prac1A. Aim: Perform Geometric Transformation:

1. Rotation:-

I = imread('filepath');

J = imrotate(I,35,'bilinear');

figure

imshowpair(I,J,'montage')

2. Translation:-

I = imread('C:\Users\DELL\Pictures\.LockScreen\LockScreen1626295342121.jpg');

shiftX =195;%shift columns

shiftY =195;%shift rows

subplot(1,2,1);

imshow(I);

title('Original Image');

nI = uint8(zeros(size(I,1)+shiftY-1,size(I,2)+shiftX-1,size(I,3)));

nI(shiftY:end,shiftX:end,:)=I;

subplot(1,2,2);

imshow(nI);

title('Translated Image');

3. Scaling:-

I = imread('C:\Users\DELL\Pictures\iphone12\202404\_\_\CQUE0852.JPG');

J = imresize(I, 0.5);

figure

imshowpair(I,J,'montage');

4. Reflection:-

i = imread('C:\Users\DELL\Pictures\iphone12\202312\_\_\IMG\_E1244.JPG');

subplot(2,2,1)

imshow(i)

title('Original')

ir = flipdim(i,2)

subplot(2,2,2)

imshow(ir)

title('Flipped')

prac2: Image Stitching:-

clc

clear all

i = imread('C:\Users\DELL\Desktop\computer vision\image 1.jpg');

j = imread('C:\Users\DELL\Desktop\computer vision\image2.jpg');

si = size(i);

sj = size(j);

j = imresize(j,[si(1) sj(2)]);

k = [i j];

imshow(k)

prac3.Camera Callibration:-

import cv2

import numpy as np

import os

import glob

CHECKERBOARD = (6, 9)

criteria = (cv2.TERM\_CRITERIA\_EPS +

cv2.TERM\_CRITERIA\_MAX\_ITER, 30, 0.001)

threedpoints = []

twodpoints = []

objectp3d = np.zeros((1, CHECKERBOARD[0] \* CHECKERBOARD[1], 3), np.float32)

objectp3d[0, :, :2] = np.mgrid[0:CHECKERBOARD[0],0:CHECKERBOARD[1]].T.reshape(-1, 2)

prev\_img\_shape = None

images = glob.glob('\*.jpg')

for filename in images:

image = cv2.imread(filename)

grayColor = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

ret, corners = cv2.findChessboardCorners(grayColor, CHECKERBOARD,

cv2.CALIB\_CB\_ADAPTIVE\_THRESH + cv2.CALIB\_CB\_FAST\_CHECK + cv2.CALIB\_CB\_NORMALIZE\_IMAGE)

if ret == True:

threedpoints.append(objectp3d)

corners2 = cv2.cornerSubPix(

grayColor, corners, (11, 11), (-1, -1), criteria)

twodpoints.append(corners2)

image = cv2.drawChessboardCorners(image,CHECKERBOARD,corners2, ret)

cv2.imshow('img', image)

cv2.waitKey(0)

cv2.destroyAllWindows()

h, w = image.shape[:2]

ret, matrix, distortion, r\_vecs, t\_vecs = cv2.calibrateCamera(threedpoints, twodpoints, grayColor.shape[::-1], None, None)

print(" Camera matrix:")

print(matrix)

print("\n Distortion coefficient:")

print(distortion)

print("\n Rotation Vectors:")

print(r\_vecs)

print("\n Translation Vectors:")

print(t\_vecs)

Prac4.Face Detection:-

clc

clear all

FDetect = vision.CascadeObjectDetector;

I =imread('C:\Users\DELL\Pictures\.LockScreen\LockScreen1626295154709.jpg');

BB = step(FDetect,I);

figure,

imshow(I);

hold on

for i = 1:size(BB,1)

rectangle('Position',BB(i,:),'Linewidth',5,'LineStyle','-','EdgeColor','r');

end

title('FaceDetection');

hold off;

prac5.Video Face Recognition:-

faceDetector = vision.CascadeObjectDetector();

videoFileReader = vision.VideoFileReader('C:\Users\DELL\Pictures\whatsapp\Media\WhatsApp Video\Sent\VID-20230506-WA0033.mp4');

videoFrame = step(videoFileReader);

bbox = step(faceDetector, videoFrame);

videoOut = insertObjectAnnotation(videoFrame,'rectangle',bbox,'Face');

figure,

imshow(videoOut),

title('Detected face');

prac6: Perform Text Detection and recognition

Step 1: Download and Install “pytesseract.exe” on computer

[Note: you can download application using this link “https://github.com/UB-Mannheim/tesseract/wiki”)

Step 2: Install below packages

!pip install opencv-python

!pip install pytesseract

import cv2

import pytesseract as pytesseract

#locate the installed tesseract application

pytesseract.pytesseract.tesseract\_cmd =r'C:/Program Files/Tesseract-OCR/tesseract.exe'

#read image

img=cv2.imread("hello.jpg")

#read height and width of image

height,width,c=img.shape

#extract the words from image

words\_in\_image=pytesseract.image\_to\_string(img)

print(words\_in\_image)

#draw boxes around the words

letter\_boxes=pytesseract.image\_to\_boxes(img)

#fix the boxes size

for box in letter\_boxes.splitlines():

box=box.split()

x,y,w,h=int(box[1]),int(box[2]),int(box[3]),int(box[4])

cv2.rectangle(img,(x,height-y),(w,height-h),(0,0,255),3)

cv2.putText(img,box[0], (x,height+32),cv2.FONT\_HERSHEY\_COMPLEX,1,(0,255,0),2) #recognize display

#letters in green colour

#display image

cv2.imshow("window",img)

cv2.waitKey(0)

prac7:-Perform Image Matting and Composition

import cv2

import matplotlib.pyplot as plt

import numpy as np

I = cv2.imread('GT04.png') # load image I (BGR format)

I = cv2.cvtColor(I, cv2.COLOR\_BGR2RGB)/255 # convert to RGB and normalize

plt.imshow(I), plt.axis('off') # plot I

plt.show();

alpha\_ex = cv2.imread('GT04\_alpha.png', cv2.IMREAD\_GRAYSCALE)/255 # load exact

alpha and normalize

plt.imshow(alpha\_ex, cmap='gray'), plt.axis('off') # plot alpha

plt.show();

n\_rows = I.shape[0] # number of rows in I

n\_cols = I.shape[1] # number of columns in I

n\_pixels = n\_rows \* n\_cols # number of pixels in I

I = np.reshape(I, (n\_pixels, 3))

I[:, 0] = I[:, 1] # red = green

I[:, 2] = I[:, 1] # blue = green

alpha\_ex = np.reshape(alpha\_ex, (n\_pixels, 1))

G\_B = 1 # green screen value

I = alpha\_ex \* I + (1 - alpha\_ex) \* [0, G\_B, 0] # replace background with green screen

I = np.reshape(I, (n\_rows, n\_cols, 3))

plt.imshow(I), plt.axis('off') # plot I

plt.show();

R\_I = I[:, :, 0] # red component of I

G\_I = I[:, :, 1] # green component of I

B\_I = I[:, :, 2] # blue component of I

compute the matte 𝛼:

alpha = (R\_I - (G\_I - G\_B))/G\_B # compute alpha with formula

plt.imshow(alpha, cmap='gray'), plt.axis('off') # plot alpha

plt.show();

alpha = np.reshape(alpha, (n\_pixels, 1))

error = np.linalg.norm(alpha - alpha\_ex)/np.linalg.norm(alpha\_ex)

print(f'Error (alpha): {error:.2e}')

Error (alpha): 3.63e-17

K = cv2.imread('toronto.jpg') # load image (BGR format)

K = cv2.cvtColor(K, cv2.COLOR\_BGR2RGB)/255 # convert to RGB and normalize

K = K[:n\_rows, :n\_cols, :] # adjust size of K to size of I

plt.imshow(K), plt.axis('off') # plot K

plt.show();

K = np.reshape(K, (n\_pixels, 3))

R\_I = np.reshape(R\_I, (n\_pixels, 1))

J = np.tile(R\_I, 3) + (1 - alpha) \* K # new image J with different background K

J = np.reshape(J, (n\_rows, n\_cols, 3))

plt.imshow(J), plt.axis('off') # plot J

plt.show();