Homework5 Stitching

胡成成 2101210578

Question

二选一：

1. 自己拍摄两张有重叠区域的照片，然后拼接起来
2. 自选一张图分别用Harris， SIFT，SURF， ORB提取特征点，并比较提取的特征点数和耗时。

Answer

选择第一题：

- 代码：

```
import cv2
import numpy as np

# set a windows
top, bot, left, right = 100, 100, 0, 500
img1 = cv2.imread('image2.jpg')
img2 = cv2.imread('image1.jpg')

# copy image to windows
srcImg = cv2.copyMakeBorder(img1, top, bot, left, right, cv2.BORDER_CONSTANT, value=(0, 0, 0))
testImg = cv2.copyMakeBorder(img2, top, bot, left, right, cv2.BORDER_CONSTANT, value=(0, 0, 0))
# transform to gray
img1gray = cv2.cvtColor(srcImg, cv2.COLOR_BGR2GRAY)
img2gray = cv2.cvtColor(testImg, cv2.COLOR_BGR2GRAY)
# setting SIFT
sift = cv2.SIFT_create()
```

```
# find the keypoints and descriptors with SIFT
kp1, des1 = sift.detectAndCompute(img1gray, None)
kp2, des2 = sift.detectAndCompute(img2gray, None)
# FLANN parameters
FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
search_params = dict(checks=50)
flann = cv2.FlannBasedMatcher(index_params, search_params)
matches = flann.knnMatch(des1, des2, k=2)

# Need to draw only good matches, so create a mask
matchesMask = [[0, 0] for i in range(len(matches))]
good = []
pts1 = []
pts2 = []
# ratio test as per Lowe's paper
for i, (m, n) in enumerate(matches):
    if m.distance < 0.7 * n.distance:
        good.append(m)
        pts2.append(kp2[m.trainIdx].pt)
        pts1.append(kp1[m.queryIdx].pt)
        matchesMask[i] = [1, 0]

draw_params = dict(matchColor=(0, 255, 0), singlePointColor=(255, 0, 0),
                   matchesMask=matchesMask, flags=0)
img3 = cv2.drawMatchesKnn(img1gray, kp1, img2gray, kp2, matches, None, **draw_params)
cv2.imshow("stiching", img3)
cv2.imwrite("stiching.jpg", img3)

rows, cols = srcImg.shape[:2]
MIN_MATCH_COUNT = 10
if len(good) > MIN_MATCH_COUNT:
    src_pts = np.float32([kp1[m.queryIdx].pt for m in good]).reshape(-1, 1, 2)
    dst_pts = np.float32([kp2[m.trainIdx].pt for m in good]).reshape(-1, 1, 2)
    M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)
    warpImg = cv2.warpPerspective(testImg, np.array(M), (testImg.shape[1], testImg.shape[0]),
                                  flags=cv2.WARP_INVERSE_MAP)

    for col in range(0, cols):
        if srcImg[:, col].any() and warpImg[:, col].any():
            left = col
            break
    for col in range(cols - 1, 0, -1):
        if srcImg[:, col].any() and warpImg[:, col].any():
            right = col
            break
    #
    res = np.zeros([rows, cols, 3], np.uint8)
    for row in range(0, rows):
        for col in range(0, cols):
            if not srcImg[row, col].any():
                res[row, col] = warpImg[row, col]
            elif not warpImg[row, col].any():
                res[row, col] = srcImg[row, col]
            else:
                srcImgLen = float(abs(col - left))
                testImgLen = float(abs(col - right))
                alpha = srcImgLen / (srcImgLen + testImgLen)
                res[row, col] = np.clip(srcImg[row, col] * (1 - alpha) + warpImg[row, col] * alpha, 0, 255)
            # show the result
    cv2.imshow("p12", res)
    cv2.imwrite("p12.jpg", res)
```

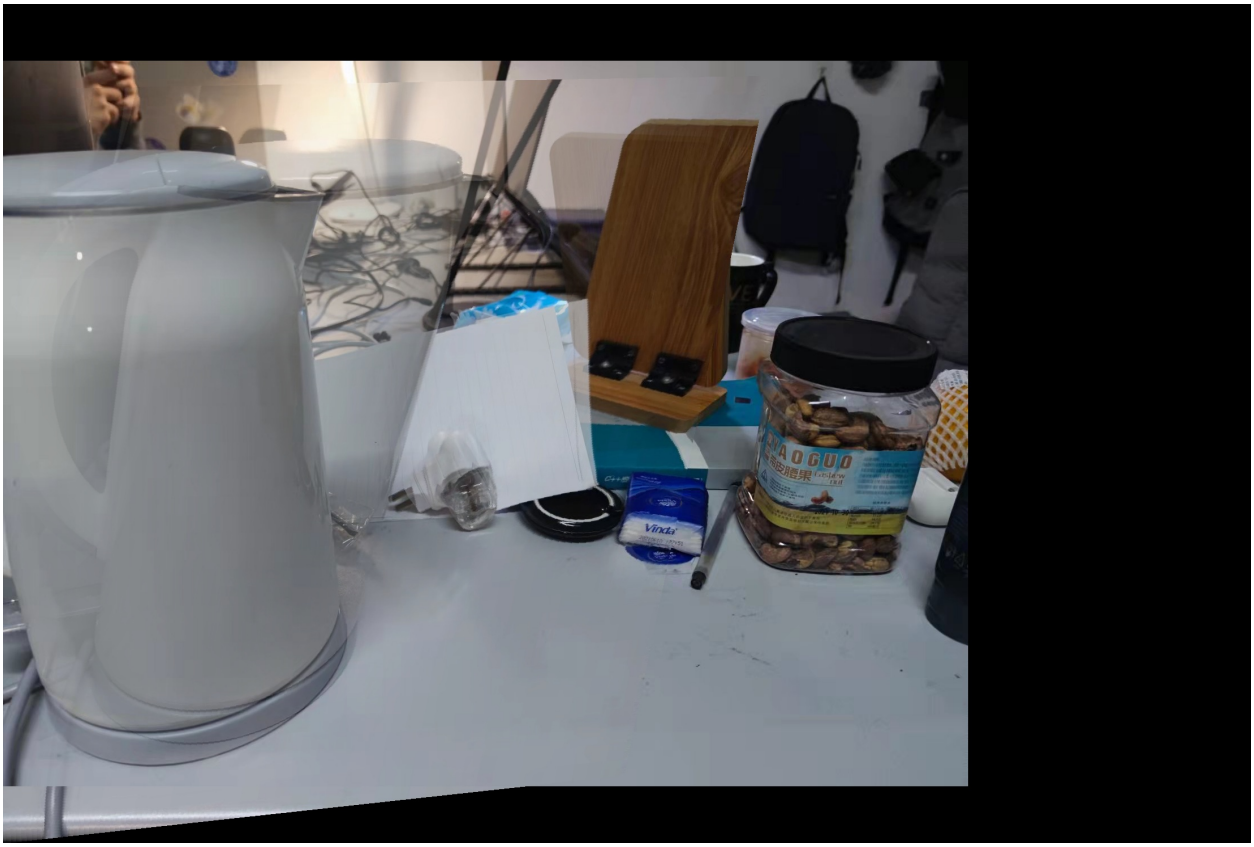
```
else:
    print("Not enough matches are found - {}/{}".format(len(good), MIN_MATCH_COUNT))
    matchesMask = None

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

- 测试图片1：

- 测试图片2：

- 运行结果：



- 分析：可以看出整体的拼接效果还可以，但是镜子的拼接部分缺了一块。