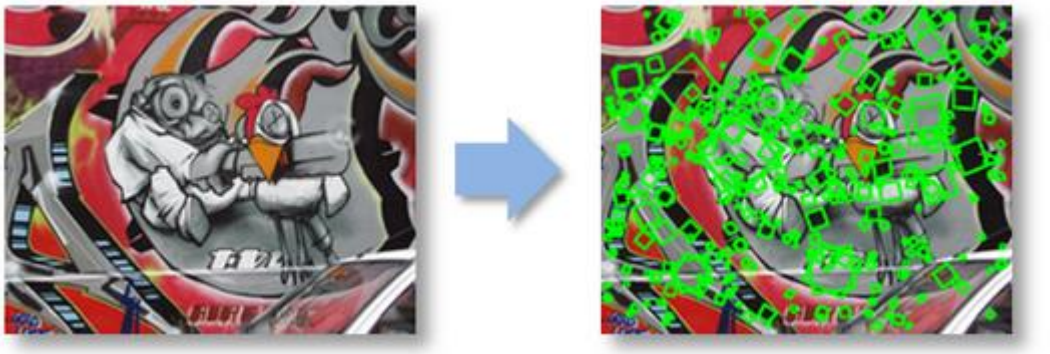
实验二：特征检测与匹配



0. 关键信息

本实验 2 人 1 组完成（报告中要注明两人工作量的比例）。

代码与报告提交信箱：visionexp@126.com

截止日期：2020 年 10 月 31 日 23 点 59 分

1. 概述

特征检测与匹配的目标是识别一个图像中的关键点与另一个图像中的对应点之间的配对。在此实验中，你将编写代码以检测图像中的特征点（对于平移、旋转和照明变化具有一定的不变性），并在另一个图像中找到最佳匹配特征。

为了帮你可视化结果并调试程序，我们提供了一个用户界面，可以显示检测到的特征和最佳匹配。我们还提供了一个示例 ORB 特征检测器，用于结果比较。

2. 实施细节

该实验有三个部分：特征检测、特征描述和特征匹配。您所需要实现的所有代码都在 **features.py** 中。

2.1 特征检测

你将使用 Harris 角点检测方法识别图像中的关键点，Harris 角点检测的详细信息请参阅课堂讲义（第 4 讲：Harris 角点检测）和论文（harris.pdf）。对于

图像中的每个点，考虑该点周围的像素窗口，计算该点的 Harris 矩阵 H，定义为

$$
\begin{aligned}
H &= \sum_p w_p \nabla I_p (\nabla I_p)^\top \\
&= \sum_p w_p \begin{pmatrix} I_{x_p}^2 & I_{x_p} I_{y_p} \\ I_{x_p} I_{y_p} & I_{y_p}^2 \end{pmatrix} \\
&= \sum_p \begin{pmatrix} w_p I_{x_p}^2 & w_p I_{x_p} I_{y_p} \\ w_p I_{x_p} I_{y_p} & w_p I_{y_p}^2 \end{pmatrix} \\
&= \begin{pmatrix} \sum_p w_p I_{x_p}^2 & \sum_p w_p I_{x_p} I_{y_p} \\ \sum_p w_p I_{x_p} I_{y_p} & \sum_p w_p I_{y_p}^2 \end{pmatrix}
\end{aligned}
$$

I_{x_p}是点 p 处的 x 方向导数，I_{y_p}是 y 方向导数。你应该使用 3×3 Sobel 算子计算 x 方向和 y 方向的导数（超出图像边界的点用 reflection 模式外推）。权重 w_p应该是圆对称的（实现旋转不变性），使用标准差为 0.5 的高斯卷积核（超出图像边界的点用 reflection 模式外推）实现。

注意，H 是 2x2 的矩阵。然后使用 H 计算每个像素处的角点强度函数 c(H)，$c(H) = det(H) - 0.1(trace(H))^2$。

我们还需要每个像素的度数方向。以梯度的方向作为近似方向。零角度指向右侧，正角度为逆时针方向。注意：不要通过结构张量的特征向量分析来计算方向。

我们将根据 c(H)选择最强的关键点，它们是 7×7 邻域中的局部最大值。

你需要补充完成类 HarrisKeypointDetector 中的三个函数，computeHarrisValues（TODO 1，为图像中每个像素计算 Harris 强度函数与方向）、computeLocalMaxima（TODO 2，计算布尔矩阵，指示每个像素是否是局部最大值）、detectKeypoints（TODO 3，根据像素的 Harris 强度与是否局部最大值生成特征点集合）。

可以使用以下库函数：

- scipy.ndimage.sobel：使用 Sobel 算子对输入图像滤波。
- scipy.ndimage.gaussian_filter：使用高斯卷积核对输入图像滤波。
- np.arctan2
- scipy.ndimage.filters.maximum_filter：使用最大过滤器过滤输入图像。
- scipy.ndimage.filters.convolve：使用选定的滤波器对输入图像滤波。

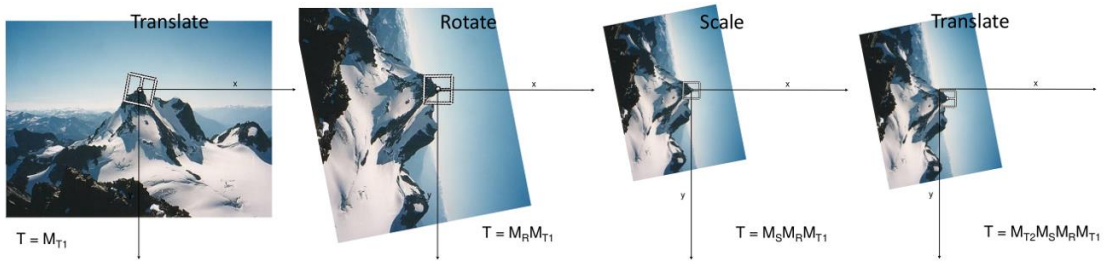
2.2 特征描述

下一步是为每个特征点提供一个描述符，此描述符将用于比较不同图像中的要素以查看它们是否匹配。

你将实现两个特征描述符，SimpleFeatureDescriptor 类和 MOPSFeaturesDescriptor 类。这些类的 describeFeatures 函数获取一组特征点的位置和方向信息，并计算这些特征点的描述符，然后将这些描述符存储在一个 numpy 二维数组中，该二维数组的行数为特征点的个数，列数为该特征描述符的大小（例如，对于 5×5 简单特征描述符，为 25）。

作为初学者，你将首先实现一个简单的描述符 SimpleFeatureDescriptor（TODO 4），它是 5×5 邻域中的像素强度值（即灰度值）。当所比较的图像是平移关系时应该可以正常工作。

其次，你将实现 MOPS 描述符的简化版本 MOPSFeaturesDescriptor（TODO 5）。MOPS 的详情可参考讲义（第 6 讲：特征描述符与特征匹配）和论文（MOPS.pdf）。你将计算从特征点周围的 40×40 像素区域子采样的 8×8 定向图像块。你必须提供一个变换矩阵，它将围绕特征点的 40×40 窗口变换为 8×8 图像块，使其特征点方向指向右侧。你将使用 cv2.warpAffine 函数实现变换。warpAffine 采用 2×3 前向卷绕的仿射矩阵，它从左边乘以原坐标使得变换后的坐标为列向量。生成 2×3 变换矩阵的最简单方法是组合多个转换：平移（T1）、旋转（R）、缩放（S）和平移（T2）。变换矩阵是矩阵乘积 T2×S×R×T1，下图说明了该序列。变换与卷绕的详情可参考讲义（第 7 讲：变换与卷绕），注意这些变换采用齐次坐标。你可能会发现 transformations.py 中的函数很有用，它们实现了 3D 仿射变换（你需要手动将它们转换为 2D 变换）。



你还应该将 8×8 图像块规范化为零均值和单位方差（TODO 6）。如果方差非常接近零（幅度小于 10^{-5}），那么你应该返回一个全零向量以避免除零错误。你可能会用到 np.std、np.mean 函数。

2.3 特征匹配

下一步是编写匹配它们的代码（即，在一个图像中给定一个特征，在另一个图像中找到最佳匹配特征）。最简单的方法如下：比较两个特征并计算它们之间的标量距离，最佳匹配是距离最小的特征。你将实现两个距离函数：

1) 差异平方和（SSD）：这是两个特征向量之间的欧氏距离平方。
2) 比率测试：通过 SSD 距离查找最近和次近的两个特征，比率测试距离是它们的比率（即，最接近的特征匹配的 SSD 距离除以第二接近的特征匹配的 SSD 距离）。

你将实现 SSDFeatureMatcher（TODO 7）和 RatioFeatureMatcher（TODO 8）的 matchFeatures 函数，该函数返回 cv2.DMatch 对象的列表。你应该将 queryIdx 属性设置为第一个图像中特征的索引，将 trainIdx 属性设置为第二个图

像中特征的索引，将距离属性设置为两个特征之间的距离（例如，SSD 或比率测试）。你可能会用到 scipy.spatial.distance.cdist 和 numpy.argmin 两个库函数。

3. 编码规则

可以使用 NumPy、SciPy 和 OpenCV2 函数来实现数学、滤波和变换操作。

不要直接使用关键点检测或特征匹配的函数。

使用 Sobel 算子或高斯滤波器时，应使用“reflect”模式，该模式在边缘处给出零梯度。

4. 测试和可视化

可以通过运行 tests.py 文件来测试你的 TODO 代码。这将加载一个小图像、运行你的代码并与正确的输出进行比较。这将允许你以递增的方式测试代码，而无需完成所有 TODO 块。

我们已经为 TODO 1-6 提供了测试代码。你应该为 TODO 7 和 8 设计自己的测试。最后，请注意提供的测试非常简单——它们可以帮助你入门。但是通过提供的测试用例并不意味着将通过评分测试用例。

通过运行 featuresUI.py，你将看到一个 UI，你可以进行特征点检测、特征匹配和基准测试。UI 是一种工具，可帮助你可视化特征点检测和匹配结果。请记住，你的代码将以数字方式进行评分，而不是以可视方式进行评分。

我们提供了一组基准图像，用于测试算法的性能，作为不同类型的受控变化（即旋转，缩放，照明，透视，模糊）的函数。对于这些图像中的每一个，我们都知道正确的变换，因此可以衡量每个特征匹配的准确性。这是使用我们在框架代码中提供的例程来完成的。你还应该出去拍摄自己的照片，看看你的方法在更有趣的数据集上的效果如何。例如，你可以拍摄几个不同对象（例如，书籍，办公室，建筑物等）的图像，并查看它的工作情况。

5. 所提供的文件

features.py：需要你实现的函数都在这个文件中。
tests.py：可以用该文件测试 TODO1-6。
featuresUI.py：提供了一个 GUI，可用来调试算法，并获得 benchmark 图像。
transformations.py：提供了 3D 仿射变换的矩阵获取函数。
resources 目录下是可用的测试图片。
harris.pdf：Harris 角点检测算法论文。
MOPS.pdf：MOPS 算法论文。

6. 实验环境配置

先在 Linux 或 Windows 上安装 Python3，之后安装 Numpy、Scipy、OpenCV for python。如果要运行 featuresUI.py，还要安装 PIL（Pillow）。

7. 提交文件

features.py：内含需要完成的 TODO1-8。

harris.png：每次计算 Harris 角点强度，都会自动生成一个 harris.png 文件，用 **resources** 目录下的 yosemite1.jpg 来生成此 harris.png 文件。

report(.docx,.pdf)：报告中应包括 **resources** 目录下的 yosemite 数据集的基准测试结果中的 ROC 曲线和 AUC。你可以通过运行 featuresUI.py，切换到“Benchmark”选项卡，按“Run Benchmark”，选择目录“resources/ yosemite”并等待一会儿来获得 ROC 曲线，同时 AUC 将显示在屏幕底部，可以通过按“Screenshot”保存 ROC 曲线。记录参数，如 Threshold，并使用 ROC 曲线和 AUC 来比较几种特征检测、描述及匹配方法。报告中应注明两人工作量的比例。

将以上文件打包压缩，压缩后的文件命名为“学号 1+姓名 1+学号 2+姓名 2+expX”;如“16030140095 李乐天+16030140012 王学斌 exp2.rar”、“16030140095 李乐天+16030140012 王学斌 exp2.zip”。Email 的标题类似命名。

8. 额外加分

三种方式可以获得额外加分：

（1） 在实现 MOPSFeatureDescriptor 类的 describeFeatures 方法时，加入尺度不变特性。一种方法可以参考 MOPS 论文（MOPS.pdf），你也可以设计其它方法。

（2） 实现 MOPS 论文（MOPS.pdf）中的自适应非最大抑制。

（3） 设计并实现其它的特征检测与描述方法，或对已有算法进行改进。报告你算法的性能，并分析提高性能的原因。如果最终的比率测试 AUC 提高 15%（AUC 提高 15%的意思是(1-AUC)减小 15%)以上,将被认可为实质性改进。

如果你希望有额外加分，除提交代码外，在报告中必须详细描述你的改进算法及性能提升结果。可以在 5 个数据集（bikes、graf、leuven、wall、yosemite）上运行 UI 基准测量你的平均 AUC。

简单的超参数更改将获得很小的额外加分，实质性的算法改进更容易获得认可。

在尝试额外加分之前，你首先应该完成基本任务。基本任务比额外加分更容易得分。

9. 调试中的注意事项

工程由 python2.7 转化而来，并在 python3.6 下经过测试。

以下三个问题是 opencv 的问题，不一定会被触发，哪怕大家用的都是 python3 也取决于你具体安装的 opencv 版本，解决方法如下：

1） 所有路径中均不允许存在中文，否则会报出 NoneType 错误；

2） 如果你的 ORB 不起作用并且报错——Incorrect type of self (must be 'Feature2D' or its derivative)，请将 features.py 文件中的 orb = cv2.ORB()替换为：orb = cv2.ORB_create()；反之亦然。

3） 如果 features.py 的以下代码报出错误——OpenCV error: (-215) K == 1 && update == 0 && mask.empty() in function cv::batchDistance，则进行调整：

```
class ORBFeatureMatcher(FeatureMatcher):

    ## ...(略)...



    def matchFeatures(self, desc1, desc2):

        return self.bf.match(desc1.astype(np.uint8), desc2.astype(np.uint8), k=1)    # 移除或修改这个 k 的值，让它直接通过断言。具体值取决于你的 OpenCV 和底层 C 语言库的版本。
```

其它杂项:

1） 不要在 Python2 里用 print('hello world')输出；不要在 Python3 中用 print 'hello world'输出。

2） 在 python3 中请使用 range()替代 xrange()；在 python2 中请使用 xrange()而不是 range()。

3） 有些算法的运行速度就是比较慢，如果你写了太多的 for 循环或者申请了太多的内存空间，请尝试优化你的代码。