《计算机视觉》实验报告

姓名：严昕宇 学号：20121802

实验 7

一. 任务 1

a) 核心代码：

实验7 SIFT特征

完成SIFT特征匹配，实现图片拼接，包括以下步骤

(1) 输入两张同一场景不同视角拍摄的图片
(2) 分别提取图片的SIFT特征
(3) 关键点匹配
(4) 采用RANSAC算法进行提纯
(5) 获取两张图片的变换关系（单应性矩阵，可通过4对匹配点求出），完成拼接（选做加分）
以上步骤都需要给出实验结果图

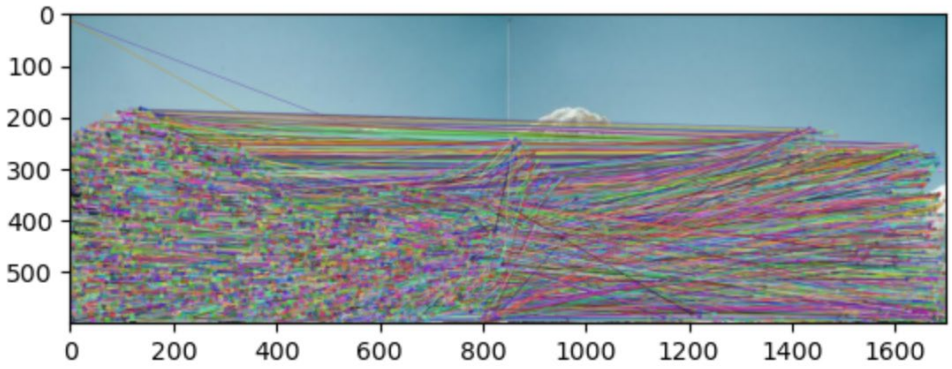
[1]:
```python
import random
import math
import cv2
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from PIL import Image
import numpy as np

#设置一个至少10个匹配的条件（有MinMatchNum指定）来找目标
MinMatchNum = 20
#读取照片
L = cv2.imread('1.png')          # queryImage
R = cv2.imread('2.png')          # trainImage
#高斯滤波
L = cv2.GaussianBlur(L,(3,3),0)
R = cv2.GaussianBlur(R,(3,3),0)
```

2. 分别提取图片的SIFT特征

[2]:
```python
#创建sift检测器
sift = cv2.SIFT_create()
# 计算所有特征点的特征值kp和特征向量des并获取
left_kp, left_des = sift.detectAndCompute(R, None)
righ_kp, righ_des = sift.detectAndCompute(L, None)
```

3. 关键点匹配

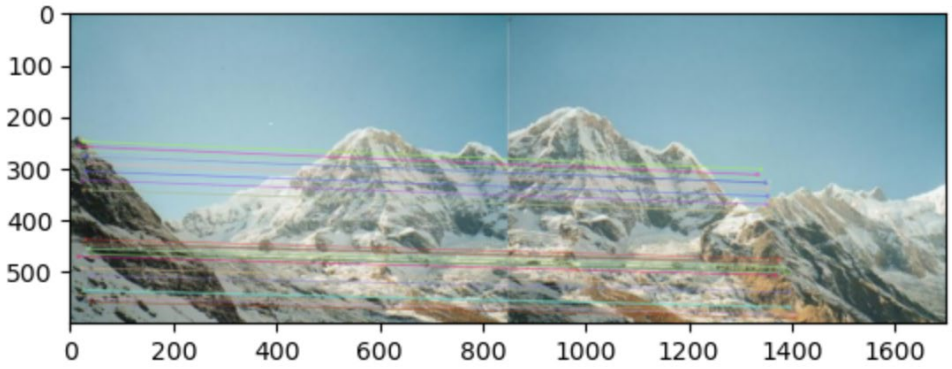
[3]:
```
# BFMatcher爆力解决匹配，但是不好的特征值匹配较多
bf = cv2.BFMatcher()
matches = bf.knnMatch(left_des, righ_des, k=2)
# 进行特征点匹配筛选
BetterChoose1 = []
for m, n in matches:
    #认为第一近的点小于第二近的点一倍以上是好的匹配BetterChoose1
    if m.distance < 0.5 * n.distance:
        BetterChoose1.append(m)
# 绘制匹配结构
match_result = cv2.drawMatchesKnn(L, left_kp, R, righ_kp, matches, None)
plt.imshow(cv2.cvtColor(match_result, cv2.COLOR_BGR2RGB))
plt.show()
```



4. 采用RANSAC算法进行提纯

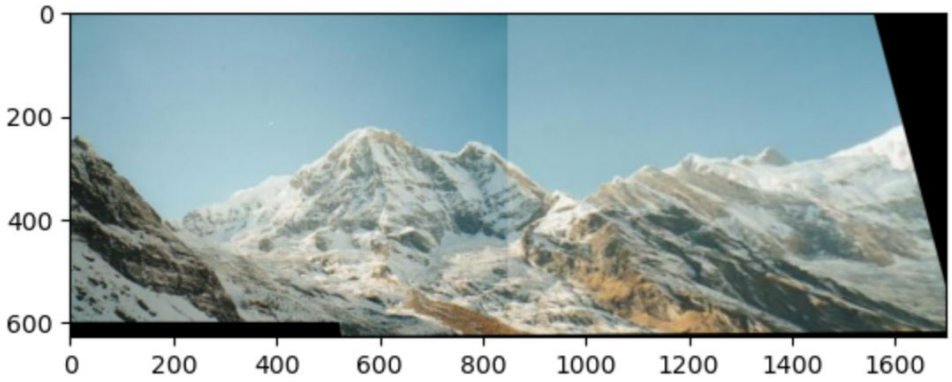
[4]:
```
# 但是由于爆力匹配的较好结果BetterChoose1匹配效果仍然不理想。
# 所以我们想到用Ransac的方法优化匹配结果
BetterChoose2 = np.expand_dims(BetterChoose1, 1)
match = cv2.drawMatchesKnn(L, left_kp, R, righ_kp, BetterChoose2[:30], None, flags=2)
# 判断是否当前模型已经符合超过MinMatchNum个点
if len(BetterChoose1) > MinMatchNum:
    # 获取关键点的坐标
    src_pts = np.float32([left_kp[m.queryIdx].pt for m in BetterChoose1]).reshape(-1, 1, 2)
    dst_pts = np.float32([righ_kp[m.trainIdx].pt for m in BetterChoose1]).reshape(-1, 1, 2)
    #在这里调用RANSAC方法得到解H
    H, modle = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)
    wrap = cv2.warpPerspective(R, H, (R.shape[1] + R.shape[1], R.shape[0] + R.shape[0]))
    wrap[0:R.shape[0], 0:R.shape[1]] = L
    #得到新的位置
    rows, cols = np.where(wrap[:, :, 0] != 0)
    min_row, max_row = min(rows), max(rows) + 1
    min_col, max_col = min(cols), max(cols) + 1
    # 去除黑色无用部分
    LeftAndRight = wrap[min_row:max_row, min_col:max_col, :]
```

[5]:
```
plt.imshow(cv2.cvtColor(match, cv2.COLOR_BGR2RGB))
plt.show()
```



5. 获取两张图片的变换关系，完成拼接

[6]:
```
# 将拼接结果进行显示
plt.imshow(cv2.cvtColor(LeftAndRight, cv2.COLOR_BGR2RGB))
plt.show()
```



b) 实验结果截图

图 1 实验结果 1

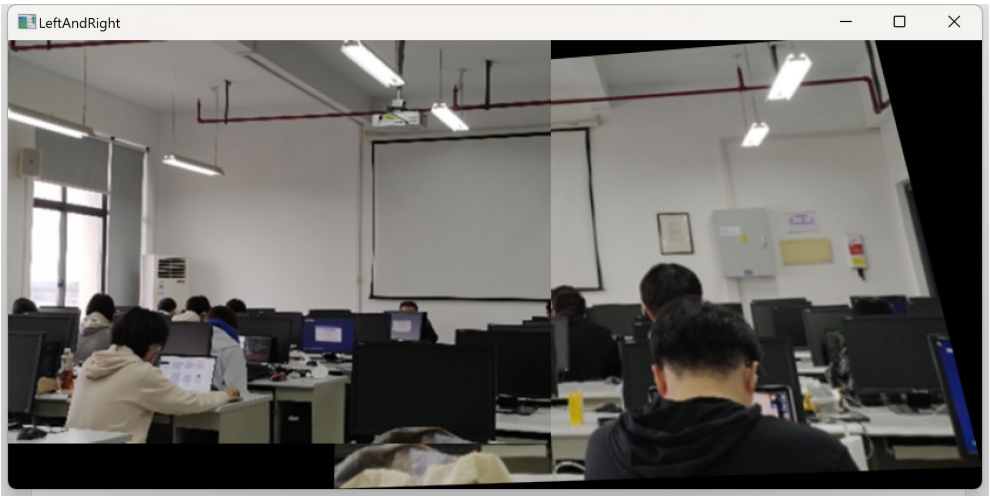


图 2 实验结果 2

c) 实验小结

图像拼接是将同一场景的多个重叠图像拼接成较大的图像的一种方法，在医学成像、卫星数据、军事目标自动识别等领域具有重要意义。本次实验使用 SIFT 特征实现了图像凭借，最后拼接效果较好(如测试结果中的雪山)。但是如果两张图差距较大或者有明显的色差和边缘也会影响到拼接效果(如实验结果 1 中的树木、实验结果 2 中的教室场景拼接边缘)，因此个人认为此算法比较适用于场景较暗或者色调单一的场景。