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();
```