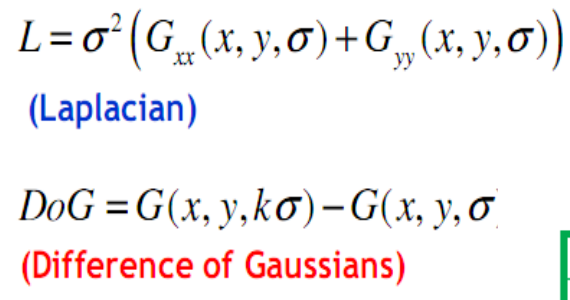
Homework3：图像SIFT特征提取与匹配&基于单应矩阵的图像视点变化与拼接

人工智能91 卢佳源 2191121196

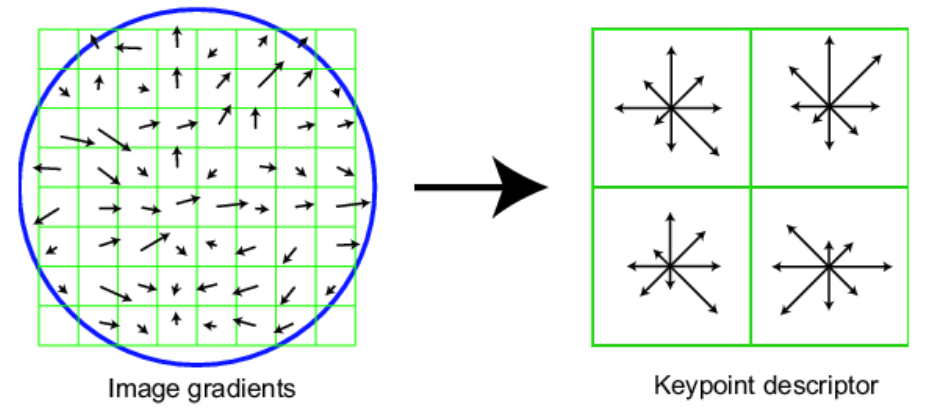
1. 掌握图像的SIFT特征检测原理：

SIFT，即尺度不变的特征变换，算法过程为特征点提取（生成DOG尺度空间），寻找局部极值点并规定一个阈值进行特征点筛选，来确定特征点主方向，再根据SIFT特征描述子相似性来进行特征点匹配、匹配对比值提纯以及用RANSAC方法剔除离群的匹配对。



1. 掌握图像特征描述子的匹配度量（距离比），RANSAC方法：

1、匹配度量：



如图，左图中央为一个关键点的位置，每个小格为该关键点邻域所在的尺度空间的一个像素，箭头表示每个像素的梯度方向，长度表示该梯度的幅值，然后利用高斯窗口对其进行加权运算，形成右图对每个大块的八个方向的梯度直方图，即把每个大块里的小块（一个像素）进行梯度累加得到右图的一个块（种子点），关键点描述子（SIFT特征描述子）即为这样形成的四个种子点组成。

特征匹配即是通过两个点集合内关键点描述子的匹配度量来完成，这里选用欧式距离作为相似性度量。

假设模板图中的关键点描述子为：（128维SIFT特征向量）

待匹配图（实时图）中的关键点描述子为：

任意两个描述子之间的匹配度量（欧式距离）为：

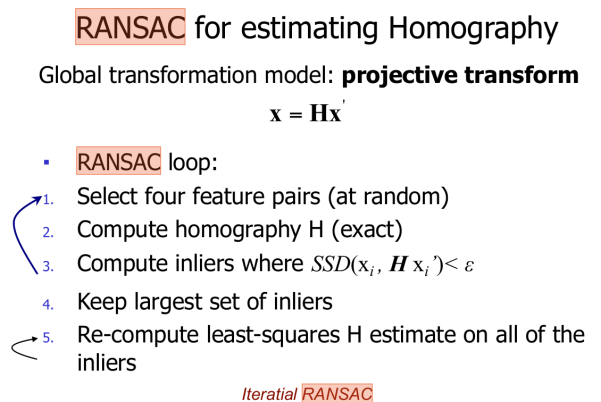
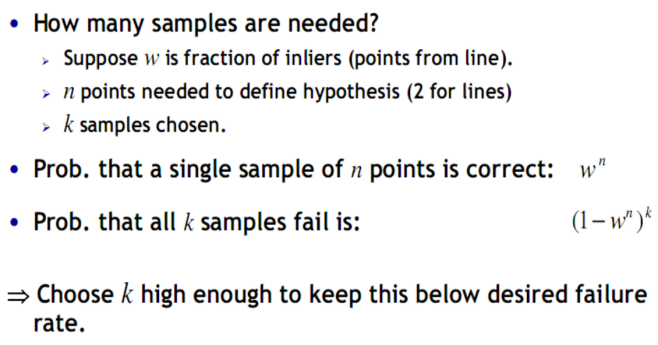
则最终按照阈值筛选出来的配对的关键点描述子满足：

1. RANSAC方法：

RANSAC方法是用来解决最小二乘法中离群值对样本拟合模型的干扰问题。RANSAC方法的基本假设是内群数据可以通过一些模型参数来描述出它们的分布，而离群的数据（噪声）则是不适合模型化的数据。“RANSAC假定，给定一组（通常很小的）内群，存在一个 程序，这个程序可以估算最佳解释或最适用于这一数据模型的参数。”

RANSAC方法的迭代过程：（要求参与评估的数学模型已知）

1. 在输入的观测数据中随机选择n个点并将其设定为内群；
2. 计算出适合这n个点的模型，如线性直线模型或多项式模型；
3. 把观测数据中未被选中为内群点的点代入到2）中计算出的模型，计算出这些点是否为该模型下的内群点，并记录纳入内群点的个数；
4. 重复上述步骤，迭代k次；
5. 比较上面k次迭代中计算出的模型，在步骤3）中记录下的内群点个数最多所对应的模型即为最终求解的模型。



1. 完成图像的SIFT特征提取与匹配：
2. 实验原理和思路：
3. 特征点提取（cv2.xfeatures2d.SIFT\_create（））：

在函数内部首先进行高斯模糊，得到不同blur程度的图片；然后进行下采样，获得多分辨图片，再用之前高斯模糊得到的图片减去下采样后的图片得到差分金字塔，利用DOG公式获得极值点，再用Taylor公式展开得到精确的极值点并消除边界点。

1. 计算得到关键点和关键点描述子（SIFT向量）（surf.detectAndCompute（））：

利用Sobel算子计算1）中特征点的梯度幅值和梯度方向，统计每个特征点的邻域的梯度方向，形成梯度直方图，得到每一个关键点的特征向量（如128维），利用匹配度量来筛选出特征点的主要方向。（详细过程见二）

1. 画出关键点（cv2.drawKeypoints（））：
2. 实验核心代码：

hessian = 300

surf = cv2.xfeatures2d.SIFT\_create(hessian)

kp1, des1 = surf.detectAndCompute(left\_gray, None)

img\_with\_drawKeyPoint\_left = cv2.drawKeypoints(left\_gray, kp1, np.array([]), flags=cv2.DRAW\_MATCHES\_FLAGS\_DRAW\_RICH\_KEYPOINTS)

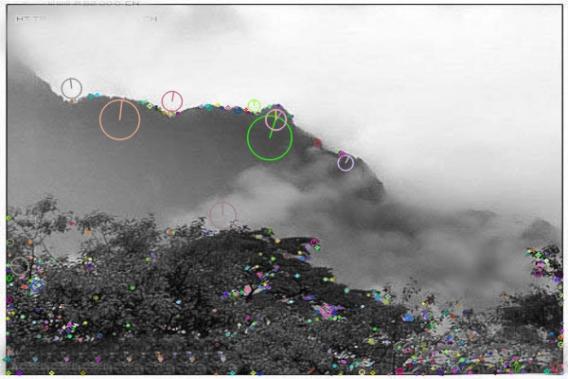
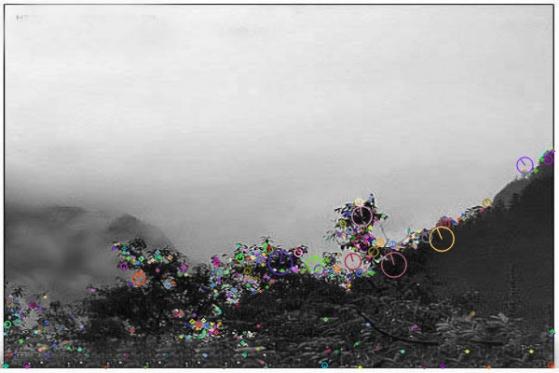
1. 实验结果分析：

下面第一行的两张图分别为输入图像的左图和右图，经过第二行的SIFT特征提取（圈出的点为提取出的SIFT特征点），得到第三行的特征匹配图像（即对第二行的特征点进行匹配连线）。代码中在cv2.xfeatures2d.SIFT\_create（）函数里面设定的阈值参数越小，能检测到的特征点越少。

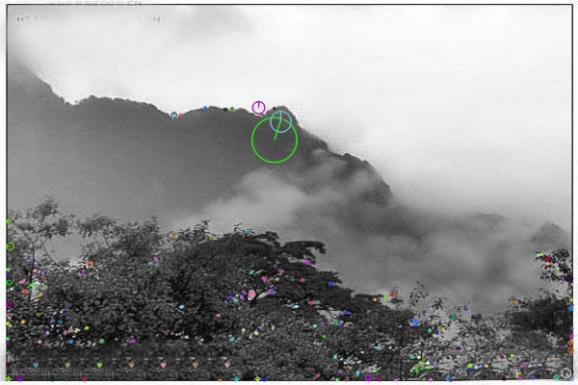
Left\_src\_img: Right\_src\_img:

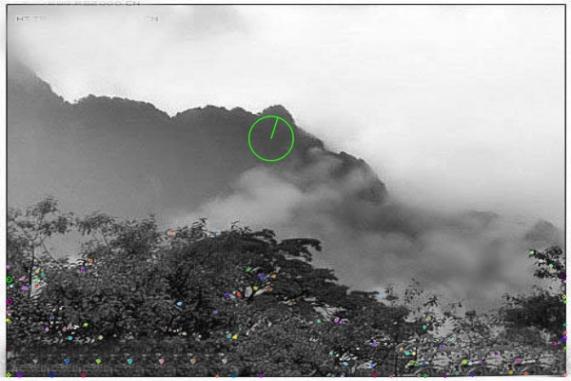
Left\_SIFT: threshold=500 Right\_SIFT:threshold=500

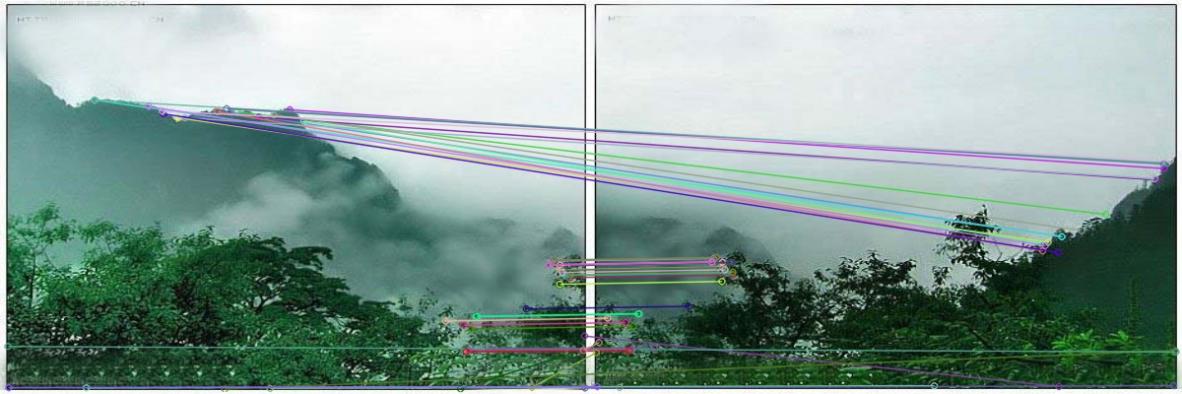
Left\_SIFT: threshold=300 Right\_SIFT:threshold=300

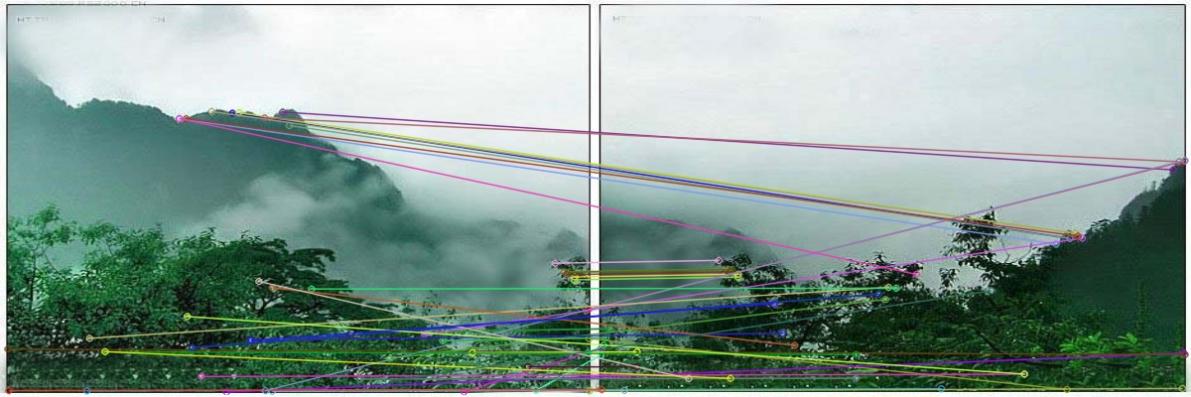
Left\_SIFT: threshold=150 Right\_SIFT:threshold=150

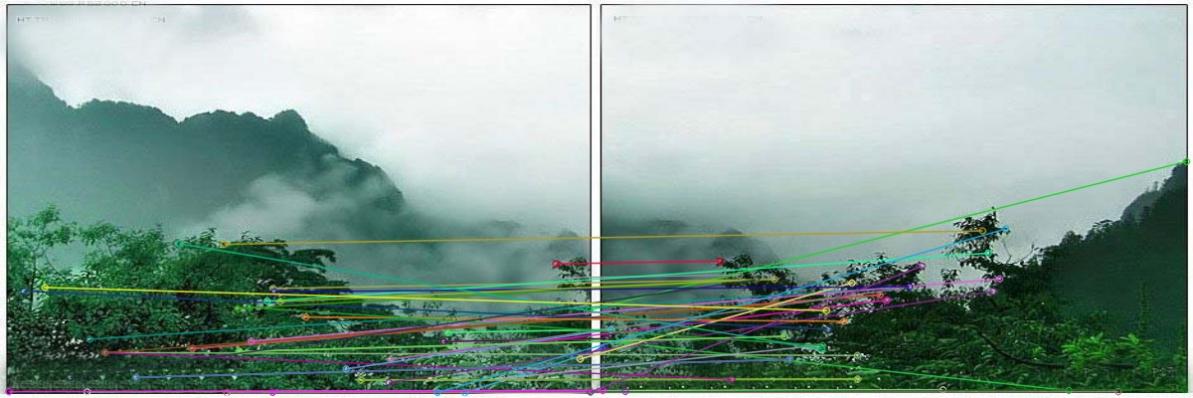
Match: threshold=500



Match: threshold=300

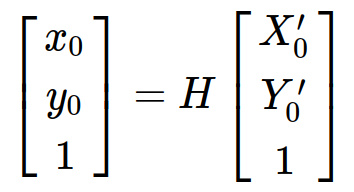
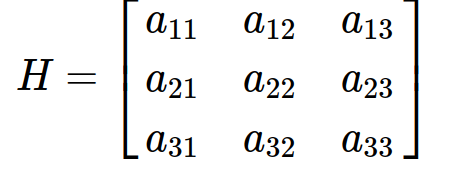


Match: threshold=150



1. 完成基于单应矩阵H（2D射影变换）的图像视点变换与拼接；讨论图像融合方法与鲁棒匹配/估计方法：
2. 实验原理和思路：

单应矩阵是指用两个不同的相机角度来拍摄同一个场景，在提取完两幅图片的匹配特征点后，利用这些匹配特征点的对应关系来计算得到一个矩阵，即单应矩阵（反映了两组坐标点的对应关系），可以用于将两张图片拼接到一起。

cv2.findHomography（）即为计算出单应矩阵的函数。

1. 实验核心代码：

import numpy as np

import cv2

left\_img = cv2.imdecode(np.fromfile('18.jpg', dtype=np.uint8), 1)

left\_img = cv2.resize(left\_img, (600, 400))

right\_img = cv2.imdecode(np.fromfile('17.jpg', dtype=np.uint8), 1)

right\_img = cv2.resize(right\_img, (600, 400))

left\_gray = cv2.cvtColor(left\_img, cv2.COLOR\_BGR2GRAY)

right\_gray = cv2.cvtColor(right\_img, cv2.COLOR\_BGR2GRAY)

hessian = 300

SIFT = cv2.xfeatures2d.SIFT\_create(hessian)

# surf = cv2.xfeatures2d.SIFT\_create()

keypoint1, descriptor1 = SIFT.detectAndCompute(left\_gray, None)

keypoint2, descriptor2 = SIFT.detectAndCompute(right\_gray, None)

drawKeyPoint\_left = cv2.drawKeypoints(left\_gray, keypoint1, np.array([]), flags=cv2.DRAW\_MATCHES\_FLAGS\_DRAW\_RICH\_KEYPOINTS)

cv2.imshow("img\_with\_drawKeyPoint\_left", drawKeyPoint\_left)

cv2.imwrite('left\_SIFT.jpeg',drawKeyPoint\_left)

drawKeyPoint\_right = cv2.drawKeypoints(right\_gray, keypoint2, np.array([]), flags=cv2.DRAW\_MATCHES\_FLAGS\_DRAW\_RICH\_KEYPOINTS)

cv2.imshow("img\_with\_drawKeyPoint\_right", drawKeyPoint\_right)

cv2.imwrite('right\_SIFT.jpeg',drawKeyPoint\_right)

bf = cv2.BFMatcher(cv2.NORM\_L1, crossCheck=True)

matches = bf.match(descriptor1,descriptor2)

matches =sorted(matches, key=lambda x:x.distance)

img3 = cv2.drawMatches(left\_img, keypoint1, right\_img, keypoint2, matches[:50], right\_img, flags=2)

cv2.imshow('img\_3',img3)

cv2.imwrite('match.jpeg',img3)

flann\_index=0

index=dict(algorithm=flann\_index,trees=5)

recursion=dict(iter=50)

flann\_match=cv2.FlannBasedMatcher(index,recursion)

key\_match=flann\_match.knnMatch(descriptor1,descriptor2,k=2)

feature\_point=[]

for i ,j in key\_match:

if i.distance<0.7\*j.distance:

feature\_point.append(i)

src\_descript=np.array([keypoint1[i.queryIdx].pt for i in feature\_point])

train\_descript=np.array([keypoint2[i.trainIdx].pt for i in feature\_point])

H=cv2.findHomography(src\_descript,train\_descript,cv2.RANSAC,5)

high1,wide1=left\_gray.shape[:2]

high2,wide2=right\_gray.shape[:2]

trans=np.array([[1.0, 0, wide1], [0, 1.0, 0], [0, 0, 1.0]])

touying=np.dot(trans,H[0])

train=cv2.warpPerspective(left\_img,touying,(wide1+wide2,max(high1,high2)))

cv2.imshow('left\_img', train)

cv2.imwrite('left.jpeg',train)

train[0:high2, wide1:wide1+wide2] = right\_img

cv2.imshow('total\_img', train)

cv2.imwrite('merge.jpg',train)

cv2.imshow('leftgray', left\_img)

cv2.imshow('rightgray', right\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. 实验结果分析:

下图是图像拼接后的生成图像，其下是匹配右图后左图作出的变换。由图可知：

（例如比较图中圆圈里面的叶子的拼接）

1. 当提取SIFT特征时设定的阈值越大，图像拼接效果越好，两张图片的连接处越平滑。
2. 在使用KNN算法时，设定保留最近距离比次近距离的m倍小的最近距离的点时，当m增大时，拼接连接处的图像内容连续程度越低。

Merge: threshold=500，m=0.7



Merge\_left:threshold=500，m=0.7



Merge: threshold=300，m=0.7



Merge\_left:threshold=300，m=0.7



Merge: threshold=300，m=0.4



Merge\_left:threshold=300，m=0.4



1. 讨论图像融合方法与鲁棒匹配/估计方法
2. 图像融合SIFT算法的优缺点：

优点：具有旋转、尺度缩放和亮度不变性，同时支持视角变化和放射变换在一定程度上的稳定性。该特征提取方法也适用于特征点较少的情况。

缺点：当图像内容纹理很少时，用SIFT提取出的128维特征向量的区别性很小，此时该算法很大程度上失效了。

1. RANSAC（鲁棒匹配）优缺点：

优点：可以鲁棒地估计模型参数，可以从大量离群点存在的情况下仍然能估计出较高精度的参数。

缺点：该算法本质上不满足最优性，即仅当迭代次数为无穷次时才能得到最优解，可见RANSAC算法得到精准模型的概率与迭代次数成正比。同时阈值的设定要与具体问题相关，因此要对不同问题进行调参。另外，该方法要求估计模型之前该模型是已知模型，而不能得到一个RANSAC本身不知道的模型。