EXP\_5

**Image transform:**

1)Translation

import numpy as np

import cv2

img = cv2.imread('image.jpg')

if img is None:

print("Error: Unable to load the image.")

else:

cv2.imshow('img.jpg', img)

rows, cols, \_ = img.shape

M = np.float32([[1, 0, 100], [0, 1, 50]])

dst = cv2.warpAffine(img, M, (cols, rows))

cv2.imshow('img', dst)

cv2.waitKey(0)

cv2.destroyAllWindows()

2) Reflection

import numpy as np

import cv2

img = cv2.imread('image.jpg',0)

cv2.imshow('image',img)

rows, cols = img.shape

M = np.float32([[1, 0, 0],[0, -1, rows],[0, 0, 1]])

reflected\_img = cv2.warpPerspective(img, M,(int(cols),int(rows)))

cv2.imshow('img', reflected\_img)

cv2.imwrite('reflection\_out.jpg', reflected\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

3)Rotation

import numpy as np

import cv2 as cv

img = cv.imread('image.jpg', 1)

cv.imshow('image', img)

rows, cols, \_ = img.shape

M = cv.getRotationMatrix2D((cols/2, rows/2), 30, 0.6)

img\_rotation = cv.warpAffine(img, M, (cols, rows))

cv.imshow('img', img\_rotation)

cv.imwrite('rotation\_out.jpg', img\_rotation)

cv.waitKey(0)

cv.destroyAllWindows()

4)Scaling

import numpy as np

import cv2 as cv

img = cv.imread('image.jpg', 0)

rows, cols = img.shape

img\_shrinked = cv.resize(img, (250, 200),interpolation=cv.INTER\_AREA)

cv.imshow('img', img\_shrinked)

img\_enlarged = cv.resize(img\_shrinked, None,fx=1.5, fy=1.5,interpolation=cv.INTER\_CUBIC)

cv.imshow('img', img\_enlarged)

cv.waitKey(0)

cv.destroyAllWindows()

5)Shearing\_x

import numpy as np

import cv2 as cv

img = cv.imread('image.jpg', 1)

rows, cols,\_ = img.shape

M = np.float32([[1, 0.5, 0], [0, 1, 0], [0, 0, 1]])

sheared\_img = cv.warpPerspective(img, M, (int(cols\*1.5), int(rows\*1.5)))

cv.imshow('img', sheared\_img)

cv.waitKey(0)

cv.destroyAllWindows()

6)Shearing\_y

import numpy as np

import cv2 as cv

img = cv.imread('image.jpg', 1)

rows, cols,\_ = img.shape

M = np.float32([[1, 0, 0], [0.5, 1, 0], [0, 0, 1]])

sheared\_img = cv.warpPerspective(img, M, (int(cols\*1.5), int(rows\*1.5)))

cv.imshow('sheared\_y-axis\_out.jpg', sheared\_img)

cv.waitKey(0)

cv.destroyAllWindows()

**Extract** **ORB** **Image** **features:**

import cv2

img = cv2.imread('image.jpg')

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

orb = cv2.ORB\_create(nfeatures=2000)

kp = orb.detect(gray, None)

kp, des = orb.compute(gray, kp)

img1 = cv2.drawKeypoints(gray, kp, None, (0,0,255), flags=0)

cv2.imshow("ORB Keypoints", img1)

cv2.waitKey(0)

cv2.destroyAllWindows()

**FEATURE MATCHING:**

import cv2

image = cv2.imread('image.jpg')

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

fast = cv2.FastFeatureDetector\_create()

fast.setNonmaxSuppression(False)

kp = fast.detect(gray\_image, None)

kp\_image = cv2.drawKeypoints(image, kp, None, color=(0, 255, 0))

cv2.imshow('FAST', kp\_image)

cv2.waitKey()

**Cloning**

**Code will work if apt images is selected. Choose image such that the background of one image can merge with other**

import cv2

import numpy as np

src = cv2.imread("clone2.jpg")

dst = cv2.imread("clone1.jpg")

src\_mask = np.zeros(src.shape, src.dtype)

poly = np.array([ [4,80], [30,54], [151,63], [254,37], [298,90], [272,134], [43,122] ],np.int32)

cv2.fillPoly(src\_mask, [poly], (255, 255, 255))

center = (800,100)

output = cv2.seamlessClone(src, dst, src\_mask, center, cv2.NORMAL\_CLONE)

cv2.imwrite("images/opencv-seamless-cloning-example.jpg", output)

OUTPUT:

1)TRANSLATION



2)REFLECTION



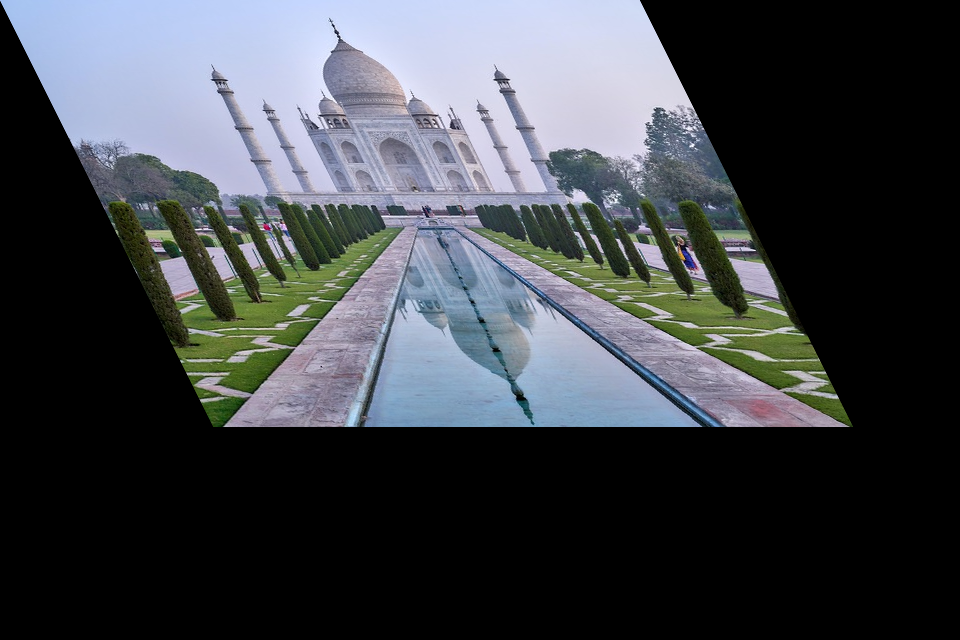
3)ROTATION



4)SCALING



5)SHEARING\_X



6)SHEARING\_Y



7)Extract ORB Image features



8)Feature Matching

