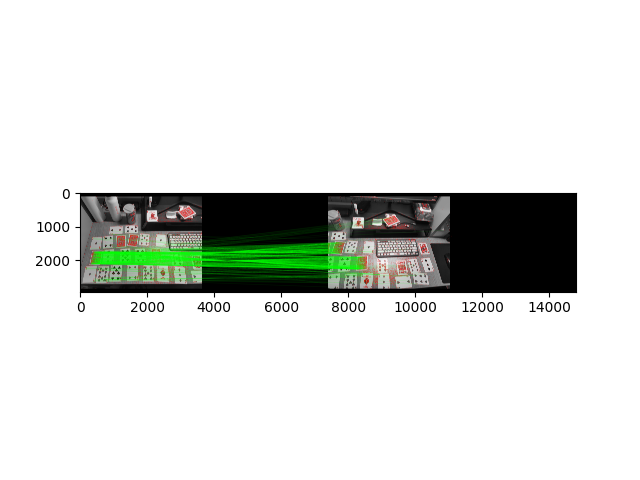
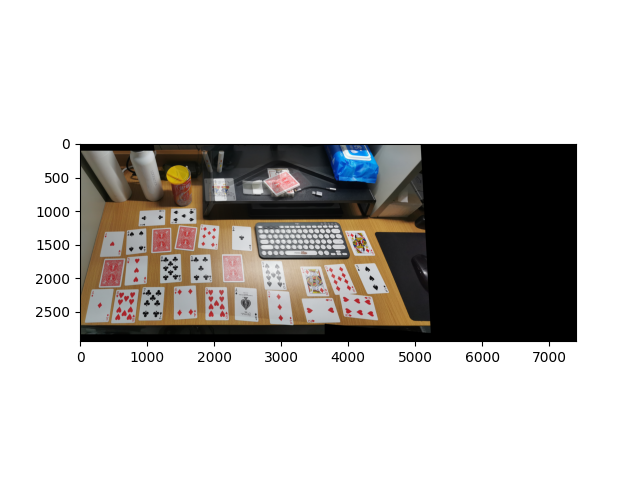
**计算机视觉第四次作业-图像拼接与融合**

梁芮槐-2019302789

1. 给出你用手机拍摄两张有重叠视野的照片（建议两幅图像打光不同，以体现第五步融合的效果）。
2. 用sift特征或者任意你熟悉的特征进行匹配。（附特征点图）
3. 用RANSAC方法去除误匹配点对。（附筛除后的特征点图）
4. 给出变换矩阵（仿射模型或投射模型都可以，需注明），并完成拼接（前向映射或反向映射都可以，需注明），给出拼接后的图像。
5. 对拼接后的图像进行融合（任意你喜欢的融合方法），附融合后的图像。

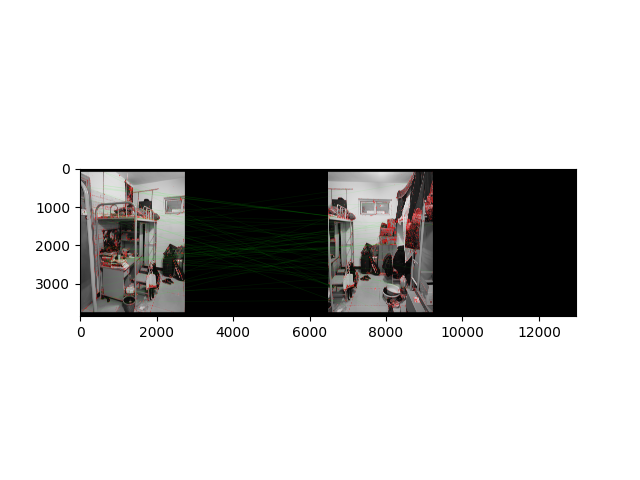
第一组桌面，原图-筛除后的特征点图-融合结果





第二组房间，原图-筛除后的特征点图-融合结果







根据欧氏距离相邻程度初步筛选然后RANSAC筛除后的匹配点还是略微有些误匹配，比如相同花色的墙纸，会把不同位置的点匹配到一起，最终结果没有太大问题，但是有鬼影的现象。

代码

import sys

import numpy as np

import cv2 as cv

from matplotlib import pyplot as plt

if \_\_name\_\_ == '\_\_main\_\_':

    # 手机照片尺寸3648\*2736 先给左图加个边框

    top, bot, left, right = 100, 100, 0, 3748

    if len(sys.argv) != 3:

        print("Usage: python3 fusion.py <leftimg> <rightimg>\n")

        exit()

    img1 = cv.imread(sys.argv[1])

    img2 = cv.imread(sys.argv[2])

    srcImg = cv.copyMakeBorder(img1, top, bot, left, right, cv.BORDER\_CONSTANT, value=(0, 0, 0))

    testImg = cv.copyMakeBorder(img2, top, bot, left, right, cv.BORDER\_CONSTANT, value=(0, 0, 0))

    img1gray = cv.cvtColor(srcImg, cv.COLOR\_BGR2GRAY)

    img2gray = cv.cvtColor(testImg, cv.COLOR\_BGR2GRAY)

    # SIFT特征检测

    sift = cv.xfeatures2d\_SIFT().create()

    kp1, des1 = sift.detectAndCompute(img1gray, None)

    kp2, des2 = sift.detectAndCompute(img2gray, None)

    # FLANN 特征匹配参数

    FLANN\_INDEX\_KDTREE = 1

    index\_params = dict(algorithm=FLANN\_INDEX\_KDTREE, trees=5)

    search\_params = dict(checks=50)

    flann = cv.FlannBasedMatcher(index\_params, search\_params)

    matches = flann.knnMatch(des1, des2, k=2)

    # 内部匹配掩码

    matchesMask = [[0, 0] for i in range(len(matches))]

    good = []

    pts1 = []

    pts2 = []

    # 根据特征点接近程度筛选相邻匹配

    for i, (m, n) in enumerate(matches):

        if m.distance < 0.7\*n.distance:

            good.append(m)

            pts1.append(kp1[m.queryIdx].pt)

            pts2.append(kp2[m.trainIdx].pt)

            matchesMask[i] = [1, 0]

    # 画出部分匹配点

    draw\_params = dict(matchColor=(0, 255, 0),

                       singlePointColor=(255, 0, 0),

                       matchesMask=matchesMask[:5000],

                       flags=0)

    img3 = cv.drawMatchesKnn(img1gray, kp1, img2gray, kp2, matches[:5000], None, \*\*draw\_params)

    # plt.imshow(img3)

    # plt.show()

    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)

        # RANSAC计算单应矩阵

        M, mask = cv.findHomography(src\_pts, dst\_pts, cv.RANSAC, 5.0)

        # 透视变换

        warpImg = cv.warpPerspective(testImg, np.array(M), (testImg.shape[1], testImg.shape[0]), flags=cv.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)

        # 把opencv的图像转到matplotlib显示

        res = cv.cvtColor(res, cv.COLOR\_BGR2RGB)

        # 显示结果

        plt.figure()

        plt.imshow(res)

        plt.show()

    else:

        print("Not enough matches are found - {}/{}".format(len(good), MIN\_MATCH\_COUNT))

        matchesMask = None