:\\Users\\91875\\OneDrive\\Desktop\\Set_to_zero.jpg",
thresh4)
cv2.imwrite("C:\\Users\\91875\\OneDrive\\Desktop\\Set_to_zero_inverted.jpg",
thresh5)
#key binding function
cv2.waitKey(0)
#Destroyed all window we created earlier.
cv2.destroyAllWindows()
```

Output :


Spyder (Python 3.10)
Original image
Set to 0
Set to 0 Inverted
Binary Threshold Inver...
Truncated Threshold
Grayscale image
Binary Threshold

Practical 5

• Title: Program to implement smoothing or averaging filter in spatial domain.

- The mean filter is an example of a linear filter. It basically replaces each pixel in the output image with the mean (average) value of the neighborhood. This has the effect of smoothing the image (reducing the amount of intensity variations between a pixel and the next), removing noise from the image, and brightening the image.
- Thus, in mean filtering, each pixel of the image will be replaced with the mean value of its neighbors, including the pixel itself. The 3x3 kernel used for mean filtering is as shown in the figure below, although other kernel sizes could be used (i.e. 5x5):

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

- What the above kernel is actually trying to tell us is that we sum all the elements under the kernel and take the mean (average) of the total. • An important point to mention here is that all the elements of the mean kernel should:
  - sum to 1
  - be the same

Theory:

Program to Implement smoothing or averaging filter in spatial domain.

Spatial Filtering technique is used directly on pixels of an image. Mask is usually considered to be added in size so that it has a specific center pixel. This mask is moved on the image such that the center of the mask traverses all image pixels.

Neighborhood processing in spatial domain: Here, to modify one pixel, we consider values of the immediate neighboring pixels also. For this purpose, 3X3, 5X5, or 7X7 neighborhood mask can be considered. An example of a 3X3 mask is shown below. f(x-1, y-1) f(x-1, y) f(x-1, y+1) f(x, y-1) f(x, y) f(x, y+1) f(x+1, y-1) f(x+1, y) f(x+1, y+1)

Low Pass filtering: It is also known as the smoothing filter. It removes the high-frequency content from the image. It is also used to blur an image. A low pass averaging filter mask is as shown.

1/9 1/9 1/9

1/9 1/9 1/9

1/9 1/9 1/9

High Pass Filtering: It eliminates low-frequency regions while retaining or enhancing the high-frequency components. A high pass filtering mask is as shown.

-1/9 -1/9 -1/9

-1/9 8/9 -1/9

-1/9 -1/9 -1/9

Median Filtering: It is also known as nonlinear filtering. It is used to eliminate salt and pepper noise. Here the pixel value is replaced by the median value of the neighboring pixel.

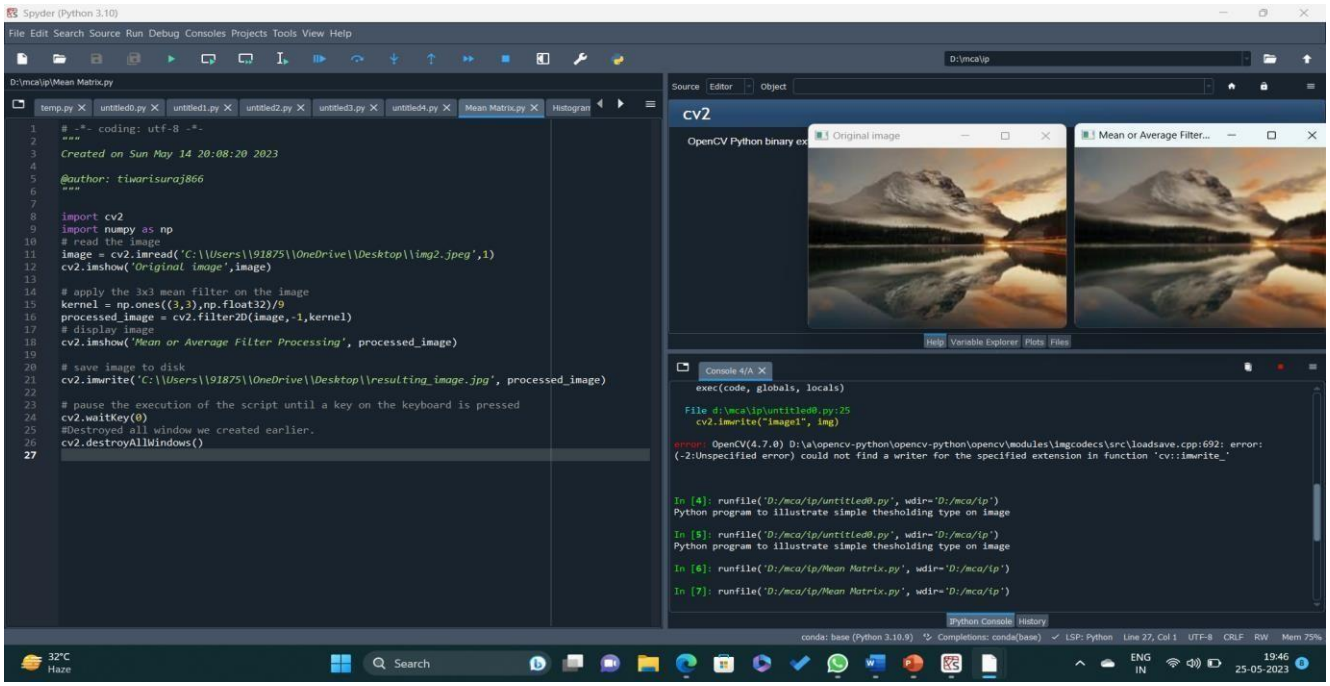
Source Code :

```
# -*- coding: utf-8 -*-
"""
Created on Sun May 14 20:08:20 2023

@author: OMKAR
""" import
cv2

import numpy as np # read the image image =
cv2.imread('C:\\Users\\91875\\OneDrive\\Desktop\\img2.jpeg',1)
cv2.imshow('Original image',image) # apply the 3x3 mean filter on the
image kernel = np.ones((3,3),np.float32)/9 processed_image =
cv2.filter2D(image,-1,kernel)
# display image cv2.imshow('Mean or Average Filter Processing',
processed_image)
# save image to disk
cv2.imwrite("C:\\Users\\91875\\OneDrive\\Desktop\\resulting_image.jpg", processed_image)
# pause the execution of the script until a key on the keyboard is pressed
cv2.waitKey(0)
```

#Destroyed all window we created earlier.

cv2.destroyAllWindows()

Output :



Practical 6

- Title: Program to produce the Histogram, Equalized Histogram and Equalized image of an input image.

- Histogram is considered as a graph or plot which is related to frequency of pixels in an

  Gray Scale Image with pixel values (ranging from 0 to 255). Grayscale image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information where pixel value varies from 0 to 255. Images of this sort, also known as black-and-white, are composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest where Pixel can be considered as a every point in an image.

- we use cv2.calcHist()(in-built function in OpenCV) to find the histogram.

  cv2.calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate]])

- **images :** it is the source image of type uint8 or float32 represented as "[img]". **channels :** it is the index of channel for which we calculate histogram. For grayscale image, its value is [0] and color image, you can pass [0], [1] or [2] to calculate histogram of blue, green or red channel respectively. **mask :** mask image. To find histogram of full image, it is given as "None". **histSize :** this represents our BIN count. For full scale, we pass [256]. **ranges** : this is our RANGE. Normally, it is [0,256].

Theory :

Program to produce the Histogram, Equalized Histogram, and Equalized image of an input image.

Histogram equalization is a method in image processing of contrast adjustment using the image's histogram.

This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher

contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values. The method is useful in images with backgrounds and foregrounds that are both bright or both dark.

OpenCV has a function to do this, **cv2.equalizeHist()**. Its input is just grayscale image and output is our histogram equalized image.

The histogram of a digital image, with intensity levels between 0 and (L-1), is a function $h(r_k) = n_k$, where r_k is the kth intensity level and n_k is the number of pixels in the image having that intensity level. We can also normalize the histogram by dividing it by the total number of pixels in the image. For an N x N image, we have the following definition of a normalized histogram function:

This $p(r_k)$ function is the probability of the occurrence of a pixel with the intensity level r_k. Clearly,

The histogram of an image, as shown in the figure, consists of the x-axis representing the intensity levels r_k and the y-axis denoting the $h(r_k)$ or the $p(r_k)$ functions.

The histogram of an image gives important information about the grayscale and contrast of the image. If the entire histogram of an image is centered towards the left end of the x-axis, then it implies a dark image. If the histogram is more inclined towards the right end, it signifies a white or bright image. A narrow-width histogram plot at the center of the intensity axis shows a low-contrast image, as it has a few levels of grayscale. On the other hand, an evenly distributed histogram over the entire x-axis gives a high-contrast effect to the image.

In image processing, there frequently arises the need to improve the contrast of the image. In such cases, we use an intensity transformation technique known as histogram equalization. Histogram equalization is the process of uniformly distributing the image histogram over the entire intensity axis by choosing a proper intensity transformation function. Hence, histogram equalization is an intensity transformation process.

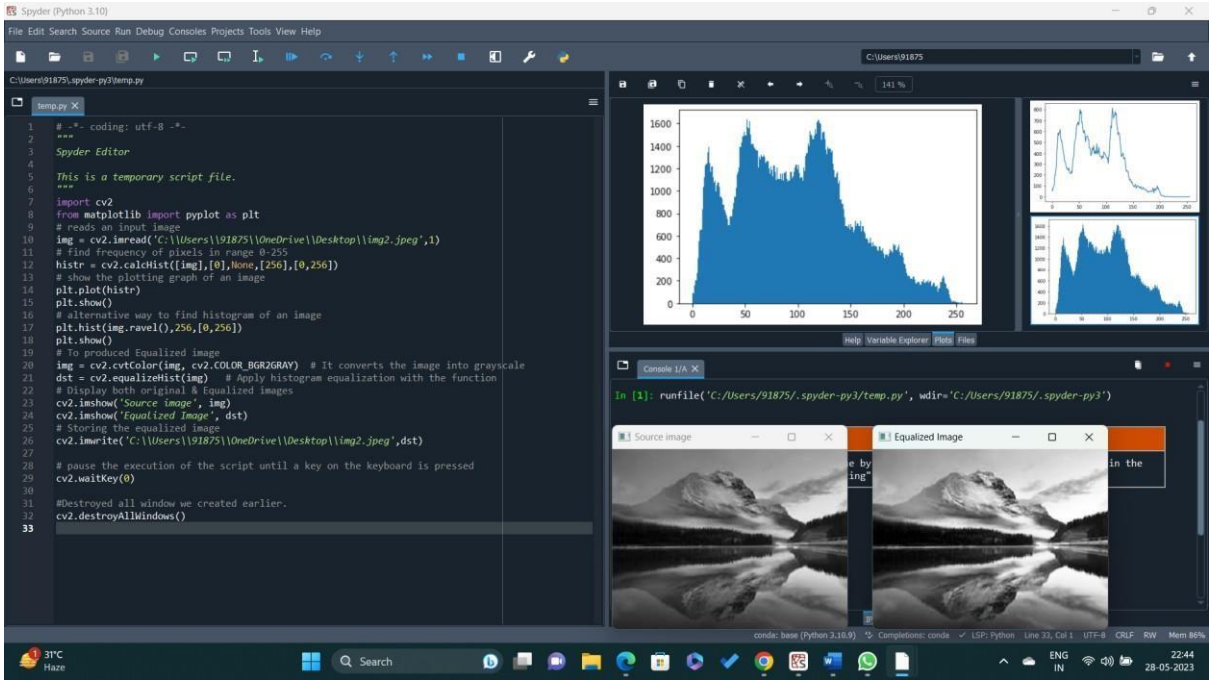
The choice of the ideal transformation function for uniform distribution of the image histogram is mathematically explained below.

Source Code :

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
""" import
cv2

from matplotlib import pyplot as plt
# reads an input image img =
cv2.imread('C:\\Users\\91875\\OneDrive\\Desktop\\img2.jpeg',1)
# find frequency of pixels in range 0-255 histr =
cv2.calcHist([img],[0],None,[256],[0,256])
# show the plotting graph of an image
plt.plot(histr) plt.show()

# alternative way to find histogram of an image
plt.hist(img.ravel(),256,[0,256]) plt.show()

# To produced Equalized image
```

```
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # It converts the image into grayscale

dst = cv2.equalizeHist(img) # Apply histogram equalization with the function

# Display both original & Equalized images


cv2.imshow('Source image', img)

cv2.imshow('Equalized Image', dst) #

Storing the equalized image

cv2.imwrite('C:\\Users\\91875\\OneDrive\\Desktop\\img2.jpeg',dst)


# pause the execution of the script until a key on the keyboard is pressed

cv2.waitKey(0)


#Destroyed all window we created earlier.

cv2.destroyAllWindows()
```

Output :


Spyder (Python 3.10)
File Edit Search Source Run Debug Consoles Projects Tools View Help
temp.py
-*- coding: utf-8 -*-
"""
Spyder Editor
This is a temporary script file.
"""
import cv2
from matplotlib import pyplot as plt
# reads an input image
img = cv2.imread('C:\\Users\\91875\\OneDrive\\Desktop\\img2.jpeg',1)
# find frequency of pixels in range 0-255
histr = cv2.calcHist([img],[0],None,[256],[0,256])
# show the plotting graph of an image
plt.plot(histr)
plt.show()
# alternative way to find histogram of an image
plt.hist(img.ravel(),256,[0,256])
plt.show()
# To produced Equalized image
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # It converts the image into grayscale
dst = cv2.equalizeHist(img) # Apply histogram equalization with the function
# Display both original & Equalized images
cv2.imshow('Source image', img)
cv2.imshow('Equalized Image', dst)
# Storing the equalized image
cv2.imwrite('C:\\Users\\91875\\OneDrive\\Desktop\\img2.jpeg',dst)
# pause the execution of the script until a key on the keyboard is pressed
cv2.waitKey(0)
#Destroyed all window we created earlier.
cv2.destroyAllWindows()
Console 1/A
In [1]: runfile('C:/Users/91875/.spyder-py3/temp.py', wdir='C:/Users/91875/.spyder-py3')
Source image
Equalized Image
31°C
Haze
Search
ENG
IN
22:44
28-05-2023

Practical 7

• Title: Program for smooth an image using low pass filter in frequency domain.

• As in one-dimensional signals, images also can be filtered with various low-pass filters(LPF), high-pass filters(HPF) etc. LPF helps in removing noises, blurring the images etc. • OpenCV provides a function **cv2.filter2D()** to convolve a kernel with an image.

Theory :

Program for smooth an image using low pass filter in frequency domain.

Low pass filter removes the high frequency components that means it keeps low frequency components. It is used for smoothing the image. It is used to smoothen the image by attenuating high frequency components and preserving low frequency components.

G(u, v) = H(u, v) . F(u, v)

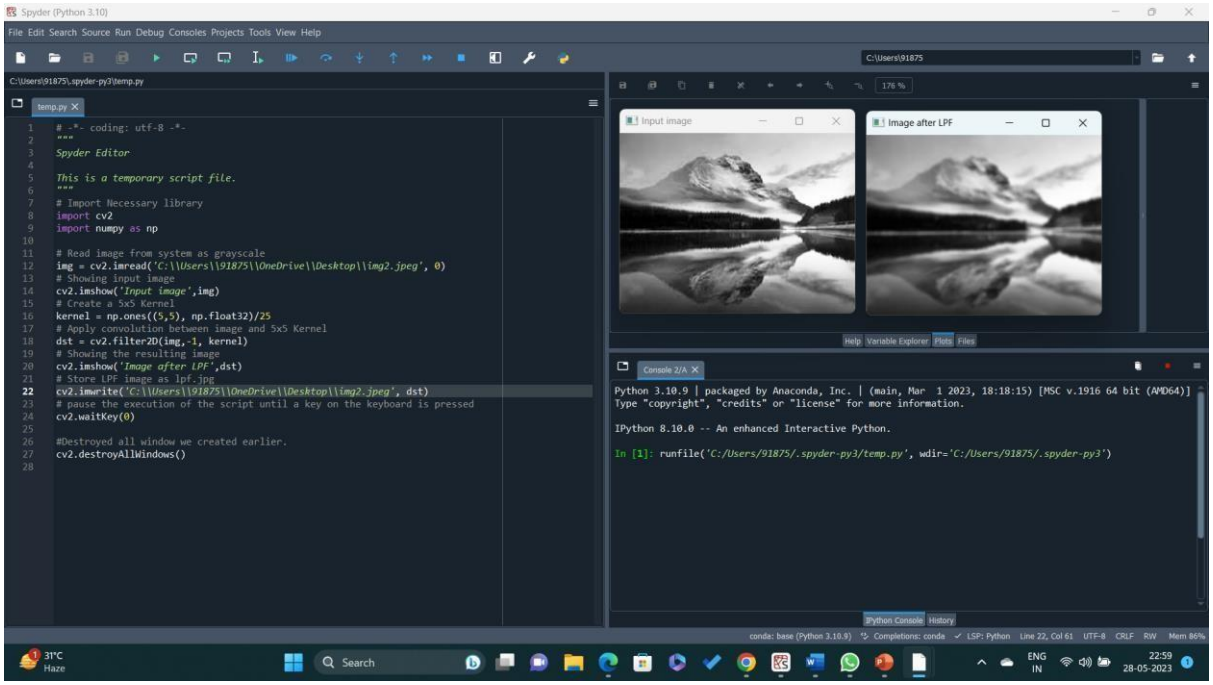
where F(u, v) is the Fourier Transform of original image and

H(u, v) is the Fourier Transform of filtering mask .

Source Code :

```
# -*- coding: utf-8 -*-

# Import Necessary library

import cv2 import numpy

as np


# Read image from system as grayscale

img = cv2.imread('C:\\Users\\91875\\OneDrive\\Desktop\\img2.jpeg', 0)

# Showing input image

cv2.imshow('Input image',img) #

Create a 5x5 Kernel kernel =

np.ones((5,5), np.float32)/25


# Apply convolution between image and 5x5 Kernel dst =

cv2.filter2D(img,-1, kernel) # Showing the resulting image

cv2.imshow('Image after LPF',dst) # Store LPF image as lpf.jpg

cv2.imwrite('C:\\Users\\91875\\OneDrive\\Desktop\\img2.jpeg', dst)


# pause the execution of the script until a key on the keyboard is pressed

cv2.waitKey(0)

#Destroyed all window we created earlier.

cv2.destroyAllWindows()
```

Output :


Spyder (Python 3.10)
File Edit Search Source Run Debug Consoles Projects Tools View Help
temp.py
-*- coding: utf-8 -*-
"""
Spyder Editor
This is a temporary script file.
"""
# Import Necessary library
import cv2
import numpy as np
# Read image from system as grayscale
img = cv2.imread('C:\\Users\\91875\\OneDrive\\Desktop\\img2.jpeg', 0)
# Showing input image
cv2.imshow('Input image',img)
# Create a 5x5 Kernel
kernel = np.ones((5,5), np.float32)/25
# Apply convolution between image and 5x5 Kernel
dst = cv2.filter2D(img,-1, kernel)
# Showing the resulting image
cv2.imshow('Image after LPF',dst)
# Store LPF image as lpf.jpg
cv2.imwrite('C:\\Users\\91875\\OneDrive\\Desktop\\img2.jpeg', dst)
# pause the execution of the script until a key on the keyboard is pressed
cv2.waitKey(0)
#Destroyed all window we created earlier.
cv2.destroyAllWindows()
Input image
Image after LPF
Python 3.10.9 | packaged by Anaconda, Inc. | (main, Mar 1 2023, 18:18:15) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
IPython 8.10.0 -- An enhanced Interactive Python.
In [1]: runfile('C:/Users/91875/.spyder-py3/temp.py', wdir='C:/Users/91875/.spyder-py3')

Practical 8

- **Title: Program for smooth an image using high pass filter in frequency domain.**
- A **High Pass Filter(HPF)** main advantage is used to sharpen the image by attenuating the low frequency. When the impulse response or signal is passed through a high pass filter, an HPF mainly allows high frequencies to pass through. As High pass filters are used for sharpening the images, the frequency obtained is less compared to cut-off frequency(ωc). In OpenCV and in digital image processing we also use HPF functionality to find the edges in an image.
- The reason to blur the image is to add smoothening effect to an image. By blurring the image we can filter the unwanted noise from the image. In the OpenCV library, we widely use Gaussian Filter. It employs the technique "kernel convolution".
- **Note:** 127 is added after subtracting the image with a blurred image to add the greyish look. We shall use Gaussian Blur to blur the image.
- dst = cv2.GaussianBlur(src, ksize, sigmaX[, dst[, sigmaY[, borderType=BORDER_DEFAULT]]])
- hpf = img – cv2.GaussianBlur(img,(21,21),3)+127

Theory :

Program for sharpen the image using high pass filter in frequency domain

High pass filter removes the low frequency components that means it keeps high frequency components. It is used for sharpening the image. It is used to sharpen the image by attenuating low frequency components and preserving high frequency components.

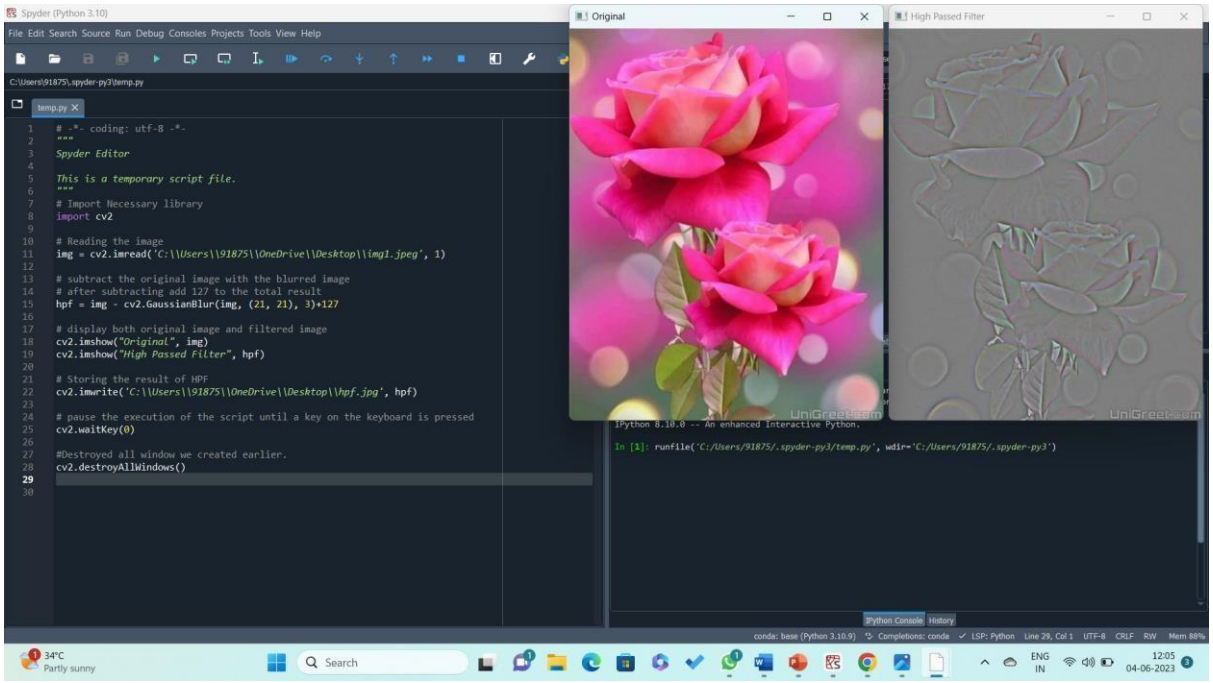
Mechanism of high pass filtering in frequency domain is given by:

H(u, v) = 1 - H'(u, v) where H(u, v) is the Fourier Transform of high pass filtering and H'(u, v) is the Fourier Transform of low pass filtering

Source Code :

```
# -*- coding: utf-8 -*-
""" Spyder
Editor

This is a temporary script file.
"""
# Import Necessary library
import cv2 # Reading the
image

img = cv2.imread('C:\\Users\\91875\\OneDrive\\Desktop\\img1.jpeg', 1)
# subtract the original image with the blurred image #
after subtracting add 127 to the total result

hpf = img - cv2.GaussianBlur(img, (21, 21), 3)+127

# display both original image and filtered image cv2.imshow("Original",
img) cv2.imshow("High Passed Filter", hpf) # Storing the result of HPF
cv2.imwrite('C:\\Users\\91875\\OneDrive\\Desktop\\hpf.jpg', hpf) #
pause the execution of the script until a key on the keyboard is pressed
cv2.waitKey(0)
#Destroyed all window we created earlier.
cv2.destroyAllWindows()
```

Output :


Original
High Passed Filter
-*- coding: utf-8 -*-
"""
Spyder Editor
This is a temporary script file.
"""
# Import Necessary library
import cv2
# Reading the image
img = cv2.imread('C:\\Users\\91875\\OneDrive\\Desktop\\img1.jpeg', 1)
# subtract the original image with the blurred image
# after subtracting add 127 to the total result
hpf = img - cv2.GaussianBlur(img, (21, 21), 3)+127
# display both original image and filtered image
cv2.imshow("Original", img)
cv2.imshow("High Passed Filter", hpf)
# Storing the result of HPF
cv2.imwrite('C:\\Users\\91875\\OneDrive\\Desktop\\hpf.jpg', hpf)
# pause the execution of the script until a key on the keyboard is pressed
cv2.waitKey(0)
#Destroyed all window we created earlier.
cv2.destroyAllWindows()
In [1]: runfile('C:/Users/91875/.spyder-py3/temp.py', wdir='C:/Users/91875/.spyder-py3')

Practical 9

- **Title: Program to find DFT/FFT Forward and Inverse Transform of Image**
- Fast Fourier Transform (FFT) to transform image to frequency domain.
- Digital images are not continuous so we use Fast Fourier Transform (DFT) instead of Fourier transform.

Theory :

Program to find DFT/FFT forward and Inverse Transform of Image.

Fourier Transform is a mathematical technique that helps to transform Time Domain function x(t) to Frequency Domain function X(ω). In this article, we will see how to find Fourier Transform in MATLAB.

Properties of Fourier Transform:

Linearity: The addition of two functions corresponding to the addition of the two frequency spectrum is called linearity. If we multiply a function by a constant, the Fourier transform of the resultant function is multiplied by the same constant. The Fourier transform of the sum of two or more functions is the sum of the Fourier transforms of the functions.

Scaling: Scaling is the method that is used to change the range of the independent variables or features of data. If we stretch a function by the factor in the time domain then squeeze the Fourier transform by the same factor in the frequency domain.

Differentiation: Differentiating function with respect to time yields to the constant multiple of the initial function.

Convolution: It includes the multiplication of two functions. The Fourier transform of a convolution of two functions is the point-wise product of their respective Fourier transforms.

Frequency Shift and Time Shift: Frequency is shifted according to the coordinates. There is a duality between the time and frequency domains and frequency shift affects the time shift. The time variable shift also affects the frequency function. The time-

shifting property concludes that a linear displacement in time corresponds to a linear phase factor in the frequency domain.

Equation for DFT: $X(k) = \sum_{n=0}^{N-1} x[n].e^{\frac{-j2\pi kn}{N}}$

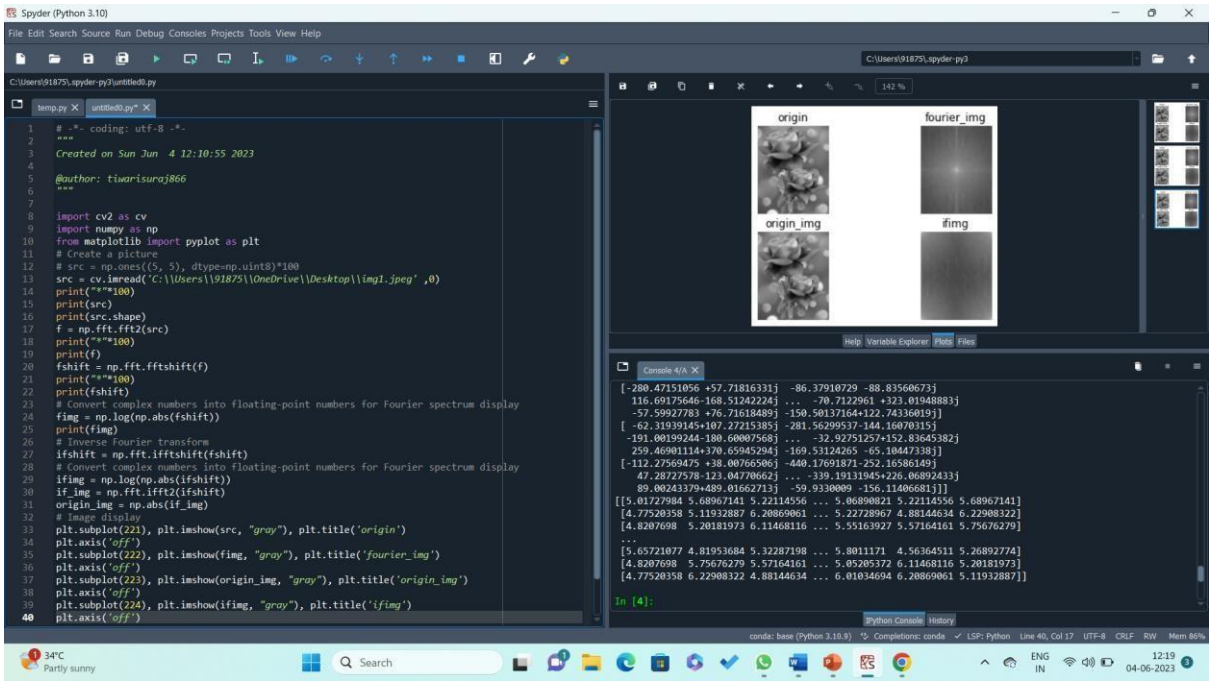
Equation for IDFT: $x(n) = \sum_{k=0}^{N-1} X[k].e^{\frac{j2\pi kn}{N}}$

Source Code :

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jun 4 12:10:55 2023
@author: OMKAR
"""
import cv2 as cv import numpy as
np from matplotlib import pyplot as
plt
# Create a picture
# src = np.ones((5, 5), dtype=np.uint8)*100
src = cv.imread('C:\\Users\\91875\\OneDrive\\Desktop\\img1.jpeg' ,0)
print("*"*100) print(src) print(src.shape) f = np.fft.fft2(src)
print("*"*100) print(f) fshift = np.fft.fftshift(f) print("*"*100)
print(fshift)
# Convert complex numbers into floating-point numbers for Fourier spectrum display fimg
= np.log(np.abs(fshift))

print(fimg)
# Inverse Fourier transform ifshift
= np.fft.ifftshift(fshift)
```

```
# Convert complex numbers into floating-point numbers for Fourier spectrum display

ifimg = np.log(np.abs(ifshift)) if_img = np.fft.ifft2(ifshift) origin_img =

np.abs(if_img)

# Image display plt.subplot(221), plt.imshow(src, "gray"),

plt.title('origin') plt.axis('off') plt.subplot(222), plt.imshow(fimg,

"gray"), plt.title('fourier_img') plt.axis('off') plt.subplot(223),

plt.imshow(origin_img, "gray"), plt.title('origin_img') plt.axis('off')

plt.subplot(224), plt.imshow(ifimg, "gray"), plt.title('ifimg')

plt.axis('off')

plt.show()
```

Output :


Spyder (Python 3.10)
origin
fourier_img
origin_img
ifimg
Search
12:19
04-06-2023

Practical 10

- **Title: Program to find DCT forward and Inverse Transform of Image.**
- Discrete Cosine Transform is used in lossy image compression because it has very strong energy compaction, i.e., its large amount of information is stored in very low frequency component of a signal and rest other frequency having very small data which can be stored by using very less number of bits (usually, at most 2 or 3 bit).

Theory :

Program to find DCT forward and Inverse Transform of Image.

Discrete Cosine Transform is used in lossy image compression because it has very strong energy compaction, i.e., its large amount of information is stored in very low frequency component of a signal and rest other frequency having very small data which can be stored by using very less number of bits (usually, at most 2 or 3 bit).

To perform DCT Transformation on an image, first we have to fetch image file information (pixel value in term of integer having range 0 – 255) which we divides in block of 8 X 8 matrix and then we apply discrete cosine transform on that block of data.

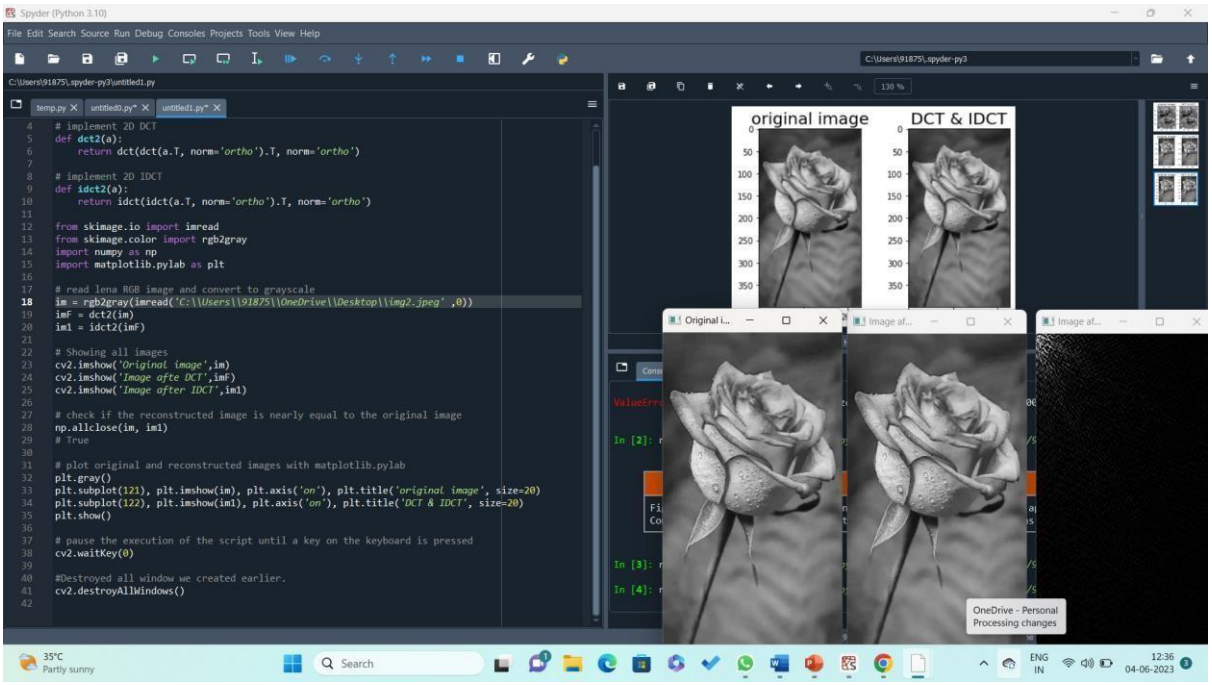
After applying discrete cosine transform, we will see that its more than 90% data will be in lower frequency component. For simplicity, we took a matrix of size 8 X 8 having all value as 255 (considering image to be completely white) and we are going to perform 2-D discrete cosine transform on that to observe the output.

Source Code :

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jun 4 12:10:55 2023
@author: OMKAR
"""
from scipy.fftpack import dct, idct
import cv2 # implement 2D DCT
def dct2(a):
    return dct(dct(a.T, norm='ortho').T, norm='ortho')
# implement 2D IDCT
def idct2(a):
    return idct(idct(a.T, norm='ortho').T, norm='ortho')
from skimage.io import imread from skimage.color
import rgb2gray import numpy as np import
matplotlib.pylab as plt

# read lena RGB image and convert to grayscale
im = rgb2gray(imread('C:\\Users\\91875\\OneDrive\\Desktop\\img2.jpeg' ,0))
imF = dct2(im) im1 = idct2(imF)
```

```
# Showing all images

cv2.imshow('Original       image',im)

cv2.imshow('Image  afte  DCT',imF)

cv2.imshow('Image after IDCT',im1)

# check if the reconstructed image is nearly equal to the original image np.allclose(im,

im1)

# True

# plot original and reconstructed images with matplotlib.pylab plt.gray()

plt.subplot(121), plt.imshow(im), plt.axis('on'), plt.title('original image', size=20)

plt.subplot(122), plt.imshow(im1), plt.axis('on'), plt.title('DCT & IDCT',

size=20) plt.show()

# pause the execution of the script until a key on the keyboard is pressed

cv2.waitKey(0)

#Destroyed all window we created earlier.

cv2.destroyAllWindows()
```

Output :


original image
DCT & IDCT
OneDrive - Personal
Processing changes

Practical 11

- **Title: Program to find Edges using Prewit/Sobel/Fri-chen/Robert operators.**
- Edge detection involves mathematical methods to find points in an image where the brightness of pixel intensities changes distinctly.
- cv2.Sobel(original_image,ddepth,xorder,yorder,kernelsize)

Theory :

Program to find Edges using Prewit/ Sobel/ Fri-chen / Robert operators.

The Prewitt operator was developed by Judith M. S. Prewitt. Prewitt operator is used for edge detection in an image. Prewitt operator detects both types of edges, these are:
Horizontal edges or along the x-axis, Vertical
Edges or along the y-axis.
Wherever there is a sudden change in pixel intensities, an edge is detected by the mask. Since the edge is defined as the change in pixel intensities, it can be calculated by using differentiation. Prewitt mask is a first-order derivative mask. In the graph representation of Prewitt-mask's result, the edge is represented by the local maxima or local minima.

Both the first and second derivative masks follow these three properties:
More weight means more edge detection.
The opposite sign should be present in the mask. (+ and -) The
Sum of the mask values must be equal to zero.
Prewitt operator provides us two masks one for detecting edges in the horizontal direction and another for detecting edges in a vertical direction.
Prewitt Operator [X-axis] = [-1 0 1; -1 0 1; -1 0 1]
Prewitt Operator [Y-axis] = [-1 -1 -1; 0 0 0; 1 1 1]
It is named after Irwin Sobel and Gary Feldman. Like the Prewitt operator Sobel operator is also used to detect two kinds of edges in an image:

1. Vertical direction
2. Horizontal direction

The difference between Sobel and Prewitt Operator is that in Sobel operator the coefficients of masks are adjustable according to our requirement provided they follow all properties of derivative masks.

Sobel-X Operator = [-1 0 1; -2 0 2; -1 0 1]

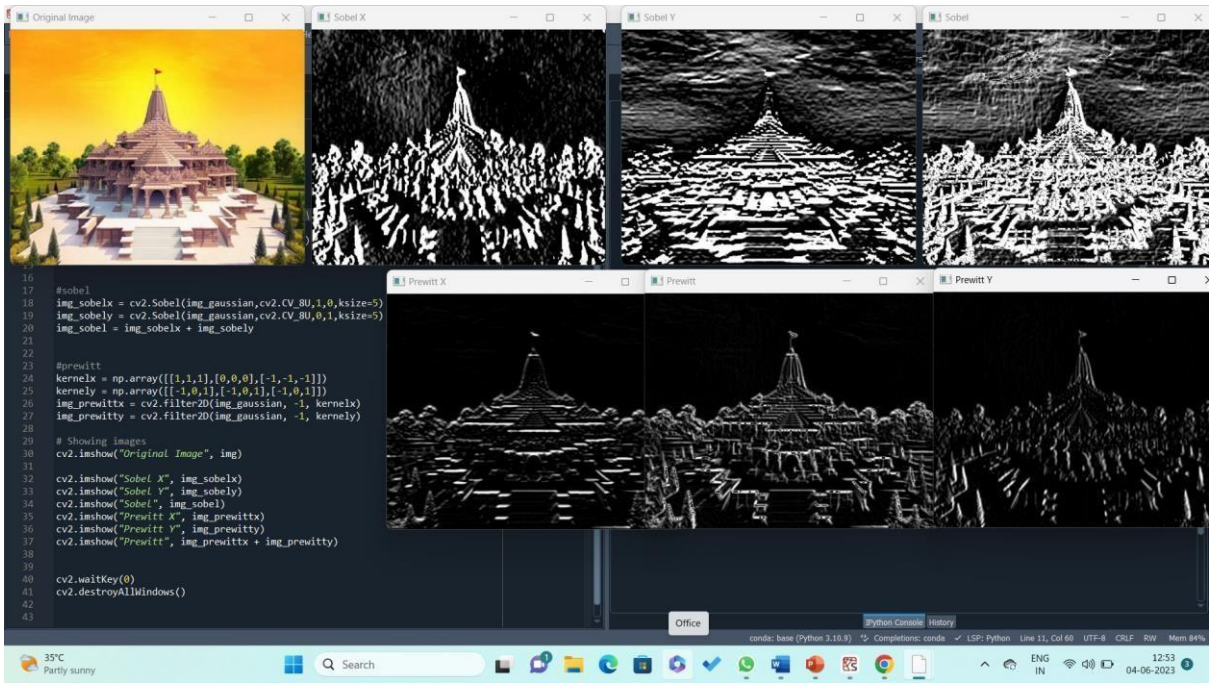
Sobel-Y Operator = [-1 -2 -1; 0 0 0; 1 2 1]

Source Code :

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jun 4 12:10:55 2023
@author: OMKAR
"""
import cv2
import numpy as np
img = cv2.imread('C:\\Users\\91875\\OneDrive\\Desktop\\img3.jpeg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) img_gaussian
= cv2.GaussianBlur(gray,(3,3),0)
#sobel img_sobelx =
cv2.Sobel(img_gaussian,cv2.CV_8U,1,0,ksize=5) img_sobely
= cv2.Sobel(img_gaussian,cv2.CV_8U,0,1,ksize=5) img_sobel
= img_sobelx + img_sobely
#prewitt
kernelx = np.array([[1,1,1],[0,0,0],[-1,-1,-1]])
kernely = np.array([[-1,0,1],[-1,0,1],[-1,0,1]]) img_prewittx =
cv2.filter2D(img_gaussian, -1, kernelx) img_prewitty =
cv2.filter2D(img_gaussian, -1, kernely)
#  Showing  images
cv2.imshow("Original Image", img)
cv2.imshow("Sobel X", img_sobelx)
cv2.imshow("Sobel Y", img_sobely)
```

```
cv2.imshow("Sobel", img_sobel)
cv2.imshow("Prewitt X", img_prewittx) cv2.imshow("Prewitt Y", img_prewitty)
cv2.imshow("Prewitt", img_prewittx + img_prewitty)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output :


Original Image
Sobel X
Sobel Y
Sobel
Prewitt X
Prewitt
Prewitt Y
#sobel
img_sobelx = cv2.Sobel(img_gaussian,cv2.CV_8U,1,0,ksize=5)
img_sobely = cv2.Sobel(img_gaussian,cv2.CV_8U,0,1,ksize=5)
img_sobel = img_sobelx + img_sobely
#prewitt
kernelx = np.array([[1,1,1],[0,0,0],[-1,-1,-1]])
kernely = np.array([[-1,0,1],[-1,0,1],[-1,0,1]])
img_prewittx = cv2.filter2D(img_gaussian, -1, kernelx)
img_prewitty = cv2.filter2D(img_gaussian, -1, kernely)
# Showing images
cv2.imshow("Original Image", img)
cv2.imshow("Sobel X", img_sobelx)
cv2.imshow("Sobel Y", img_sobely)
cv2.imshow("Sobel", img_sobel)
cv2.imshow("Prewitt X", img_prewittx)
cv2.imshow("Prewitt Y", img_prewitty)
cv2.imshow("Prewitt", img_prewittx + img_prewitty)
cv2.waitKey(0)
cv2.destroyAllWindows()
Office
IPython Console
History
Search

Practical 12

- **Title: Program to find edges using canny Edge Detection.**
- Canny edge detector is an edge detection operator that uses multi-stage algorithm to detect a wide range of edges in images. • The main stages are:
- Filtering out noise using Gaussian blur algorithm.
- Finding the strength and direction of edges using Sobel Filters.
- Isolating the strongest edges and thin them to one-pixel wide lines by applying non-maximum suppression.
- Using hysteresis to isolate the best edges.

Theory :

Program to find edges using canny Edge Detection.

Canny Edge detection is an Algorithm consisting of 4 major steps:

Reduce Noise using Gaussian Smoothing.

Compute image gradient using Sobel filter.

Apply Non-Max Suppression or NMS to just jeep the local maxima

Finally, apply Hysteresis thresholding which that 2 threshold values T_upper and T_lower which is used in the Canny() function.

Canny Edge filter in OpenCV. Canny() Function in OpenCV is used to detect the edges in an image.

Syntax: *cv2.Canny(image, T_lower, T_upper, aperture_size, L2Gradient)* ***Where:***

1. *Image: Input image to which Canny filter will be applied*
2. *T_lower: Lower threshold value in Hysteresis Thresholding 3.*

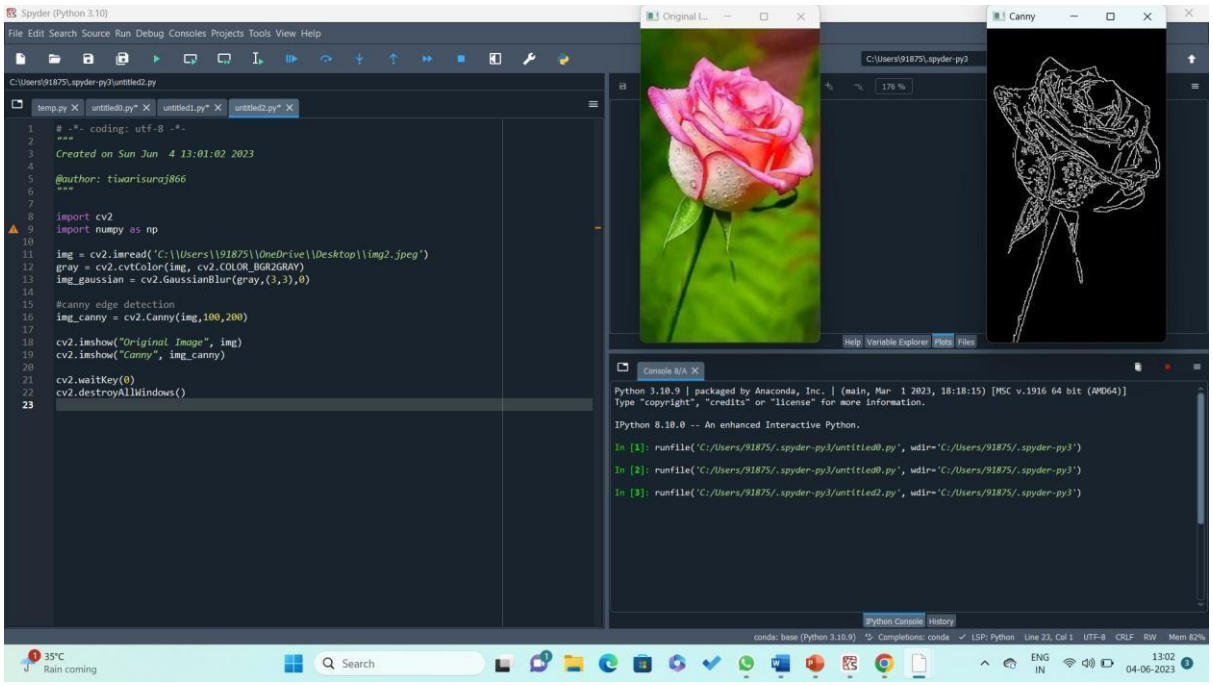
   T_upper: Upper threshold value in Hysteresis Thresholding

4. *aperture_size: Aperture size of the Sobel filter.*
5. *L2Gradient: Boolean parameter used for more precision in calculating Edge Gradient.*

Source Code :

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jun  4 13:01:02 2023

@author: OMKAR
"""

import cv2
import numpy as np

img = cv2.imread('C:\\Users\\91875\\OneDrive\\Desktop\\img2.jpeg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) img_gaussian
= cv2.GaussianBlur(gray,(3,3),0)

#canny edge detection
img_canny = cv2.Canny(img,100,200)

cv2.imshow("Original Image", img) cv2.imshow("Canny",
img_canny)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output :


Spyder (Python 3.10)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Original i...
Canny
-*- coding: utf-8 -*-
"""
Created on Sun Jun 4 13:01:02 2023
@author: tiwarisuraj866
"""
import cv2
import numpy as np
img = cv2.imread('C:\\Users\\91875\\OneDrive\\Desktop\\img2.jpeg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_gaussian = cv2.GaussianBlur(gray,(3,3),0)
#canny edge detection
img_canny = cv2.Canny(img,100,200)
cv2.imshow("Original Image", img)
cv2.imshow("Canny", img_canny)
cv2.waitKey(0)
cv2.destroyAllWindows()
Python 3.10.9 | packaged by Anaconda, Inc. | (main, Mar 1 2023, 18:18:15) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
IPython 8.10.0 -- An enhanced Interactive Python.
In [1]: runfile('C:/Users/91875/.spyder-py3/untitled0.py', wdir='C:/Users/91875/.spyder-py3')
In [2]: runfile('C:/Users/91875/.spyder-py3/untitled0.py', wdir='C:/Users/91875/.spyder-py3')
In [3]: runfile('C:/Users/91875/.spyder-py3/untitled2.py', wdir='C:/Users/91875/.spyder-py3')
Search

Practical 13

- **Title: Program to implement Huffmann coding technique for image compression.**
- Huffman Encoding is a Lossless Compression Algorithm used to compress the data. It is an algorithm developed by David A. Huffman. • Following are the two steps in Huffman Coding

1. Building Huffman Tree
2. Assigning codes to Leaf Nodes

Theory :

Program to implement Huffman coding technique for image compression.

Huffman coding is one of the basic compression methods, that have proven useful in image and video compression standards. When applying Huffman encoding technique on an Image, the source symbols can be either pixel intensities of the Image, or the output of an intensity mapping function.

The first step of Huffman coding technique is to reduce the input image to a ordered histogram, where the probability of occurrence of a certain pixel intensity value is as prob_pixel = numpix/totalnum

where **numpix** is the number of occurrence of a pixel with a certain intensity value and **totalnum** is the total number of pixels in the input Image.

Source Code :

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jun  4 13:10:04 2023

@author: OMKAR
"""

import re import
numpy as np from PIL
import Image import
cv2
print("Huffman Compression Program")
print("==========================================================
=
=========")
h = int(input("Enter 1 if you want to input an colour image file, 2 for default gray
scale case:")) if h == 1:
   file = input("Enter the filename:")
   my_string =
   np.asarray(Image.open(file),np.uint8) shape =
   my_string.shape a = my_string print ("Enetered
   string is:",my_string) my_string =
   str(my_string.tolist())
elif h == 2:
   array = np.arange(0, 737280, 1, np.uint8)
   my_string = np.reshape(array, (1024, 720))
   print ("Enetered string is:",my_string) a =
```

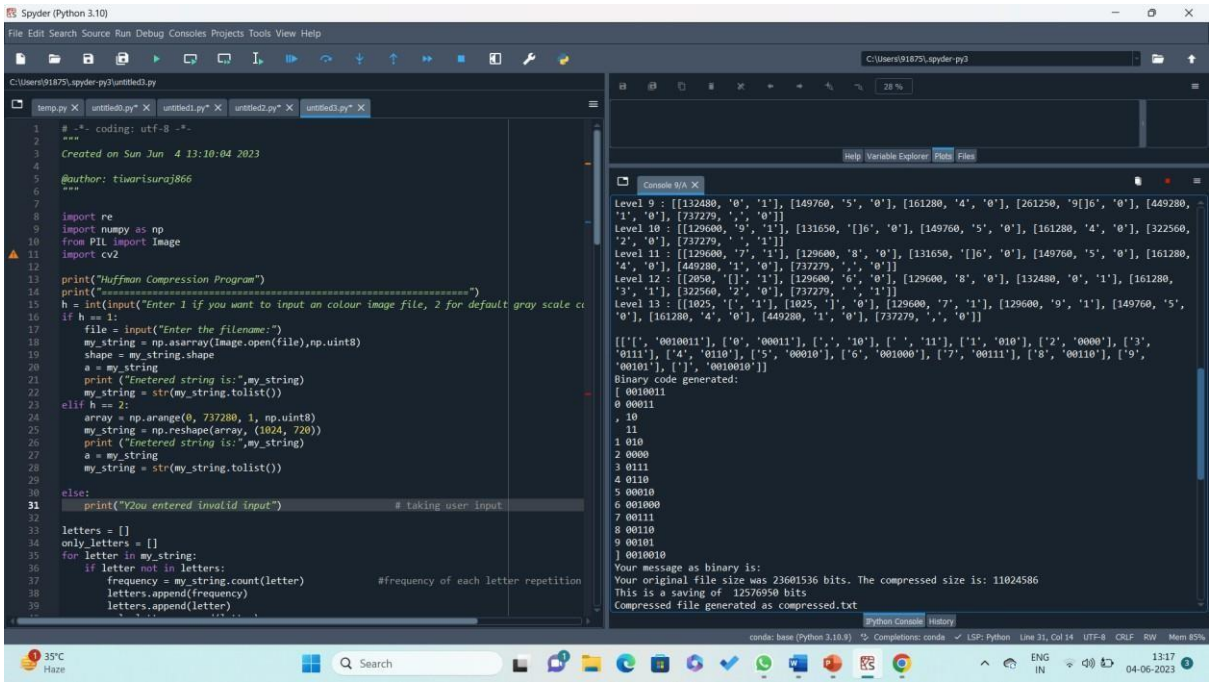
```
    my_string my_string =
    str(my_string.tolist())


else:
    print("You entered invalid input")                # taking user input


letters = [] only_letters
=    []    for    letter    in
my_string:
    if letter not in letters:
        frequency = my_string.count(letter)     #frequency of each letter repetition
        letters.append(frequency) letters.append(letter) only_letters.append(letter)


nodes = [] while len(letters)
> 0:
nodes.append(letters[0:2])
    letters = letters[2:]                        # sorting according to frequency
nodes.sort() huffman_tree
= []
huffman_tree.append(nodes) node                  #Make each unique character as a leaf


def combine_nodes(nodes):
    pos = 0 newnode
    = [] if len(nodes)
    > 1: nodes.sort()




        nodes[pos].append("1")                   # assigning values 1 and 0
        nodes[pos+1].append("0") combined_node1 =
        (nodes[pos] [0] + nodes[pos+1] [0])
        combined_node2 = (nodes[pos] [1] + nodes[pos+1] [1]) # combining the nodes to
generate pathways
        newnode.append(combined_node1)
        newnode.append(combined_node2)
```

```
            newnodes=[]
            newnodes.append(newnode)
            newnodes = newnodes + nodes[2:] nodes
            = newnodes
            huffman_tree.append(nodes)
            combine_nodes(nodes)
        return huffman_tree                    # huffman tree generation

    newnodes = combine_nodes(nodes)

    huffman_tree.sort(reverse = True) print("Huffman
    tree with merged pathways:")
checklist = [] for level in
    huffman_tree: for node in
    level:
        if node not in checklist:
            checklist.append(node)
        else:
            level.remove(node)
    count = 0 for level in
    huffman_tree:
        print("Level", count,":",level)      #print huffman tree
    count+=1 print()


    letter_binary = [] if
    len(only_letters) == 1:
        lettercode = [only_letters[0], "0"] letter_binary.append(letter_code*len(my_string))
    else:
        for letter in
            only_letters: code
            ="" for node in
            checklist:
                if len (node)>2 and letter in node[1]:          #genrating binary code
```

```
            code = code + node[2]
        lettercode =[letter,code] letter_binary.append(lettercode)
print(letter_binary)
print("Binary code generated:")
for letter in letter_binary:
   print(letter[0], letter[1])

bitstring ="" for character
in my_string:
   for item in letter_binary:
      if character in item:
         bitstring = bitstring + item[1]
binary ="0b"+bitstring print("Your
message as binary is:")
                              # binary code generated

uncompressed_file_size = len(my_string)*7 compressed_file_size = len(binary)-2
print("Your original file size was", uncompressed_file_size,"bits. The compressed size
is:",compressed_file_size) print("This is a saving of ",uncompressed_file_size-
compressed_file_size,"bits") output = open("compressed.txt","w+")
print("Compressed file generated as compressed.txt") output =
open("compressed.txt","w+") print("Decoding. ..... ")
output.write(bitstring)

bitstring = str(binary[2:])
uncompressed_string =""
code ="" for digit in
bitstring: code =
code+digit
   pos=0 #iterating and decoding for letter in letter_binary:
      if code ==letter[1]:
         uncompressed_string=uncompressed_string+letter_binary[pos] [0]
         code=""
```

```
        pos+=1

print("Your UNCOMPRESSED data is:") if
h == 1:
    temp = re.findall(r'\d+', uncompressed_string) res = list(map(int, temp)) res =
    np.array(res) res = res.astype(np.uint8) res = np.reshape(res, shape) print(res)
    print("Observe the shapes and input and output arrays are matching or not")
    print("Input image dimensions:",shape) print("Output image
    dimensions:",res.shape) data = Image.fromarray(res)
    data.save('C:\\Users\\IM160\\Desktop\\Python Practicals\\Practical
    13\huffman.jpg') if a.all() == res.all():
        print("Success")
if h == 2:
    temp = re.findall(r'\d+',
    uncompressed_string) res = list(map(int,
    temp)) print(res) res = np.array(res) res =
    res.astype(np.uint8) res = np.reshape(res,
    (1024, 720)) print(res)
    data = Image.fromarray(res)
    data.save('C:\\Users\\IM160\\Desktop\\Python Practicals\\Practical
    13\huffman.jpg') print("Success")
```

Output :


Spyder (Python 3.10)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\91875\.spyder-py3\untitled3.py
temp.py
untitled0.py*
untitled1.py*
untitled2.py*
untitled3.py*
-*- coding: utf-8 -*-
"""
Created on Sun Jun 4 13:10:04 2023
@author: tiwarisuraj866
"""
import re
import numpy as np
from PIL import Image
import cv2
print("Huffman Compression Program")
print("==")
h = int(input("Enter 1 if you want to input an colour image file, 2 for default gray scale c
if h == 1:
file = input("Enter the filename:")
my_string = np.asarray(Image.open(file),np.uint8)
shape = my_string.shape
a = my_string
print ("Enetered string is:",my_string)
my_string = str(my_string.tolist())
elif h == 2:
array = np.arange(0, 737280, 1, np.uint8)
my_string = np.reshape(array, (1024, 720))
print ("Enetered string is:",my_string)
a = my_string
my_string = str(my_string.tolist())
else:
print("Y2ou entered invalid input") # taking user input
letters = []
only_letters = []
for letter in my_string:
if letter not in letters:
frequency = my_string.count(letter) #frequency of each letter repetition
letters.append(frequency)
letters.append(letter)
Console 9/A
Level 9 : [[132480, '0', '1'], [149760, '5', '0'], [161280, '4', '0'], [261250, '9[]6', '0'], [449280, '1', '0'], [737279, ',', '0']]
Level 10 : [[129600, '9', '1'], [131650, '[]6', '0'], [149760, '5', '0'], [161280, '4', '0'], [322560, '2', '0'], [737279, ' ', '1']]
Level 11 : [[129600, '7', '1'], [129600, '8', '0'], [131650, '[]6', '0'], [149760, '5', '0'], [161280, '4', '0'], [449280, '1', '0'], [737279, ',', '0']]
Level 12 : [[2050, '[]', '1'], [129600, '6', '0'], [129600, '8', '0'], [132480, '0', '1'], [161280, '3', '1'], [322560, '2', '0'], [737279, ' ', '1']]
Level 13 : [[1025, '[', '1'], [1025, ']', '0'], [129600, '7', '1'], [129600, '9', '1'], [149760, '5', '0'], [161280, '4', '0'], [449280, '1', '0'], [737279, ',', '0']]
[['[', '0010011'], ['0', '00011'], [',', '10'], [' ', '11'], ['1', '010'], ['2', '0000'], ['3', '0111'], ['4', '0110'], ['5', '00010'], ['6', '001000'], ['7', '00111'], ['8', '00110'], ['9', '00101'], [']', '0010010']]
Binary code generated:
[0010011
0 00011
, 10
11
1 010
2 0000
3 0111
4 0110
5 00010
6 001000
7 00111
8 00110
9 00101
] 0010010
Your message as binary is:
Your original file size was 23601536 bits. The compressed size is: 11024586
This is a saving of 12576950 bits
Compressed file generated as compressed.txt
Python Console
History
conda: base (Python 3.10.9)
Completions: conda
LSP: Python
Line 31, Col 14
UTF-8
CRLF
RW
Mem 85%
35°C
Haze
Search
ENG
IN
13:17
04-06-2023