赋值 2:局部特征匹配

概述

本任务的目标是使用 Szeliski 第 4.1 章中描述的技术创建一个局部特征匹配算法。我们建议的管道是著名的 SIFT 管道的简化版本。匹配管道旨在用于实例级匹配——同一物理场景的多个视图。

细节

对于这个项目，你需要实现局部特征匹配算法的三个主要步骤:

- SID< studententid >_harris.py 中的兴趣点检测(参见 Szeliski 4.1.1)
- SID< studententid >_sift.py 中的局部特征描述(参见 Szeliski 4.1.2)
- SID< studententid >_feature_matching.py 中的特征匹配(参见 Szeliski 4.1.3)

兴趣点检测(SID< studententid >_harris.py)
你将实现讲座材料和 Szeliski 4.1.1 中描述的 Harris 角点检测器。伪代码参见课本中的算法 4.1。入门代码给出了一些额外的建议。你不需要担心基线 Harris 角点检测器的尺度不变性或关键点方向估计。克里斯·哈里斯(Chris Harris)和迈克·斯蒂芬斯(Mike Stephens)描述他们角点检测器的原始论文可以在这里找到。

你还将实现自适应非极大值抑制。虽然大多数特征检测器只是在兴趣函数中寻找局部最大值，但这可能导致整个图像的特征点分布不均匀，例如，点在对比度较高的区域将更密集。为了缓解这个问题，Brown、Szeliski 和 Winder(2005)只检测那些既是局部最大值，而且其响应值显著(10%)大于半径 r 内所有邻居的响应值的特征。目标是只保留半径 r 像素邻域内的最大值点。一种方法是将所有的点按响应强度排序，从大到小的响应。列表中的第一个条目是全局最大值，它在任何半径上都不被抑制。然后，我们可以遍历列表，并计算到列表中它前面的每个兴趣点的距离(这些是响应强度更大的像素)。到一个关键点的更强邻居的最小距离(将这些邻居乘以>=1.1 以增加鲁棒性)是当前点处于局部最大值的半径。我们称这个为这个兴趣点的抑制半径，并保存这些抑制半径。最后，我们将抑制半径从大到小排序，并以这个排序后的顺序返回与前 n 个抑制半径相关联的 n 个关键点。大家可以随意用 n 做实验，我们用 n=1500。

局部特征描述(SID< studententid >_sift.py)
你将实现一个在讲座材料和 Szeliski 4.1.2 中描述的类似 sift 的局部特征。请参阅占位符 get_features()了解更多细节。如果你想让你的匹配管道快速工作(也许还可以帮助调试其他算法阶段)，你可能想要从规范化补丁作为你的本地特征开始。
特征匹配(SID< studententid >_sift.py)
你将实现讲座材料和 Szeliski 4.1.3 中描述的匹配局部特征的“比率测试”或“最近邻距离比率测试”方法。具体参见公式 4.18。最容易通过比率测试的潜在配对应该更倾向于正确配对——想想为什么。

使用入门代码(proj2.ipynb)

顶层项目 2。入门代码中提供的 `IPython notebook` 为你提供了文件处理、可视化和评估函

数，以及对上述三个函数的占位符版本的调用。无需修改就能运行入门代码，将在本页顶部显示的特定巴黎圣母院图像上随机匹配的兴趣点可视化。对应关系将通过 show_correspondence_circles()和 show_correspondence_lines()可视化(如果你喜欢，你可以注释其中一个或两个)。

对于巴黎圣母院图像对，在初始代码中也有一个 ground truth 评估。`Evaluate_correspondence()`将根据手工提供的匹配对对或错对每个匹配进行分类(有关详细信息，请参见 `show_ground_truth_corr()`)。起始代码还包含其他两个图像对(拉什莫尔山和高迪主教)的 ground truth 对应关系。你可以通过取消注释 proj2.ipynb 中适当的行来测试这些图像。

你可以使用在 annotate_mappings /collect_ground_truth_corr.py 中找到的通讯器().collect_ground_truth_corr()创建额外的真实值匹配(但这是一个繁琐的过程)。

在实现特征匹配流水线的过程中，应该可以看到 evaluate_correspondence()的性能有所提高。希望你发现这是有用的，但不要过度拟合初始的圣母院图像对，这是相对容易的。这里和起步代码中建议的基线算法将给你充分的信任，并在这些圣母院图像上工作得相当好，但 extra_data.zip 中提供的其他图像对就更难了。它们可能会表现出更多的视角、尺度和光照变化。如果你添加的数量足够多，你应该能够匹配更困难的图像对。

建议实施策略

强烈建议你按照下面的顺序实现这些函数:

- 首先，使用 cheat_interest_points()代替 get_interest_points()。这个函数只适用于 3 对具有 ground truth 对应关系的图像。这个函数不能在你的最终实现中使用。它直接从测试用例的 ground truth 对应中加载兴趣点。即使这样作弊，你的准确率最初也会接近零，因为起步代码特征都是零，起步代码匹配是随机的。
  `Get_interest_points()`返回非整数值，但你必须在整型坐标处剪切出补丁。你可以通过舍入坐标或进行某种形式的插值来解决这个问题。你自己的 get_features()也可以返回非整数坐标(许多方法确实尝试将兴趣点定位到亚像素坐标)。
- 其次，修改 get_features()，使其返回一个简单的特征。例如，从以每个兴趣点为中心的 16x16 个补丁开始。图像块并不是一个很好的特征(它们对亮度变化、对比度变化或小的空间偏移都不是不变的)，但这很容易实现，并提供了一个基线。你还不会看到你的准确性提高，因为 match_features()中的占位符代码是随机分配匹配。
- 第三，实现 match_features()。精度应该增加在巴黎圣母院~ 40%对如果你使用 16 x16(256 维)补丁功能,如果你只评估 100 年最自信的比赛。在其他测试用例上的准确率将较低(拉什莫尔山 25%，高迪圣公会 7%)。如果你按置信度对匹配进行排序(就像 match_features()中的启动代码所做的那样)，你应该注意到你更自信的匹配(更容易通过比率测试)更有可能是真正的匹配。
- 第四，通过实现一个类似 sift 的功能来完成 get_features()。如果你只评估你的 100 个最自信的匹配，准确率应该在 Notre Dame 对上提高到 70%，在 Mount Rushmore 上提高到 40%，在 Episcopal Gaudi 上提高到 15%。这些准确性仍然不是很好，因为人类从 cheat_interest_points()中选择的关键点可能不会根据你的特征进行特别好的匹配。
- 第五，停止使用 cheat_interest_points()，实现 get_interest_points()。Harris 角点没有我们知道对应的 ground-truth 点那么好，所以精度可能会下降。另一方面,你可以得

到数百甚至数千的兴趣点,这样你有更多的机会找到自信的比赛。如果你只在 Notre Dame 配对上评估最自信的 100 个匹配(参见 num_pts_to_evaluate 参数)，你应该能够达到 90%的准确率。只要你在巴黎圣母院图像对上的准确率为 80%，对于 100 个最自信的匹配，你就可以获得该项目的全部学分。当你实施自适应非最大抑制时，你的精度应该会提高得更多。

潜在有用的 NumPy 函数:np.arctan2()，np.sort()，np.()，np.()。`newaxis、np.argsort()、np.gradient()、np.histogram()、np.hypot()、np.fliplr()、np.flipud()、cv2.Sobel()、cv2.filter2D()、cv2.getGaussianKernel()、scipy.signal.convolve()`。

禁止功能(您可以使用测试,但不是在你的最终代码):cv2.SIFT (), cv2.SURF (), cv2.BFMatcher (), cv2.BFMatcher () .match (), cv2.FlannBasedMatcher () .knnMatch (), cv2.BFMatcher () .knnMatch (), cv2.HOGDescriptor (), cv2.cornerHarris (), cv2.FastFeatureDetector (), cv2.ORB (), skimage.feature, skimage.feature.hog (), skimage.feature.daisy, skimage.feature.corner_harris (), skimage.feature.corner_shi_tomasi (), skimage.feature.match_descriptors (), skimage.feature.ORB()。

我们还没有在这里列举所有可能的禁用函数，但是使用任何人的代码为你执行兴趣点检测、特征计算或特征匹配是禁止的。

标题

- +25 分:在 SID< studententid >_harris.py 中实现 Harris 角点检测器
- +10 pts:在 SID< studententid >_harris.py 中实现自适应非极大值抑制
- +35 pts:在 SID< studententid >_sift.py 中实现类似 sift 的局部特征
- +10 pts:实现 SID< studententid >_feature_matching.py 中的“比率检验”匹配。
- +20 分:Writeup 与几个局部特征匹配的例子。
- -5*n 分:每次不按照格式上的手牌说明操作，损失 5 分

交

这是非常重要的，如果你不按照说明做，你会被扣分。每次在第一次不按照说明做之后，你就会被扣 5 分。你上交的文件夹必须包含以下内容:
code/—包含本次分配的所有代码的目录
results/—包含你的结果(由 notebook 生成)的目录
你应该用你的 SID 替换文件名中的< studententid >。

不要在代码中使用绝对路径(例如/user/George/CV/proj1)。如果你使用绝对路径，你的代码会中断，你会因此丢分。就像起步代码已经做的那样，简单地使用相对路径。除非你已经添加了新的数据，否则不要打开/data/文件夹。通过黑板把你的项目以 zip 文件的形式交上去。你可以使用 python zip_submit .py 创建这个 zip 文件。