电 子 科 技 大 学

实　验　报　告

学生姓名：刘文晨　　学 号：2018080901006　　指导教师：沈复民，徐行

一、实验项目名称：

局部特征匹配（Local Feature Matching）

二、实验原理：

1. 全局特征与局部特征

全局特征指的是图像的方差、颜色直方图等等，描绘了图像的整体信息，但是无法分辨出图像中的前景和背景。

局部特征指的是一些局部才会出现的特征，而且该部分需要满足两个条件：第一，能够稳定出现；第二，具有良好的可区分性。

当我们所关注的对象在图像中受到部分遮挡，全局特征可能会被破坏，但局部特征依然能够稳定存在，以代表这个物体。

2. 图像特征点

一幅图像中总存在着其独特的像素点，这些点我们可以认为就是这幅图像的特征，称为特征点。计算机视觉领域中的很重要的图像特征匹配就是一特征点为基础而进行的。好的特征应该有以下几个特点：重复性、可区分性、数量适宜、高定位、有效性。

3. 尺度不变特征变换（SIFT）

SIFT 算法常用于检测图像的局部性特征，在空间尺度中寻找极值点，提取这点的位置、尺度、旋转不变量。这些关键点是一些十分突出，不会因光照和噪音等因素而变化的点，如角点、边缘点、暗区的亮点及亮区的暗点等，所以与影像的大小和旋转无关，对光线、噪声、视角改变的容忍度也很高。

SIFT 算法分为四步：

1) 尺度空间的极值检测：搜索所有尺度空间上的图像，通过高斯微分函数来识别潜在的对尺度和选择不变的兴趣点。
2) 特征点定位：在每个候选的位置上，通过一个拟合精细模型来确定位置尺度，关键点的选取依据他们的稳定程度。
3) 特征方向赋值：基于图像局部的梯度方向，分配给每个关键点位置一个或多个方向，后续的所有操作都是对于关键点的方向、尺度和位置进行变换，从而提供这些特征的不变性。
4) 特征点描述：在每个特征点周围的邻域内，在选定的尺度上测量图像的局部梯度，这些梯度被变换成一种表示，这种表示允许比较大的局部形状的变形和光照变换。

三、实验目的：

实现图像的局部特征匹配，对于给定的两幅图像，均使用 Harris 角点检测器在图像中寻找角点，利用自适应非最大抑制（Adaptive Non-Maximum Supression）算法得到角的均匀分布，为每个关键点生成一个 128 维 SIFT 描述符。然后，以欧氏距离为参数，求出两幅图像特征之间的最佳匹配。最后，对具有地面真实性的图像进行评价。

四、实验内容：

1. 了解并学习局部特征匹配的知识和操作。
2. 实现读取图像并预处理、寻找角点、获得高分特征点、构建 SIFT 描述符、特征匹配等功能。

五、实验步骤：

1. 读取图像并预处理

读取图像，将图片的长和宽放缩为原来的 1/2，再转化为灰度图。

```
1. def setup_image(img_name):
```

```
    image = load_image('../data/'+img_name+'/'+img_name+'.jpg')#载入图像
    eval_file = '../data/'+img_name+'/'+img_name+'Eval.mat'#载入标准匹配矩阵
    scale_factor = 0.5 #设置缩放系数
    image = cv2.resize(image, (0, 0), fx=scale_factor, fy=scale_factor)#将图片进行缩放
    image_bw = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY) #进行灰度处理
    return image_bw
```

2. 使用 Harris Corner Detector 查找图像中的角点

1) 获得图片的尺寸（长和宽）

```
ImageRows = image.shape[0]
ImageColumns = image.shape[1]
```

2) 利用 Sobel 算子进行边缘检测

```
imX = cv2.Sobel(image, cv2.CV_64F,1,0,ksize=5)
imY = cv2.Sobel(image, cv2.CV_64F,0,1,ksize=5)
```

3) 计算 Harris 矩阵分量

```
Ixx = (Xderivative)*(Xderivative)
Iyy = (Yderivative)*(Yderivative)
Ixy = (Xderivative)*(Yderivative)
for i in range(16, ImageRows - 16):
        for j in range(16, ImageColumns - 16):
            Ixx1 = Ixx[i-1:i+1, j-1:j+1]
            Iyy1 = Iyy[i-1:i+1, j-1:j+1]
            Ixy1 = Ixy[i-1:i+1, j-1:j+1]
            Ixxsum = Ixx1.sum()
            Iyysum = Iyy1.sum()
            Ixysum = Ixy1.sum()
            Determinant = Ixxsum*Iyysum - Ixysum**2 #计算矩阵的特征值
            Trace = Ixxsum + Iyysum #计算矩阵的迹
            R = Determinant - alpha*(Trace**2) #角点相应函数 R
            #判断每个像素相应函数是否大于阈值 R，如果大于阈值则合格。
            if R > threshold:
                XCorners.append(j)
                YCorners.append(i)
                RValues.append(R)
XCorners = np.asarray(XCorners)
YCorners = np.asarray(YCorners)
RValues = np.asarray(RValues)
```

4) 角点过滤

```
NewCorners = ANMS(XCorners, YCorners, RValues, 3025)
NewCorners = np.asarray(NewCorners)
x = NewCorners[:,0]
y = NewCorners[:,1]
scales = NewCorners[:,2]

return x,y, scales
```

3. 使用 ANMS 进行角点筛选以获得高分特征点

```
def ANMS (x , y, r, maximum):
    i = 0
    j = 0
    NewList = []
    while i < len(x):
        minimum = 1000000000000
        #获得一个 harris 角点的横纵坐标，称为基础点
        X, Y = x[i], y[i]
        while j < len(x):#遍历除该角点外每一个得分（R 值）更高的角点，称为比较点，找到离基础点最近的一个比较点，记录下基础点的横纵坐标和与最近比较点之间的距离
            CX, CY = x[j], y[j]
            if (X != CX and Y != CY) and r[i] < r[j]:
                distance = math.sqrt((CX - X)**2 + (CY - Y)**2)
                if distance < minimum:
                    minimum = distance
            j = j + 1
        NewList.append([X, Y, minimum])
        i = i + 1
        j = 0
    #根据距离大小对基础点进行排序，很显然，距离越小的点说明在该角点周围有更好的角点，所以可以适当舍弃。在舍弃一定数目的得分较小的角点后，就得到了非最大抑制后的 harris 角点坐标。
    NewList.sort(key = lambda t: t[2])
    NewList = NewList[len(NewList)-maximum:len(NewList)]

    return NewList
```

4. 构建 SIFT 功能描述符

1) 使用高斯滤波器降低图像中的噪点，并获得图像的尺寸

```
filter1 = cv2.getGaussianKernel(ksize=4,sigma=10)
```

```
filter1 = np.dot(filter1, filter1.T)
image = cv2.filter2D(image, -1, filter1)
ImageRows = image.shape[0]
ImageColumns = image.shape[1]
xlen = len(x)
ylen = len(y)
FeatureVectorIn = np.ones((xlen,128))
NormalizedFeature = np.zeros((ylen,128))
```

2) 遍历 Harris 算法得到的每一个角点坐标，提取以该角点坐标为中心的 16*16 像素的一个局部图像，对于每一个局部图像，拆分成 16 个 4*4 像素的窗口，计算窗口内每个像素的大小和方向

```
for i in range(xlen):
    temp1 = int(x[i])
    temp2 = int(y[i])
    Window = image[temp2-8:temp2 + 8, temp1-8:temp1 + 8]
    WindowRows = Window.shape[0]
    WindowColumns = Window.shape[1]
    for p in range(4):
        for q in range(4):
            WindowCut = Window[p*4:p*4 +4,q*4: q*4+4]
            NewWindowCut = cv2.copyMakeBorder(WindowCut, 1, 1, 1, 1, cv2.BORDER_REFLECT)
            Magnitude = np.zeros((4,4))
            Orientation = np.zeros((4,4))
            for r in range(WindowCut.shape[0]):
                for s in range(WindowCut.shape[1]):
                    Magnitude[r,s] = math.sqrt((NewWindowCut[r+1,s] - NewWindowCut[r-1,s])**2 + (NewWindowCut[r,s+1] - NewWindowCut[r,s-1])**2)
                    Orientation[r,s] = np.arctan2((NewWindowCut[r+1,s] - NewWindowCut[r-1,s]),(NewWindowCut[r,s+1] - NewWindowCut[r,s-1]))
```

3) 对于每一个窗口，以方向为横坐标建立直方图（共 8 个方向），窗口内所有像素在方向上的加权和为直方的幅度值，这样就得到了每一个角点的局部图像中每个窗口表示大小和方向的特征向量

```
Magnitude = Magnitude
OrientationNew = Orientation*(180/(math.pi))
hist, edges = np.histogram(OrientationNew, bins = 8, range = (-180,180), weights = Magnitude)
for t in range(8):
```

```
    l = t+p*32+q*8
    FeatureVectorIn[i,l] = hist[t]
```

4) 正则化特征向量，并返回值

```
for a in range(FeatureVectorIn.shape[0]):
    sum1 = 0
    for b in range(FeatureVectorIn.shape[1]):
        sum1 = sum1 + (FeatureVectorIn[a][b])*(FeatureVectorIn[a][b])
    sum1 = math.sqrt(sum1)
  for c in range(FeatureVectorIn.shape[1]):
    NormalizedFeature[a][c] = FeatureVectorIn[a][c]/sum1
fv = NormalizedFeature

return fv
```

5. 特征匹配

1) 对于图片 1 的每一个 SIFT 特征描述符遍历图片 2 的每一个 SIFT 特征描述符，分别计算它们和图一特征向量之间的欧式距离

```
for x in range(features1.shape[0]):
    for y in range(features2.shape[0]):
        ExtractedRow1 = features1[[x],:]
        ExtractedRow2 = features2[[y],:]
        SubtractedRow = ExtractedRow1 - ExtractedRow2
        Square = SubtractedRow*SubtractedRow
        Sum = Square.sum()
        Sum = math.sqrt(Sum)
        Distance[x,y] = Sum
```

2) 按照距离大小对图片 2 的特征向量进行升序排序，取与图一特征向量距离最小的两个，若两者的距离只比小于阈值，则说明最佳的匹配效果较好，记录下图一和图二两个特征向量的坐标。若大于阈值则不操作。在对图一所有特征向量遍历完后，返回每一对需匹配的特征向量的坐标，和两者之间的距离

```
    IndexPosition = np.argsort(Distance[x,:])
    d1 = IndexPosition[0]
    d2 = IndexPosition[1]
    Position1 = Distance[x,d1]
    Position2 = Distance[x,d2]
```

```
6.      ratio = Position1/Position2
7.      if ratio<0.8:
8.          Hitx.append(x)
9.          Hity.append(d1)
10.         Value.append(Position1)
11. Xposition = np.asarray(Hitx)
12. Yposition = np.asarray(Hity)
13. matches = np.stack((Xposition,Yposition), axis = -1)
14. confidences = np.asarray(Value)
15.
16. return matches, confidences
```

六、实验数据及结果分析：

实验数据为 3 组图片，如图 3 所示。

图 1 Episcopal Gaudi 原始图像

图 2 Mount Rushmore 原始图像

图 3 Notre Dame 原始图像

实现了 Harris 边缘检测后，以 Notre Dame 为例，特征点可视化结果如图 4 所示。

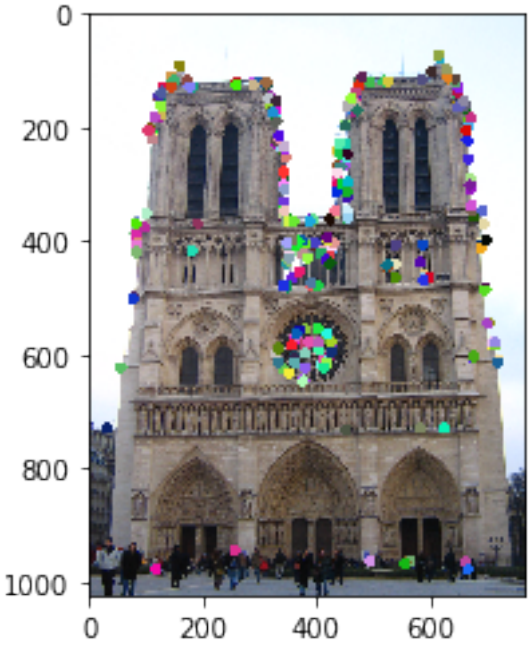


图 4 Notre Dame 特征点可视化结果

这张照片的关键是很多特征都聚集在一起，特别是在中心周围。为了解决

这个问题，需要使用非最大抑制算法（ANMS）进行角点过滤，过滤后得到的可视化图像如图 5 所示。因为每个特征点都是按它们在前一个特征点之前的距离来缩放的，所以这幅图中的特征点间隔得更远。

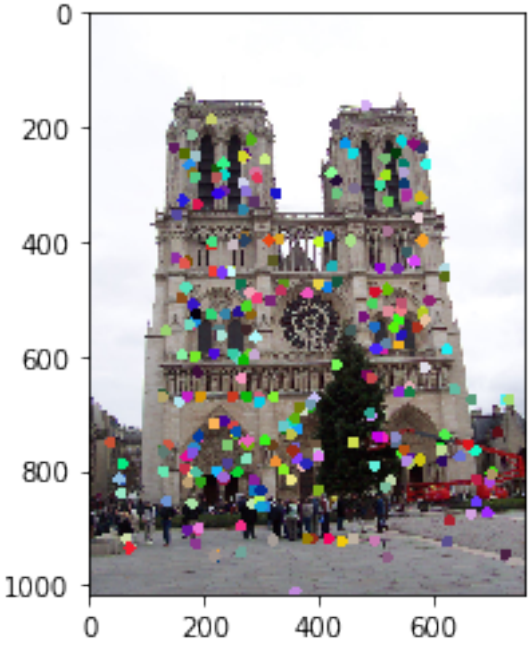
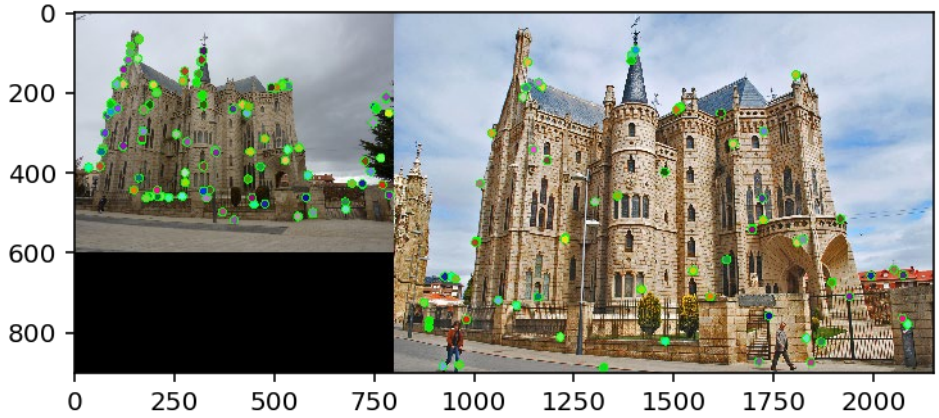
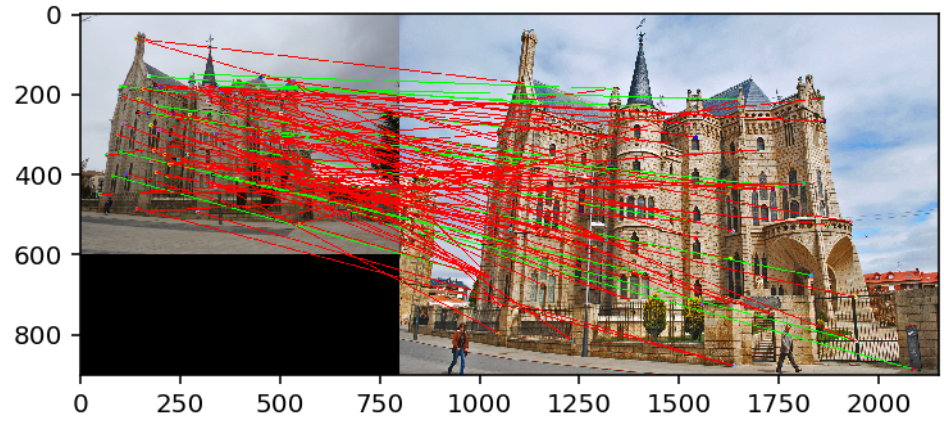
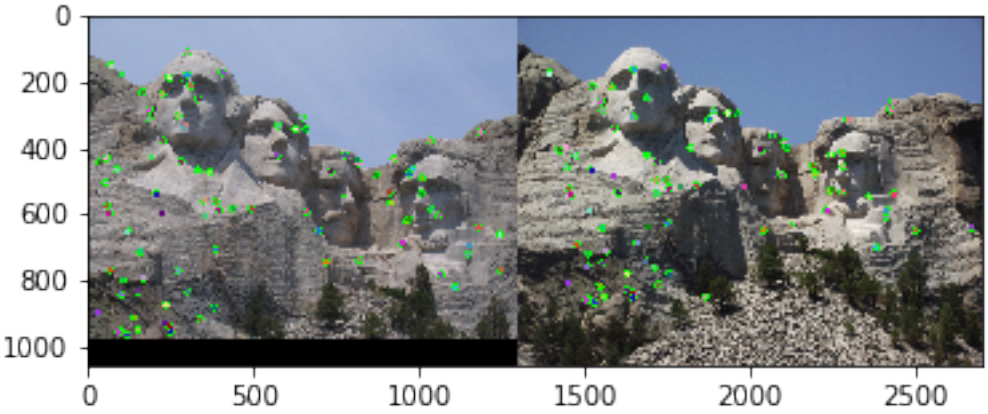


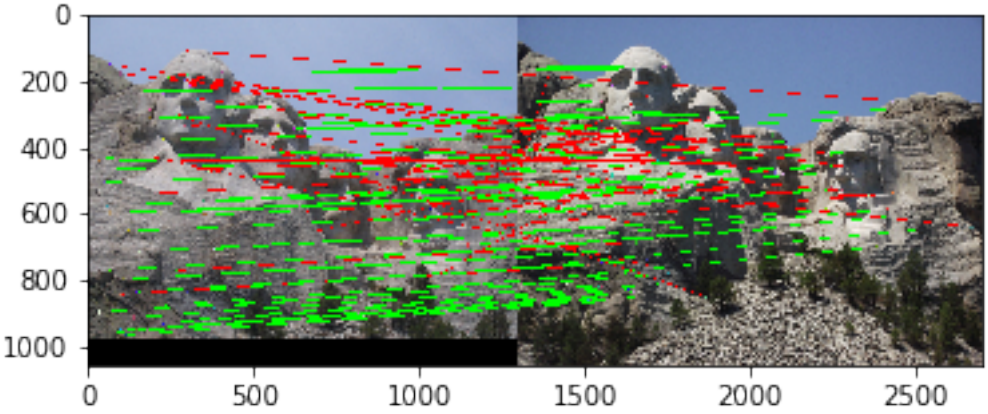
图 5 使用 ANMS 进行角点过滤后得到的可视化图像

使用 Harris 边缘点、ANMS、SIFT 和特征匹配的最终结果如图 6 所示。




0
200
400
600
800
0
250
500
750
1000
1250
1500
1750
2000


0
200
400
600
800
1000
0
500
1000
1500
2000
2500


0
200
400
600
800
1000
0
500
1000
1500
2000
2500

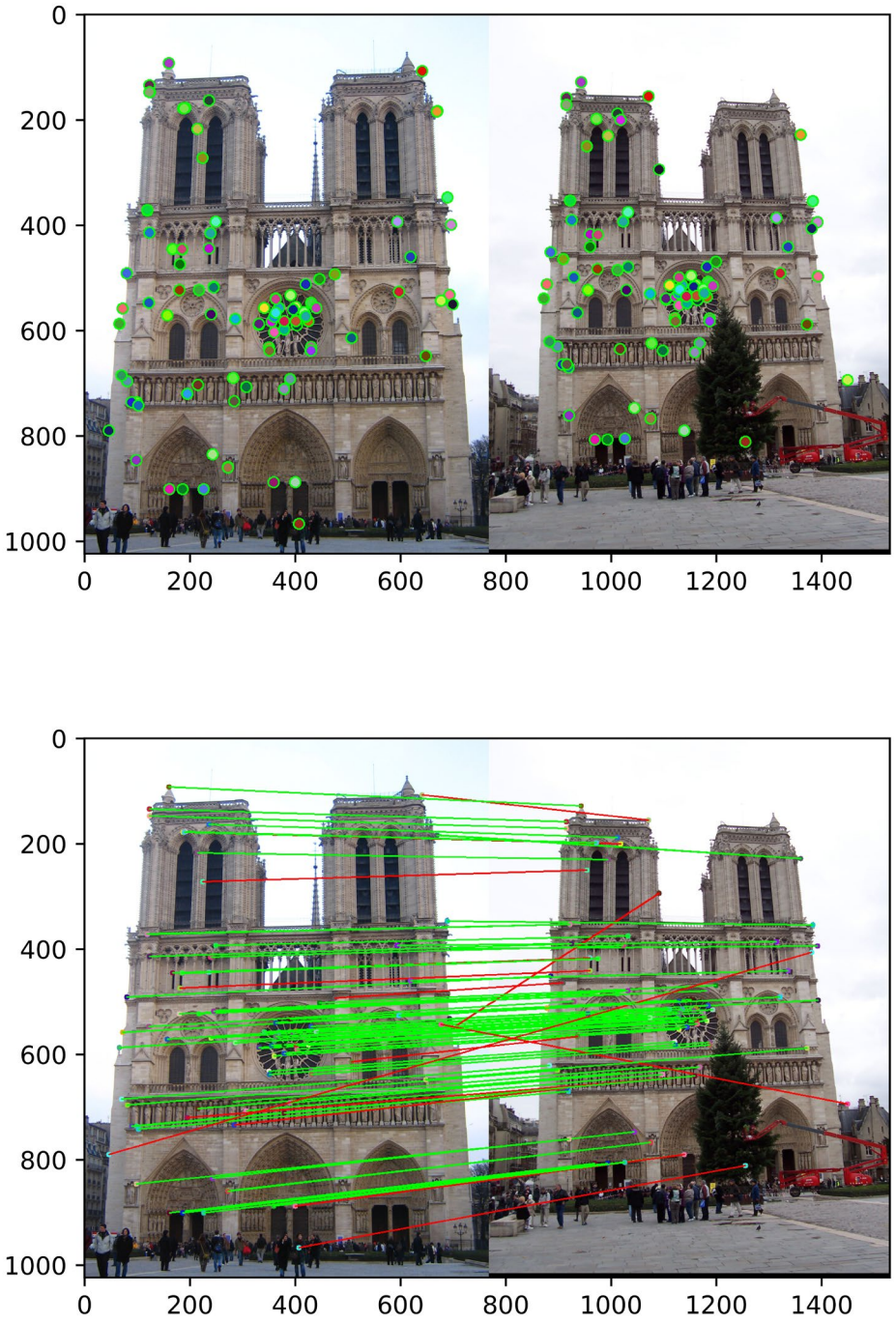


图 6 最终结果

其中，Episcopal Gaudi 的配对准确率为 10%，Mount Rushmore 的配对准确率为 65%，Notre Dame 的配对准确率为 86%。我认为 Episcopal Gaudi 准确率下降的主要原因是 Harris 对于不同尺度图片的边缘检测效果不佳。

接下来测试具有多个窗口宽度的 SIFT 算法。作为对照，我测试了 16、

28、36 和 40 的窗口特征大小，看看在没有 Harris 边缘检测和 AMNS 的情况下可以提高多少准确率。准确度结果如图 7 所示。

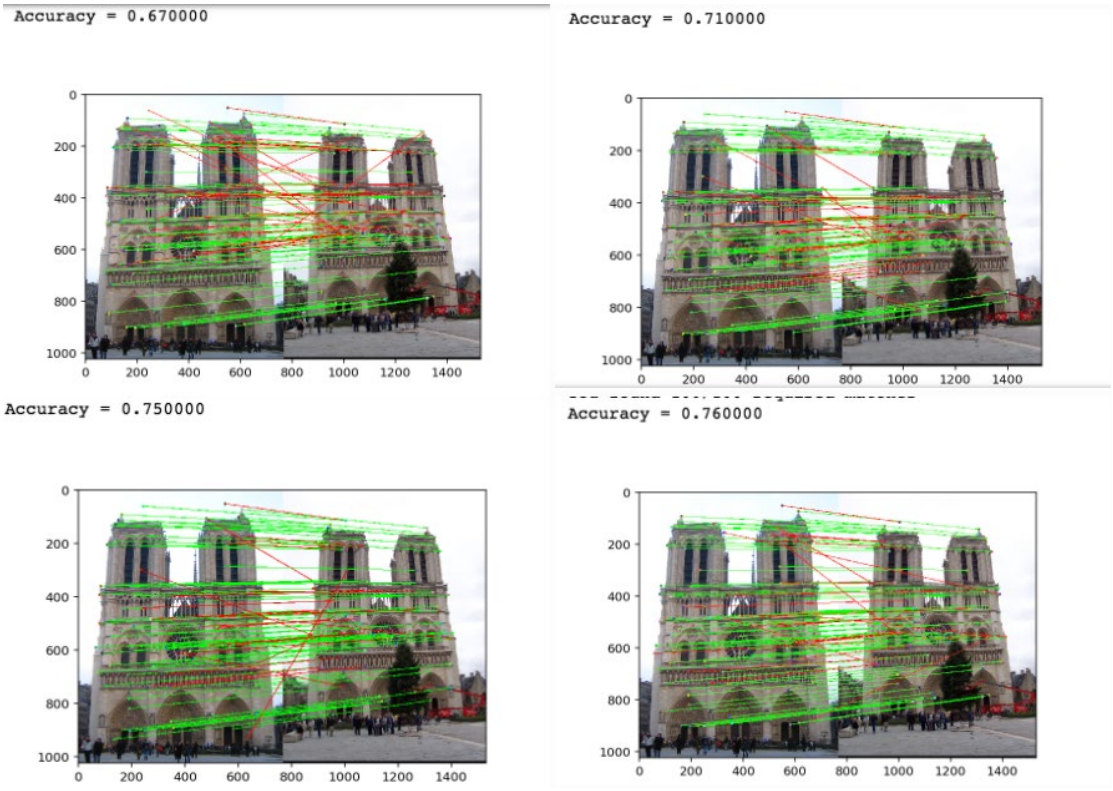


图 8 Notre Dame 准确度结果

我认为，更高的特征窗口大小和更好的精度之间肯定有关联。然而，我认为会有一个点，特征宽度变大将不再提高精度。这一点在 Mount Rushmore 中表现得尤为明显，如图 8 所示。

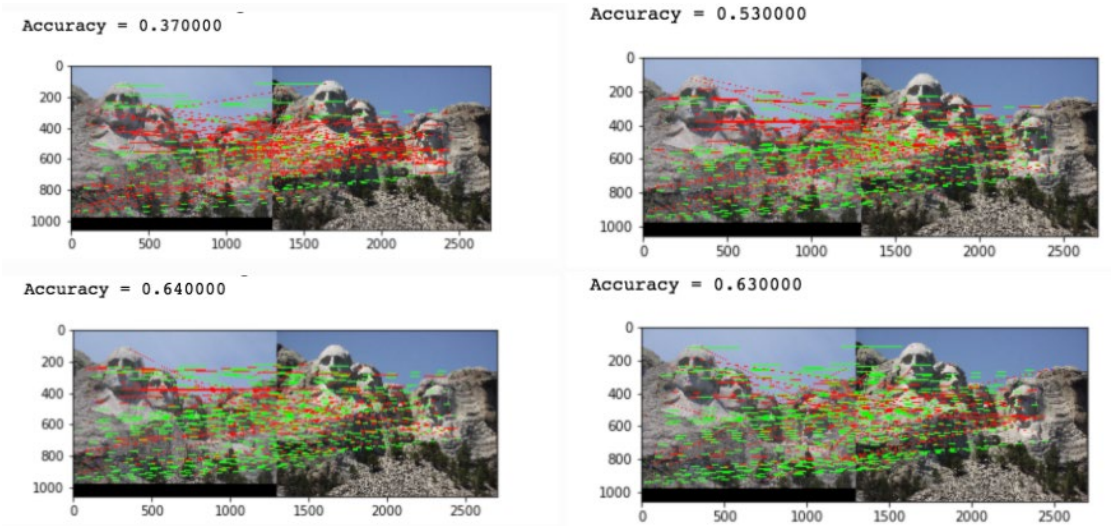


图 8 Mount Rushmore 准确度结果

七、实验结论：

以 Notre Dame 为例，在使用 Harris 角点检测器在图像中寻找角点，得到的角点分布是不均匀的，故利用 ANMS 得到角点的均匀分布，然后为每个关键点生成 128 维 SIFT 描述符。最后，以欧氏距离为参数，求出两幅图像特征之间的最佳匹配，如图 9 所示。这种方法提高了配对准确率，所以可以对具有地面真实性的图像进行评价。

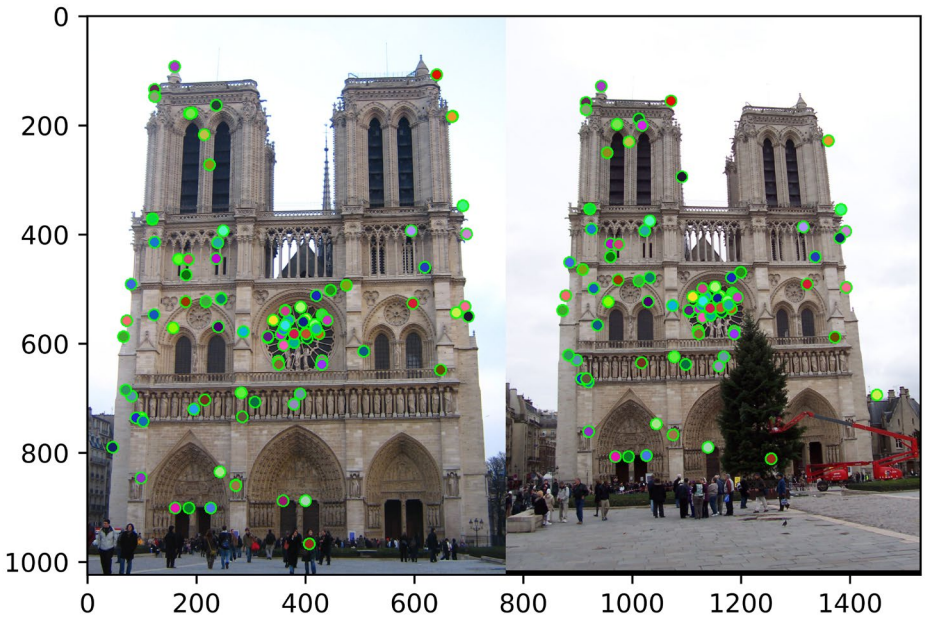


图 9 Notre Dame 两幅图像特征之间的最佳匹配

八、总结及心得体会：

本实验基本上实现了局部特征匹配，取得了较好的实验效果。实现读取图像并预处理、寻找角点、获得高分特征点、构建 SIFT 描述符、特征匹配等功能。

SIFT 通过考虑每个像素的大小和方向来唯一地描述每个关键点，该算法在没有旋转或缩放的图像上显示出良好的结果，可以用来匹配图像对。

最后还测试了具有多个窗口宽度的 SIFT 算法。

九、对本实验过程及方法的改进建议：

提供实验指导书。

报告评分：

指导教师签字：