模式识别团队作业

郑程耀 刘大晖 李传春李欢
（东南大学自动化学院，江苏 南京 210096）

The Team Course Assignment

（*School of Automation, Southeast University*）, *Nanjing* 210096, *China*）

1 概述

1.1 代码查重

在大学的计算机相关课程教授中，老师们常常会要求学生对于某个特定问题进行代码实现与探究；然而代码具有可移植性，如果同学们相互抄袭和借鉴，会妨碍授课者判断学生对于知识掌握程度。然而目前对于代码查重这一问题的研究并没有很多专门的理论和方法，也没有可用的开源代码。

本文尝试了一种基于图像方法的代码查重算法，可以快速判断代码中的重复部分，并且在改变代码中变量名以及代码块位置的情况下也具有很好的判断能力。

1.2 sift 算法

SIFT（尺度不变特征变换，Scale - Invariant Feature Transform）是在计算机视觉领域中检测和描述图像中局部特征的算法，该算法于 1999 年被 David Lowe 提出，并于 2004 年进行了补充和完善。该算法应用 SIFT 算法所检测到的特征是局部的，而且该特征对于图像的尺度和旋转能够保持不变性。同时，这些特征对于亮度变化具有很强的鲁棒性，对于噪声和视角的微小变化也能保持一定的稳定性。SIFT 特征还具有很强的可区分性，它们很容易被提取出来，并且即使在低概率的不匹配情况下也能够正确的识别出目标来。因此鲁棒性和可区分性是 SIFT 算法最主要的特点。用很广，如目标识别，自动导航，图像拼接，三维建模，手势识别，视频跟踪等。

Sift 算法一般分为以下几个步骤：

1）尺度空间极值检测：该阶段是在图像的全部尺度和全部位置上进行搜索，并通过应用高斯差分函数可以有效地识别出尺度不变性和旋转不变性的潜在特征点来；

2）特征点的定位：在每个候选特征点上，一个精细的模型被拟合出来用于确定特性点的位置和尺度。而特征点的最后选取依赖的是它们的稳定程度；

3）方向角度的确定：基于图像的局部梯度方向，为每个特性点分配一个或多个方向角度。所有后续的操作都是相对于所确定下来的特征点的角度、尺度和位置的基础上进行的，因此特征点具有这些角度、尺度和位置的不变性；

4）特征点的描述符：在所选定的尺度空间内，测量特征点邻域区域的局部图像梯度，将这些梯度转换成一种允许局部较大程度的形状变形和亮度变化的描述符形式。

本文自主实现了 sift 算法，并且按照应用进行了适当的调整，将这一特征用于特征匹配，以确定代码中的重合部分。

1.3 RANSAC 算法

RANSAC 为 Random Sample Consensus 的缩写，它是根据一组包含异常数据的样本数据集，计算出数据的数学模型参数，得到有效样本数据的算法。它于 1981 年由 Fischler 和 Bolles 最先提出[1]。

用来在一组包含离群的被观测数据中估算出数学模型的参数，是是一种不确定的算法——它有一定的概率得出一个合理的结果；为了提高概率必须提高迭代次数。RANSAC 算法的基本假设是样本中包含正确数据(inliers，可以被模型描述的数据)，也包含异常数据(outliers，偏离正常范围很远、无法适应数学模型的数据)，即数据集中含有噪声。同时 RANSAC 也假设，给定一组正确的数据，存在可以计算出符合这些数据的模型参数的方法。RANSAC 基本思想描述如下：

1）考虑一个最小抽样集的势为 n 的模型(n 为初始化模型参数所需的最小样本数)和一个样本集 P，集合 P 的样本数#(P)>n，从 P 中随机抽取包含 n 个样本的 P 的子集 S 初始化模型 M；

2）余集 SC=P\S 中与模型 M 的误差小于某一设定阈值 t 的样本集以及 S 构成 S*。S*认为是内点集，

它们构成 S 的一致集(Consensus Set)；

3）若#(S*)≥N，认为得到正确的模型参数，并利用集 S*(内点 inliers)采用最小二乘等方法重新计算新的模型 M*；重新随机抽取新的 S，重复以上过程。

4）在完成一定的抽样次数后，若未找到一致集则算法失败，否则选取抽样后得到的最大一致集判断内外点，算法结束。

在 RANSAC 算法中，需要进行调整的主要参数包括：最大迭代次数、进行模型拟合时选取的样本集数、最大距离 max_distance 设定、跳出迭代时的内点集阈值。

1.4 平行滑窗过滤算法

这是本文为解决代码图像匹配而提出的一种新的方法，在 ransac 剔除错配点之后，可以进一步去除剩下的几乎所有错配点。可以大大增加代码整体的可用性、稳健性。下图是错配点的典型情况，本算法对于此类问题具有很好的效果。

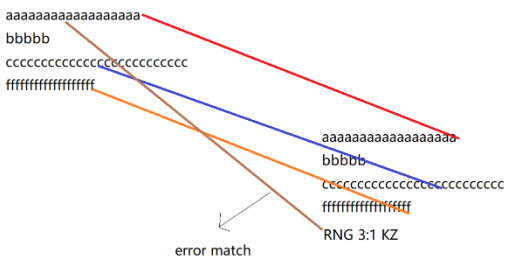

图 1.1 错配点典型情况

2 程序设计

本次实验环境依然为 ubuntu16.04，Qt Creator with GDB，c++。

2.1 sift 实现[2, 3]——尺度空间极值检测

特征点检测的第一步是能够识别出目标的位置和尺度，对于同一个目标在不同的视角下这些位置和尺度可以被重复的分配。并且这些检测到的位置是不随图像尺度的变化而改变的，因为它们是通过搜索所有尺度上的稳定特征得到的，所应用的工具就是被称为尺度空间的连续尺度函数。进行该操作的唯一方法是高斯模糊处理，因为已经被证实，高斯函数是唯一可能的尺度空间核。

图像的尺度空间用$\mathrm{L}(x,y,\sigma)$函数表示，它是由一个变尺度的高斯函数$G(x,y,\sigma)$与图像$I(x,y)$通过卷积产生，即

$$\mathrm{L}(x,y,\sigma)=G(x,y,\sigma)\otimes I(x,y)$$

其中，□ ⊗表示在 x 和 y 两个方向上进行卷积操作，而$G(x,y,\sigma)$为

$$G(x,y,\sigma)=\frac{1}{2\pi\sigma^2}e^{\frac{x^2+y^2}{2\sigma^2}}$$

σ是尺度空间因子，它决定着图像模糊平滑处理的程度。在大尺度下（σ值大）表现的是图像的概貌信息，在小尺度下（σ值小）表现的是图像的细节信息。因此大尺度对应着低分辨率，小尺度对应着高分辨率。(x,y)则表示在σ尺度下的图像像素坐标。

利用 LoG（高斯拉普拉斯方法，Laplacian of Gaussian），即图像的二阶导数，能够在不同的尺度下检测到图像的斑点特征，从而可以确定图像的特征点。但 LoG 的效率不高。因此 SIFT 算法进行了改进，通过对两个相邻高斯尺度空间的图像相减，得到一个 DoG（高斯差分，Difference of Gaussians）的响应值图像$D(x,y,\sigma)$来近似 LoG：

$$\begin{aligned}D(x,y,\sigma)&=\big(G(x,y,k\sigma)-G(x,y,\sigma)\big)\otimes I(x,y)\\&=L(x,y,k\sigma)-L(x,y,k\sigma)\end{aligned}$$

其中，k 为两个相邻尺度空间倍数的常数。可以证明 DoG 是对 LoG 的近似表示，并且用 DoG 代替 LoG 并不影响对图像斑点位置的检测。此处我本来一直不明白 DoG 能够更快的原因是什么，因为我理解的是这两个方法都是用同样大小的算子对图像进行卷积操作，后来在实际的代码中我发现其实高斯模糊（也就是高斯卷积）时一般采用 x y 两个方向分别做一维卷积，这样比二维卷积要快多了。因为如果可以把问题转化为高斯卷积的话，确实可以利用高斯卷积的性质进行大幅度的提速。

接下来，将在每一组（金字塔的每一层）内寻找极值点，方法很简单，就是比较一个点与上下尺度的 8*8 网格内所有点的大小，如果比他周围的 26 个点都要大，那么他是一个极值点。

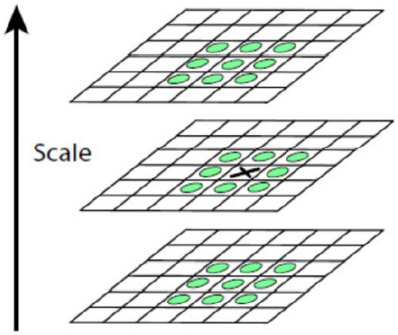

图 2.1 极值点确定

2.2 sift 实现——特征点的定位

通过上一步，我们得到了极值点，但这些极值点

还仅仅是候选的特征点，因为它们还存在一些不确定的因素。首先是极值点的搜索是在离散空间内进行的，并且这些离散空间还是经过不断的降采样得到的。如果把采样点拟合成曲面后我们会发现，原先的极值点并不是真正的极值点，也就是离散空间的极值点并不是连续空间的极值点。在这里，我们是需要精确定位特征点的位置和尺度的，也就是要达到亚像素精度，因此必须进行拟合处理。此处采用了泰勒级数展开，获得该极值点下的实际极值点偏移量$\hat{X}$，然后求得实际极值点的极值：

$$f\left(\hat{X}\right) = f\left(\hat{X}\right) + \frac{1}{2}\frac{\partial f^T}{\partial X}\hat{X}$$

设定一个阈值，当求出的极值大于该阈值时，才会保留下来，否则会剔除。在代码中，我采用了与原文相同的值 0.04。

2.3 sift 实现——方向角度的确定

经过上面两个步骤，一幅图像的特征点就可以完全找到，而且这些特征点是具有尺度不变性。但为了实现旋转不变性，还需要为特征点分配一个方向角度，也就是需要根据检测到的特征点所在的高斯尺度图像的局部结构求得一个方向基准。该高斯尺度图像的尺度 σ 是已知的，并且该尺度是相对于高斯金字塔所在组的基准层的尺度。而所谓局部结构指的是在高斯尺度图像中以特征点为中心，以 r 为半径的区域内计算所有像素梯度的幅角和幅值。像素梯度的幅值的计算公式为：

$$m(x,y) = \sqrt{L(x+1,y) - L(x-1,y)^2 + \left(L(x,y+1), L(x,y-1)\right)^2}$$

在完成特征点邻域范围内的梯度计算后，还要应用梯度方向直方图来统计邻域内像素的梯度方向所对应的幅值大小。这样，直方图的主峰值，即最高的那个柱体所代表的方向就是该特征点处邻域范围内图像梯度的主方向，也就是该特征点的主方向。

2.4 特征匹配对的筛选剔除

经过上面步骤，一幅图像 sift 特征就得到了，是一个 1*128 维的向量，首先用暴力匹配，找到每个点在另外一张图中对应的距离最近的点，接下来调用 ransac 方法筛选得到清洗后的结果。

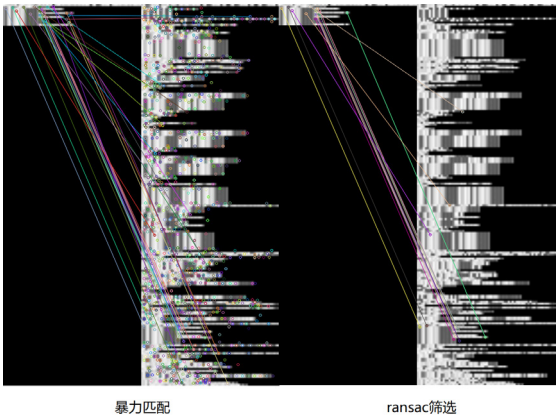

图 2.2 暴力匹配与 ransac 筛选

2.5 平行滑窗过滤算法

到此为止，我们已经得到了很好的效果，代码中存在抄袭时，就算只有一段，匹配对也达到了 30 以上。没有抄袭时则不会超过 20。但是我们还希望进一步提高区分度，彻底剔除掉错配点，于是平行滑窗过滤算法应运而生。我们注意到代码不同于图像中的物体，代码段具有统一的结构，因而如果是同样的代码段的话，匹配对应该是平行的，出现了不行平的线段则判定为错误。

如下图所示，根据斜率来判断匹配对是否合法：

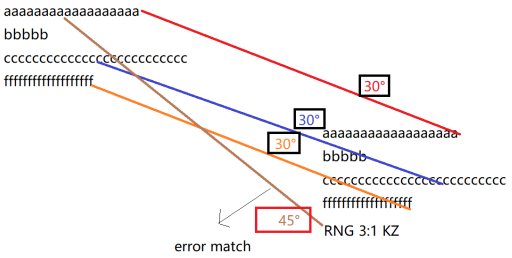

图 2.3 平行滑窗过滤算法原理

经过这一词的筛选，基本排除了所有的错配点，如下图所示：

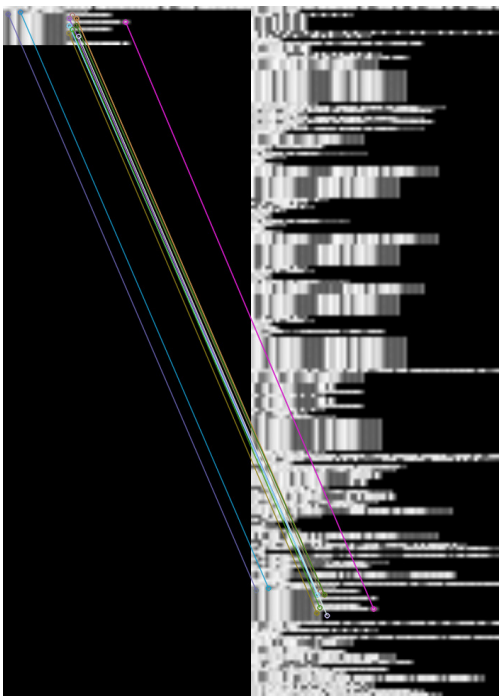

图 2.4 平行滑窗过滤效果

3 实验数据与结果

3.1 存在相同代码段时的匹配效果

存在相同代码段时，会检测出相同代码的对应位置，并且基本不包含错配点。

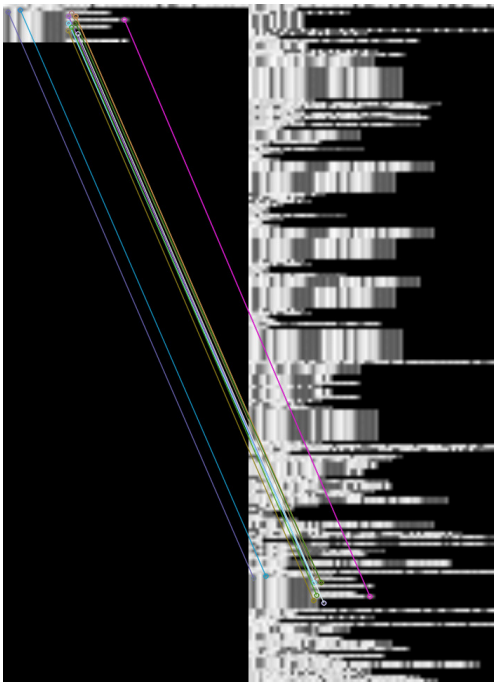

图 3.1 检测出 15 个匹配对

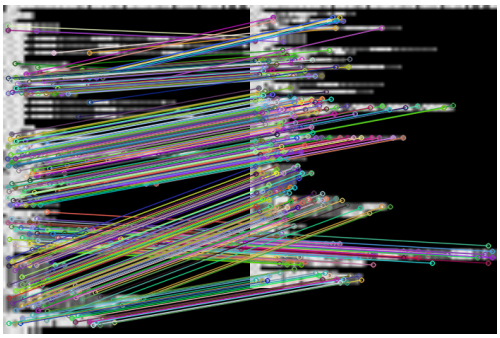

图 3.2 检测出 374 个匹配对

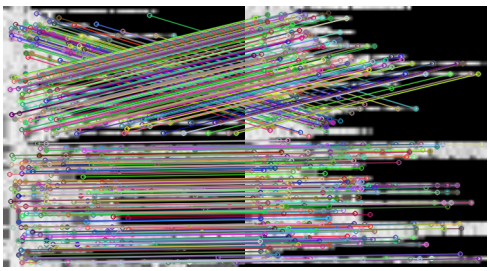

图 3.3 检测出 556 个匹配对

3.2 不存在相同代码段时的匹配效果

作为对比，我们测试了无抄袭现象代码段的匹配数目。下图是一个例子，没有任何抄袭现象的代码，暴力匹配得到 662 对，ransac 后剩余 36 对，平行过滤后则为 0 对。

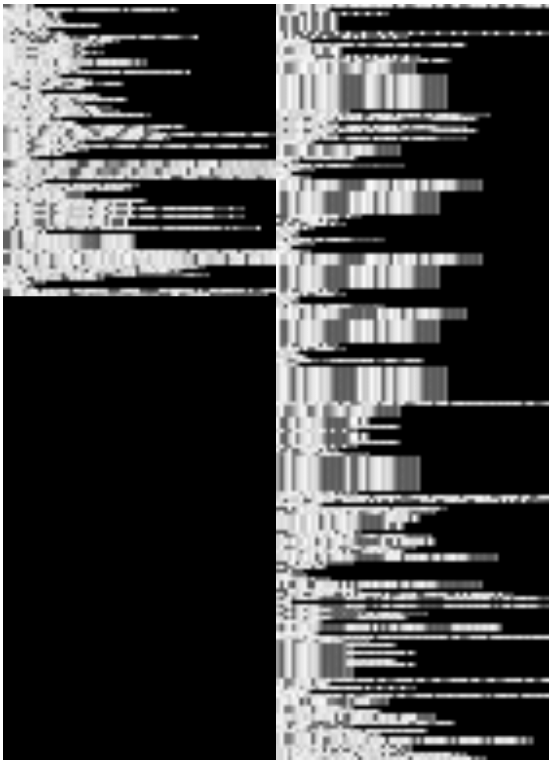

图 3.4 检测出 0 个匹配对

3.3 封装与功能增强

为了方便使用，我们进行了命令行功能增强，有 2 种使用模式，模式一可以比较两份代码相似度，使

用格式为 “–c code1 路径 code2 路径”。运行后生成匹配图，图片以匹配数命名：

图 3.5 模式一输出

模式二则为比较目录下两两文件夹内所有代码相似度，分为两步，首先生成 sift 特征的数据，保存下来。

图 3.6 模式二生成代码图像

然后遍历目录两两对比，输出可疑的代码，阈值是匹配数大于 10，可疑的两份代码会生成匹配图保存下来。

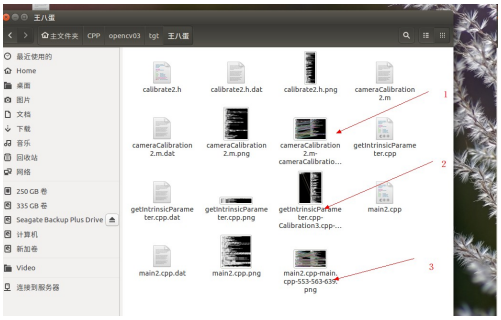

图 3.7 模式二生成可疑匹配图像

4 实验总结

经过实验，主要有以下几点结论：

1、SIFT 特征是图像的局部特征，其对旋转、尺度缩放、亮度变化保持不变性，对视角变化、仿射变换、噪声也保持一定程度的稳定性。

2、SIFT 特征独特性(Distinctiveness)好，信息量丰富，适用于在海量特征数据库中进行快速、准确的匹配。

3、SIFT 算法实现物体识别主要有三大工序，1、提取关键点；2、对关键点附加详细的信息（局部特征）也就是所谓的描述器；3、通过两方特征点（附带上特征向量的关键点）的两两比较找出相互匹配的若干对特征点，也就建立了景物间的对应关系。

4、ransac 算法可以用于特征匹配对的清洗和筛选，从而得到准确的匹配结果。

5、本文实现了用图像方法进行代码查重，取得了良好的结果，验证了猜想。也开辟出了一条代码查重的新方法。

参 考 文 献 (References)

[1] Fischler, M.A. and Bolles, R.C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Communications of the ACM , 24(6): 381–395, 1981

[2] Lowe D G. Object Recognition from Local Scale-Invariant Features[C]// The Proceedings of the Seventh IEEE International Conference on Computer Vision. IEEE, 2002:1150.

[3] Lowe D G. Distinctive Image Features from Scale-Invariant Keypoints[J]. International Journal of Computer Vision, 2004, 60(2):91-110.