Indian Institute of Information Technology, Nagpur

DIGITAL IMAGE PROCESSING

Lab Report

Submitted By
Aniruddha Gawate
(BT18ECE025)

Under the Guidance of
Dr Tapan Kumar Jain
HOD
Dept of Electronics and Communication Engineering
IIIT, NAGPUR

Table of Contents

GITHUB Link - https://github.com/ani1004?tab=repositories

EXPERIMENT NO. 1

Aim - Basic Image Processing.

Code -

```
"""
Creator
@author:  BT18ECE025
          Aniruddha Gawate
"""
import cv2
import numpy as np
from matplotlib import pyplot as plt

# image resize
def resize(frame, scale=0.75):
    width = int(frame.shape[1] * scale)
    height = int(frame.shape[0] * scale)
    dim = (width, height)
    return cv2.resize(frame, dim, interpolation=cv2.INTER_AREA)

img = cv2.imread("image/t.jpg")
re_img = resize(img)
cv2.imshow('img_r', img)
cv2.imshow('img', re_img)
cv2.waitKey(0)

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
crop = img[50:200, 200:400]
half = cv2.resize(img, (0, 0), fx=0.1, fy=0.1)
bigger = cv2.resize(img, (1050, 1610))
stretch = cv2.resize(img, (780, 540), interpolation=cv2.INTER_NEAREST)

########################################
B, G, R = cv2.split(img)
# Corresponding channels are seperated

cv2.imshow("original", img)
cv2.imshow("blue", B)
cv2.imshow("Green", G)
cv2.imshow("red", R)
```

```
while True:
    k = cv2.waitKey(0) & 0xFF
    if k == 27:
        cv2.destroyAllWindows()
        break
#####################################
# full intensity to those pixel's R channel

img[50:150, :, 0] = 255
plt.figure(figsize=(10, 10))
plt.imshow(img)
plt.show()
############################
plt.imshow(img[:, :, 0])
plt.title('R channel')  # G=1 and B=2)
plt.imshow(img[:, :, 1])
plt.title('G channel')  # G=1 and B=2)
plt.imshow(img[:, :, 2])
plt.title('B channel')  # G=1 and B=2)
```

EXPERIMENT NO. 2

Aim - Bit Plane Slicing

Code -

```
"""
Creator
@author:  BT18ECE025
          Aniruddha Gawate
"""


import cv2
import numpy as np
import matplotlib.pyplot as plt
# Read the image in greyscale
a = cv2.imread("image/t.jpg", 0)

[m, n] = a.shape
b = []


def de2bin(v):
    d = list(format(v, '08b'))
    return d


for i in range(m):
    for j in range(n):
        bb = de2bin(a[i][j])  # function to convert to binary
        b.append(bb)

c = np.asarray(b)  # converting to array from list of binary


# distributing all the bit to diffrent arrays
r1 = np.array([int(i[0]) for i in c], dtype=np.uint8).reshape(m, n)
r2 = np.array([int(i[1]) for i in c], dtype=np.uint8).reshape(m, n)
r3 = np.array([int(i[2]) for i in c], dtype=np.uint8).reshape(m, n)
r4 = np.array([int(i[3]) for i in c], dtype=np.uint8).reshape(m, n)
r5 = np.array([int(i[4]) for i in c], dtype=np.uint8).reshape(m, n)
```

```
r6 = np.array([int(i[5]) for i in c], dtype=np.uint8).reshape(m, n)
r7 = np.array([int(i[6]) for i in c], dtype=np.uint8).reshape(m, n)
r8 = np.array([int(i[7]) for i in c], dtype=np.uint8).reshape(m, n)


# ploting 8 images
plt.figure(figsize=(20, 7))
plt.subplot(2, 4, 1)

plt.imshow(r1)
plt.title('MSB Bit Plane')
plt.subplot(2, 4, 2)

plt.imshow(r2)
plt.title('7nd Bit Plane')
plt.subplot(2, 4, 3)

plt.imshow(r3)
plt.title('6rd Bit Plane')
plt.subplot(2, 4, 4)

plt.imshow(r4)
plt.title('5th Bit Plane')
plt.subplot(2, 4, 5)

plt.imshow(r5)
plt.title('4th Bit Plane')
plt.subplot(2, 4, 6)

plt.imshow(r6)
plt.title('3th Bit Plane')
plt.subplot(2, 4, 7)

plt.imshow(r7)
plt.title('2th Bit Plane')
plt.subplot(2, 4, 8)

plt.imshow(r8)
plt.title('LSB Bit Plane')
```

EXPERIMENT NO. 3

Aim - Histogram Equalization.

Code -

```
"""
Creator
@author:  BT18ECE025
          Aniruddha Gawate
"""

from matplotlib import pyplot as plt
import numpy as np

h = np.array([[52, 55, 61, 59, 79, 61, 76, 61],
              [62, 59, 55, 104, 94, 85, 59, 71],
              [63, 65, 66, 113, 144, 104, 63, 72],
              [64, 70, 70, 126, 154, 109, 71, 69],
              [67, 73, 68, 106, 122, 88, 68, 68],
              [68, 79, 60, 70, 77, 66, 58, 75],
              [69, 85, 64, 58, 55, 61, 65, 83],
              [70, 87, 69, 68, 65, 73, 78, 90],
              ])

plt.imshow(h, cmap='gray')
plt.xticks([]), plt.yticks([])
plt.show()

# Counting the frequencies of the pixels
unique_elements, counts_elements = np.unique(h, return_counts=True)


# combining the elements and their freqs
hist = np.asarray((unique_elements, counts_elements))
print(hist)

# total of all freqs
cdf = hist[1][:].cumsum()
print(cdf.min(), len(cdf))

# equailization formula for scaling data from 0 to 255
```

```
b = []
for i in range(len(cdf)):
    eq = round(((cdf[i]-1)/63)*255)
    b.append(eq)

# change dtype to uint8
h_eq = np.uint8(np.asarray(b))

# matching hist values with equailization formula values
aa = {hist[0][i]: h_eq[i] for i in range(len(h_eq))}
print(aa)

# starting value of histogram
d = h
# geting the values of aa to final according to start histogram
for i in range(8):
    for j in range(8):
        c = aa.get(d[i, j])
        d[i, j] = c

print(d)
# final image output
plt.imshow(d, cmap='gray')
plt.xticks([]), plt.yticks([])
plt.show()
```

EXPERIMENT NO. 4

Aim - Arithmetic, Logical & Geometrical Operation

Code -

```
"""
Creator
@author:  BT18ECE025
          Aniruddha Gawate
"""

from scipy import signal
from skimage import img_as_float
import cv2
import numpy as np


# images are loaded with imread command
img_1 = cv2.imread('image/ss.jpg')
img_2 = cv2.imread('image/t.jpg')
img1 = img_as_float(cv2.resize(img_1, (300, 200)))
img2 = img_as_float(cv2.resize(img_2, (300, 200)))

cv2.imshow('Original 1', img1)
cv2.imshow('Original 2', img2)
# added image
weightedSum = cv2.addWeighted(img1, 0.6, img2, 0.7, 0)
cv2.imshow('Weighted Image', weightedSum)
# subtracted image
sub = cv2.subtract(img1, img2)
cv2.imshow('Subtracted Image', sub)

# logical
# and
bitwise_and = cv2.bitwise_and(img2, img1)
cv2.imshow("bit_and", bitwise_and)
# or
bitwise_or = cv2.bitwise_or(img2, img1)
cv2.imshow("bitwise_or", bitwise_or)
# xor
bitwise_xor = cv2.bitwise_xor(img2, img1)
```

```
cv2.imshow("bitwise_xor", bitwise_xor)
# not
bitwise_not = cv2.bitwise_not(img2)
cv2.imshow("bitwise_not", bitwise_not)

# Geometric Transformations
# Scaling
re = cv2.resize(img2, (300, 200))
cv2.imshow("resize", re)

m, n, ch = re.shape
A = cv2.getRotationMatrix2D((m/2, n/2), 90, 1)
aa = cv2.warpAffine(re, A, (m, n))
cv2.imshow("rotation", aa)

# interpolation
near_img = cv2.resize(re, None, fx=10, fy=10,
                      interpolation=cv2.INTER_NEAREST)
bilinear_img = cv2.resize(re, None, fx=10, fy=10,
                          interpolation=cv2.INTER_LINEAR)
bicubic_img = cv2.resize(re, None, fx=10, fy=10,
                         interpolation=cv2.INTER_CUBIC)
cv2.imshow("INTER_NEAREST", near_img)
cv2.imshow("INTER_LINEAR", bilinear_img)
cv2.imshow("INTER_CUBIC", bicubic_img)


while True:
    k = cv2.waitKey(0) & 0xFF
    if k == 27:
        cv2.destroyAllWindows()
        break
```

EXPERIMENT NO. 5

Aim - Logarithmic, Square and Square Root Operation

Code -

```
"""
Creator
@author:  BT18ECE025
          Aniruddha Gawate
"""

import cv2
import numpy as np

img = cv2.imread('image/t.jpg')
img = (cv2.resize(img, (300, 200)))
cv2.imshow('Original', img)

#################
# Apply log transform.
c = 255/(np.log(1 + np.max(img)))
log_transformed = c * np.log(1 + img)

# Specify the data type.
log_transformed = np.array(log_transformed, dtype=np.uint8)

cv2.imshow('log', log_transformed)
#################
# Square and Square root gamma values.
for gamma in [0.5, 1, 2]:

    # Apply gamma correction.
    gamma_corrected = np.array(255*(img / 255) ** gamma, dtype='uint8')

    cv2.imshow('gamma'+str(gamma), gamma_corrected)

while True:
    k = cv2.waitKey(0) & 0xFF
    if k == 27:
        cv2.destroyAllWindows()
        break
```

EXPERIMENT NO. 6

Aim - Linear, Correlation and convolution Operation

Code -

```
"""
Creator
@author:  BT18ECE025
          Aniruddha Gawate
"""
###################
# linear

# Function to map each intensity level to output intensity level.


def pixelVal(pix, r1, s1, r2, s2):
    if (0 <= pix and pix <= r1):
        return (s1 / r1)*pix
    elif (r1 < pix and pix <= r2):
        return ((s2 - s1)/(r2 - r1)) * (pix - r1) + s1
    else:
        return ((255 - s2)/(255 - r2)) * (pix - r2) + s2


# Define parameters.
r1 = 70
s1 = 0
r2 = 140
s2 = 255

# Vectorize the function to apply it to each value in the Numpy array.
pixelVal_vec = np.vectorize(pixelVal)

# Apply contrast stretching.
contrast_stretched = pixelVal_vec(img, r1, s1, r2, s2)

cv2.imshow('contrast_stretch', contrast_stretched)


# Correlation and convolution
```

```
aa = img_as_float(cv2.imread("image/t.jpg", 0))
aa = cv2.resize(aa, (300, 200))
kernel = np.ones((5, 5), np.float32)/25
gau_kernel = np.array([[1/16, 1/8, 1/16],
                       [1/8, 1/4, 1/8],
                       [1/16, 1/8, 1/16]])

conv = cv2.filter2D(aa, -1, gau_kernel, borderType=cv2.BORDER_DEFAULT)
cv2.imshow("Ori", aa)
cv2.imshow("Correlation & Convolution ", conv)


# aa = np.asarray(re)

while True:
    k = cv2.waitKey(0) & 0xFF
    if k == 27:
        cv2.destroyAllWindows()
        break
```

EXPERIMENT NO. 7

Aim - Gaussian Filter

Code -

```
"""
Creator
@author:  BT18ECE025
          Aniruddha Gawate
"""
# ImageFilter for using filter() function
from PIL import Image, ImageFilter

# Opening the image
# (R prefixed to string in order to deal with '\' in paths)
image = Image.open(r"IMAGE_PATH")

# Blurring image by sending the ImageFilter.
# GaussianBlur predefined kernel argument
image = image.filter(ImageFilter.GaussianBlur)

# Displaying the image
image.show()

#### Blurring a small region in an image
from PIL import Image, ImageFilter

image = Image.open(r"FILE_PATH")

# Cropping the image
smol_image = image.crop((0, 0, 150, 150))

# Blurring on the cropped image
blurred_image = smol_image.filter(ImageFilter.GaussianBlur)

# Pasting the blurred image on the original image
image.paste(blurred_image, (0,0))

# Displaying the image
image.save('output.png')
```

EXPERIMENT NO. 8

Aim - Averaging and Median Filter.

Code -

```
"""
Creator
@author:  BT18ECE025
          Aniruddha Gawate
"""

import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('opencv_logo.png')

blur = cv2.blur(img,(5,5))

plt.subplot(121),plt.imshow(img),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(blur),plt.title('Blurred')
plt.xticks([]), plt.yticks([])
plt.show()

# Median Filtering


median = cv2.medianBlur(img,5)
plt.subplot(121),plt.imshow(img),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(median),plt.title('Median')
plt.xticks([]), plt.yticks([])
plt.show()
```

EXPERIMENT NO. 9

Aim - Histogram Plotting and stretching

Code -

```
"""
Creator
@author:  BT18ECE025
          Aniruddha Gawate
"""

# import cv2, numpy, matplotlib
import cv2
import numpy as np
import matplotlib.pyplot as plt


# function to obtain histogram of an image
def hist_plot(img):

    # empty list to store the count
    # of each intensity value
    count =[]

    # empty list to store intensity
    # value
    r = []

    # loop to traverse each intensity
    # value
    for k in range(0, 256):
        r.append(k)
        count1 = 0

        # loops to traverse each pixel in
        # the image
        for i in range(m):
            for j in range(n):
                if img[i, j]== k:
                    count1+= 1
        count.append(count1)
```

```
    return (r, count)


img = cv2.imread('food.jpeg', 0)

# To ascertain total numbers of rows and
# columns of the image, size of the image
m, n = img.shape
r1, count1 = hist_plot(img)

# plotting the histogram
plt.stem(r1, count1)
plt.xlabel('intensity value')
plt.ylabel('number of pixels')
plt.title('Histogram of the original image')

# Transformation to obtain stretching
constant = (255-0)/(img.max()-img.min())
img_stretch = img * constant
r, count = hist_plot(img_stretch)

# plotting the histogram
plt.stem(r, count)
plt.xlabel('intensity value')
plt.ylabel('number of pixels')
plt.title('Histogram of the stretched image')

# Storing stretched Image
cv2.imwrite('Stretched Image 4.png', img_stretch)
```

EXPERIMENT NO. 10

Aim - Erosion and Dilation of image

Code -

```
"""
Creator
@author:  BT18ECE025
          Aniruddha Gawate
"""
# Python program to demonstrate erosion and
# dilation of images.
import cv2
import numpy as np

# Reading the input image
img = cv2.imread('input.png', 0)

# Taking a matrix of size 5 as the kernel
kernel = np.ones((5,5), np.uint8)

# The first parameter is the original image,
# kernel is the matrix with which image is
# convolved and third parameter is the number
# of iterations, which will determine how much
# you want to erode/dilate a given image.
img_erosion = cv2.erode(img, kernel, iterations=1)
img_dilation = cv2.dilate(img, kernel, iterations=1)

cv2.imshow('Input', img)
cv2.imshow('Erosion', img_erosion)
cv2.imshow('Dilation', img_dilation)

cv2.waitKey(0)
```

EXPERIMENT NO. 11

Aim - Run Length Encoding

Code -

```
"""
Creator
@author:  BT18ECE025
          Aniruddha Gawate
"""
# Python code for run length encoding
from collections import OrderedDict
def runLengthEncoding(input):

      # Generate ordered dictionary of all lower
      # case alphabets, its output will be
      # dict = {'w':0, 'a':0, 'd':0, 'e':0, 'x':0}
      dict=OrderedDict.fromkeys(input, 0)

      # Now iterate through input string to calculate
      # frequency of each character, its output will be
      # dict = {'w':4,'a':3,'d':1,'e':1,'x':6}
      for ch in input:
            dict[ch] += 1

      # now iterate through dictionary to make
      # output string from (key,value) pairs
      output = ''
      for key,value in dict.items():
            output = output + key + str(value)
      return output

# Driver function
if __name__ == "__main__":
      input="wwwwaaadexxxxxx"
      print (runLengthEncoding(input))
```

EXPERIMENT NO. 12

Aim - Edge Detection

Code -

```
"""
Creator
@author:  BT18ECE025
          Aniruddha Gawate
"""
#### 1
from PIL import Image, ImageFilter

# Opening the image (R prefixed to string
# in order to deal with '\' in paths)
image = Image.open(r"Sample.png")

# Converting the image to greyscale, as edge detection
# requires input image to be of mode = Greyscale (L)
image = image.convert("L")

# Detecting Edges on the Image using the argument ImageFilter.FIND_EDGES
image = image.filter(ImageFilter.FIND_EDGES)

# Saving the Image Under the name Edge_Sample.png
image.save(r"Edge_Sample.png")

### 2

from PIL import Image, ImageFilter

img = Image.open(r"sample.png")

# Converting the image to greyscale, as Sobel Operator requires
# input image to be of mode Greyscale (L)
img = img.convert("L")

# Calculating Edges using the passed laplican Kernel
final = img.filter(ImageFilter.Kernel((3, 3), (-1, -1, -1, -1, 8,
                                               -1, -1, -1, -1), 1, 0))
final.save("EDGE_sample.png")
```

EXPERIMENT NO. 13

Aim - Edge detection using Sobel

Code -

```
"""
Creator
@author:  BT18ECE025
          Aniruddha Gawate
"""

# Python program to  Edge detection
# using OpenCV in Python
# using Sobel edge detection
# and Laplacian method
import cv2
import numpy as np

#Capture livestream video content from camera 0
cap = cv2.VideoCapture(0)

while(1):

    # Take each frame
    _, frame = cap.read()

    # Convert to HSV for simpler calculations
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # Calcution of Sobelx
    sobelx = cv2.Sobel(frame,cv2.CV_64F,1,0,ksize=5)

    # Calculation of Sobely
    sobely = cv2.Sobel(frame,cv2.CV_64F,0,1,ksize=5)

    # Calculation of Laplacian
    laplacian = cv2.Laplacian(frame,cv2.CV_64F)

    cv2.imshow('sobelx',sobelx)
    cv2.imshow('sobely',sobely)
    cv2.imshow('laplacian',laplacian)
```

```
    k = cv2.waitKey(5) & 0xFF
    if k == 27:
        break

cv2.destroyAllWindows()

#release the frame
cap.release()
```

EXPERIMENT NO. 14

Aim - Color Identification in Images

Code - For finding the green color in the image, we need to specify the lower and upper HSV color code for green color as follows.

lower=np.array([50, 100,100])

upper=np.array([70, 255, 255])

```
"""
Creator
@author:  BT18ECE025
         Aniruddha Gawate
"""
# Python program to identify
#color in images

# Importing the libraries OpenCV and numpy
import cv2
import numpy as np
# Read the images
img = cv2.imread("Resources/shapes.jpg")

# Resizing the image
image = cv2.resize(img, (700, 600))

# Convert Image to Image HSV
hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# Defining lower and upper bound HSV values
lower = np.array([50, 100, 100])
upper = np.array([70, 255, 255])

# Defining mask for detecting color
mask = cv2.inRange(hsv, lower, upper)

# Display Image and Mask
cv2.imshow("Image", image)
cv2.imshow("Mask", mask)
# Make python sleep for unlimited time
cv2.waitKey(0)
```

EXPERIMENT NO. 15

Aim - Face Detection

Code -

```
"""
Creator
@author:  BT18ECE025
          Aniruddha Gawate
"""

import cv2 as cv

# Read image from your local file system
original_image = cv.imread('img.jpg')


# Convert color image to grayscale for Viola-Jones
grayscale_image = cv.cvtColor(original_image, cv.COLOR_BGR2GRAY)

# Load the classifier and create a cascade object for face detection
face_cascade =
cv.CascadeClassifier('path/to/haarcascade_frontalface_alt.xml')

detected_faces = face_cascade.detectMultiScale(grayscale_image)


for (column, row, width, height) in detected_faces:
    cv.rectangle(
        original_image,
        (column, row),
        (column + width, row + height),
        (0, 255, 0),
        2)


cv.imshow('Image', original_image)
cv.waitKey(0)
cv.destroyAllWindows()
```