分类号_____ 编 号_____

U D C_____ 密 级_____

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

本科生毕业设计（论文）

题　　目：OpenCV 中分形 ArUco 的集成实现

姓　　名：田伦硕

学　　号：12110203

院　　系：计算机科学与工程系

专　　业：计算机科学与技术

指导教师：于仕琪副教授

诚信承诺书

1. 本人郑重承诺所呈交的毕业设计（论文），是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料均真实可靠。

2. 除文中已经注明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的作品或成果。对本论文的研究作出重要贡献的个人和集体，均已在文中以明确的方式标明。

3. 本人承诺在毕业论文（设计）选题和研究内容过程中没有抄袭他人研究成果和伪造相关数据等行为。

4. 在毕业论文（设计）中对侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

作者签名：田伦硕

2025 年 6 月 6 日

OpenCV 中分形 ArUco 的集成实现

田伦硕

（计算机科学与工程系　指导教师：于仕琪）

[摘要] 本文围绕分形 ArUco 标记的检测算法优化及其在 OpenCV 中的集成展开研究。传统 ArUco 标记在遮挡和远距离场景下鲁棒性不足；分形 ArUco 通过嵌套式 ArUco 码设计，结合多阶段检测算法，显著提升了复杂环境下的检测和姿态估计能力。研究首先分析了分形 ArUco 的结构特性与检测原理，其中重点探讨了基于 FAST 特征点检测和 FLANN 匹配的关键技术，并针对现有实现中的重复特征点问题提出了基于距离筛选的优化策略。在 OpenCV 集成方案中，以 OpenCV 内置 FLANN 模块替换第三方库 picoflann，在保证性能的前提下降低了代码复杂度。实验对比表明，优化后的 OpenCV 版本在计算效率（耗时仅增加 0.28%~1.7%）接近原版 nanofractal.h 实现的同时，特征点匹配数量提升最高达 28%，且错误匹配率极低。此外，研究还验证了分形 ArUco 在遮挡场景下的有效性。但未来工作需进一步优化特征点过滤与分类算法以减少误匹配。

[关键词]计算机视觉；机器学习；姿态估计

[ABSTRACT] This paper focuses on the optimization of fractal ArUco marker detection and its integration into OpenCV. Traditional ArUco marker lacks robustness in occluded and distant scenes. By proposing a nested marker design combined with a multi-stage detection algorithm, fractal ArUco significantly improves pose estimation capabilities in complex environments. The paper first analyzes the structural characteristics and detection principles of fractal ArUco, especially key techniques such as FAST feature point detection and FLANN-based matching. An optimized filtering strategy based on distance is proposed to resolve the issue of duplicate feature points matching in existing implementations. For the OpenCV integration, the study substitutes the third-party library picoflann with OpenCV's built-in FLANN module, reducing code complexity while maintaining performance. Experimental comparisons demonstrate that the optimized OpenCV version achieves computational efficiency comparable to the original nanofractal.h implementation (with only a 0.28%~1.7% increase in processing time) while improving feature point matching by up to 28%, with minimal error rates. Additionally, the study validates the effectiveness of fractal ArUco in occluded scenarios. But further optimization of feature point filtering and classification algorithms is demanded to reduce mismatches.

[Keywords] Computer Vision; Machine Learning; Pose Estimation

目录

1. 绪论

1.1 ArUco 的应用背景

姿态估计在许多计算机视觉应用中具有重要意义，例如机器人导航[1]、增强现实[2]等。该过程的核心在于建立真实环境中的点与其在 2D 图像投影之间的对应关系。由于这一步骤通常较为复杂，因此通常会使用合成标记或基准标记（fiducial markers）[14]来简化流程。

1.1.1 传统 ArUco 的应用背景

其中，Rafael Muñoz 和 Sergio Garrido 提出并开发的 ArUco[2]标记是最流行的基准标记之一。ArUco 码的设计基于二进制方形基准标记，依靠单个标记提供的黑色边框的四个角点便可以估计相机姿态。其内部的二进制编码设计使其具备较强的鲁棒性，可以通过选择与原码海明距离较小的候选者实现纠错机制。

图 1 ArUco 标记在机械臂定位场景的应用

1.1.2 分形 ArUco 的应用背景

传统的 ArUco 标记在机器人导航[1]、无人机定位[3]和增强现实等场景中已经广泛被用于姿态估计，如图 1 所示。由于传统 ArUco 标记检测依赖于边框检测，得到边框的角点坐标进一步去进行姿态估计，因此这种方法存在一些局限性：

（1）抗遮挡能力差：现有检测方法在遮挡情况下效果不佳，只要标记有一个边框的角点被遮挡，就会导致无法检测。

（2）检测距离受限：其可检测范围受标记尺寸限制。大型标记虽可在远距离被检测，但当相机靠近时，可能因无法完整进入视野而失效；小型标记则难以

在远距离被检测。

为此，ArUco 开发者团队提出了一种新型标记，分形 ArUco（Fractal ArUco）[4]。同传统 ArUco 相比，分形 ArUco 标记在设计上更复杂，由多个不同尺寸的 ArUco 标记嵌套组成，而且不局限于依赖边框角点来进行位姿估计，因此能够在不同距离及部分遮挡情况下实现鲁棒检测。鉴于其日益增长的使用需求（如无人机配送中的姿态估计），将分形 ArUco 标记功能集成至 OpenCV 库是一项对开发者社区有益的工作。

1.2 OpenCV 集成情况

由于 ArUco 标记的强大功能，其在开发初期便被移植至 OpenCV_contrib 工具库作为附加功能，并持续优化，最终于 2022 年 12 月正式合并进入 OpenCV 主库的 objdetect 模块中。当前 OpenCV 版本的 ArUco 模块已具备良好的性能，因此 OpenCV 已经多年没有对 ArUco 模块进行新功能的引入更新。然而截至目前，ArUco 开发者团队已经在 ArUco 方向有多项新的设计实现，有些已经有了广泛的应用，比如分形 ArUco。此外，本年度的 GSoC 与 OpenCV 合作提出的待解决问题中就有分形 ArUco 功能的合并。

1.3 研究目标

本研究旨在将分形 ArUco 标记的功能集成到 OpenCV 库的 objdetect 模块中。最基本的目标是在消除外部依赖的前提下，找到一种用户友好，结构清晰的工程设计方案[15]。此外，本研究不局限于工程上的设计实现，也会对分形 ArUco 功能的实现进行尽可能的优化。在达成这些目标之前，需要先深入了解 OpenCV 库的架构合分形 ArUco 功能的原理。此外，在完成集成后，需要设计一套针对该任务的测试数据集和评判的标准评来评估现有实现方案和集成优化后的实现方案。最终的结果应向 OpenCV 社区提供一个文档完善、高性能且扩展性强的解决方案。本研究使计算机视觉领域和机器人领域的开发者和研究人员都能受益。

1.4 文章结构

本文后续章节概括如下：第二章为分形 ArUco 标记理论技术研究，详细分析分形 ArUco 的结构特性与检测原理，重点阐述多阶段检测算法的实现过程，包括传统边角检测、特征点投影、子像素级细化特征点检测与匹配等关键技术，

并对现有分形 ArUco 库的技术实现进行对比分析；第三章为 OpenCV 集成方案设计与实现，首先提出模块架构设计方案，然后详细说明 FLANN[13]模块替换的具体实现方法，最后介绍重复特征点异常的修复方案；第四章为实验分析与验证，首先介绍实验环境和数据集，接着通过设计对比实验，从计算效率、特征匹配数量和真阳率三个维度，对优化前后的算法性能进行全面评估；第五章为结论与展望，总结本文的研究成果，并指出未来可能的改进方向。

2. 分形 ArUco 标记理论技术研究

2.1 分形 ArUco 结构特性

在 Fractal Markers: A New Approach for Long-Range Marker Pose Estimation Under Occlusion[4]这篇文章中，作者对分形 ArUco 的原理进行了解释。分形标记采用多个 ArUco 码嵌套的独特设计，每个标记由多个方形子标记组成。标记包含三个关键部分：用于快速检测和定位的黑色边框、存储二进制 ID（黑白方格代表 0 和 1）的编码区域以及确保内层标记独立检测的白色隔离带，如图 2，图 3 所示。

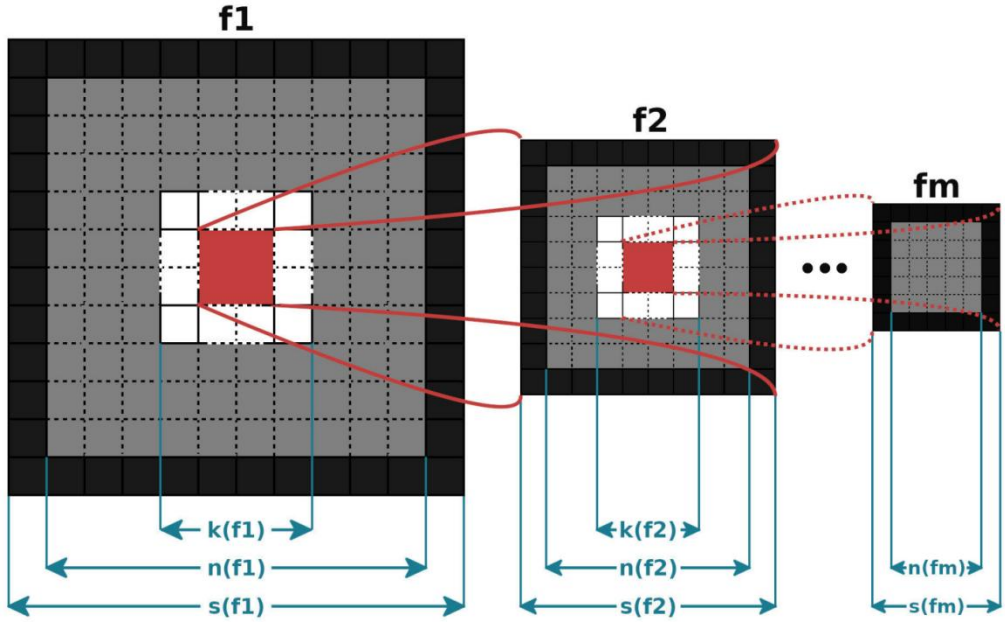

图 2 分形标记的通用结构中，每个标记由可划分为三个单元组成：黑色边框构成标记外缘，灰色单元区域用于配置并唯一确定标记 id，而白色隔离带则用于辅助内层标记的检测[4]

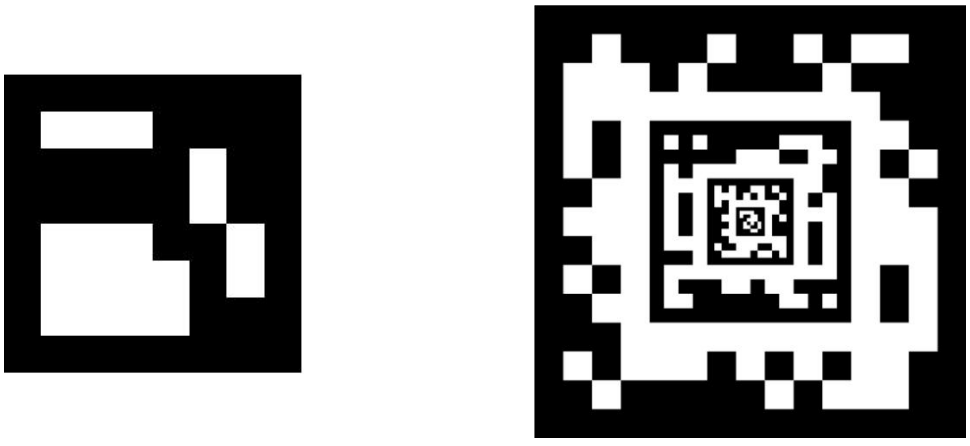

图 3 左为传统 ArUco 码，右为四层嵌套分形 ArUco 码示例

2.2 多阶段检测算法

分形 ArUco 的检测方法通过多阶段协同策略实现了对标记的高鲁棒性识别。整个检测流程大致分为三个阶段：首先是传统的边角检测，用以初步定位可能存在的标记区域并提取初始角点；接着在图像金字塔上对角点进行多尺度细化，提高定位精度并剔除不可靠点；最后，引入特征点检测与匹配机制，充分利用编码区域内部的图案结构，通过几何约束与统计优化相结合的方法，在多视角甚至部分遮挡情况下实现更稳定的位姿估计。这一体系化流程显著提升了分形 ArUco 在复杂场景下的识别能力。

2.2.1 传统边角检测

如果原始图像是彩色的话，图像预处理阶段将输入图像转换为灰度图，如图4(b)所示。接着使用自适应阈值[18]处理来对灰度图进行二值化，以此突出显示标记的边缘。这里的自适应阈值通过动态调整窗口大小来适应不同分辨率的图像，然后计算窗口平均灰度值，进而作为阈值，如图 4(c)所示。这一步骤的目的是为后续的轮廓检测提供清晰的二值图像，这样的轮廓更容易被识别和处理。

接下来是轮廓检测与初步筛选阶段。算法查找整个图像中可能的所有轮廓，并对每个轮廓执行几何分析，这一步骤用于筛选可能的标记候选。首先，系统会去除小于所需标记大小的轮廓，然后使用多边形逼近算法去近似轮廓以确定其是否可以被近似为凸四边形，如图 4(d)所示。通过验证所有轮廓，满足要求条件的轮廓角点将被排列以适用于处理阶段。

在分形标记解码阶段，系统将每个候选四边形区域透视映射到规范化视图，并提取其内部的比特矩阵。算法使用预先定义的编码规格生成比特矩阵，并从每个比特适用的双线性插值取像素值。之后，根据矩阵的平均亮度将其量化为二元，并调用解码函数，以验证比特模式是否符合已知的标记编码。如果解码成功，算法会调整角点顺序以对齐标记方向，并将有效的候选者添加入列表。

然后是候选标记去重阶段，通过调用排序函数，首先将候选列表按标记 ID 排序，同 ID 的标记则根据周长去掉一个。之后利用标准库的去重函数，筛除掉多余的同特征的候选标记，最后保证每个标记最多只出现一次。这一过程性操作的目的在于防止同一标记反复被检测出来，保证输出为唯一最优解。

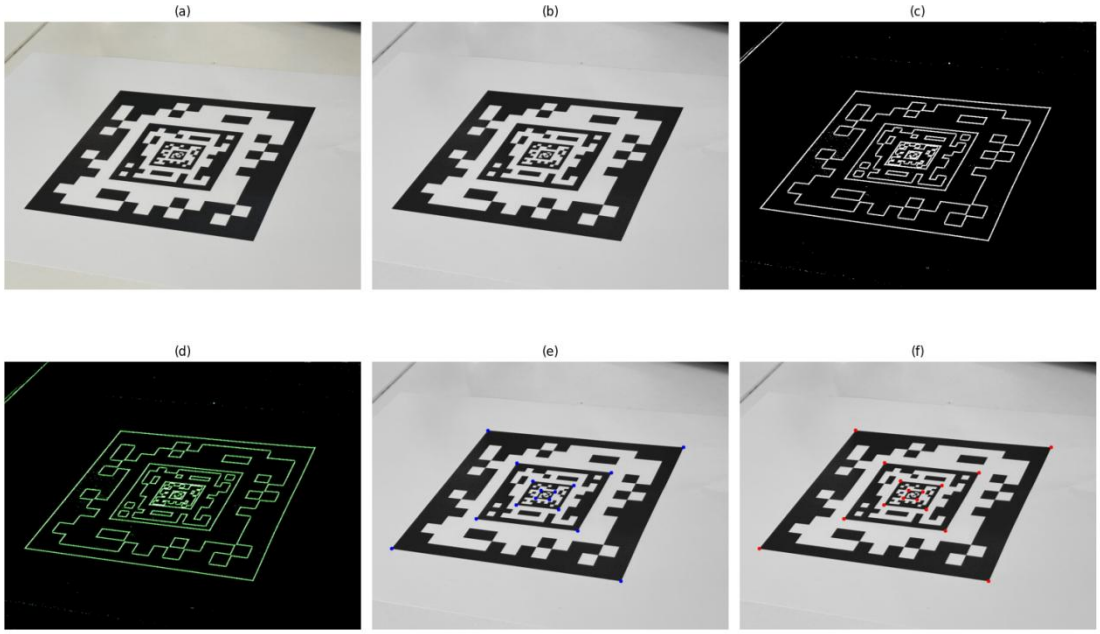

图 4 传统 ArUco 角点检测流程 (a)分形 ArUco 码输入原始图像 (b)RGB 图像转化得到灰度图像 (c)自适应阈值处理后的突出标记边缘的二值图像 (d)多边形逼近方法检测到的候选轮廓图像 (e)轮廓筛选后得到的初始角点 (g)角点亚像素优化后得到的最终角点

2.2.2 角点投影与细化

亚像素角点细化的核心在于，在角点位置周围一个小正方形区域内，分析图像导数并寻找其极大值点。在图像尺寸较小时，分析区域也随之缩小，从而显著降低计算成本。为同时保证效率与精度，细化过程采用多尺度策略，在原始图像的金字塔结构上逐层进行。具体而言，系统首先为每个角点寻找可进行初次细化的最小分辨率图像，在完成初步细化后，进入上一层更高分辨率图像继续精细化，直到最终在原始图像上完成所有角点的精确定位。

图像金字塔由多层不同分辨率的图像组成。最顶层是原始图像，后续各层是通过按比例缩小图像尺寸逐级构建而成。在进行角点细化时，首先需要为每个标记选择一个合适的图像层作为起始细化层。选择依据是该图像中标记的投影面积接近理想的参考面积。如果标记在某一层图像中的面积太小，不满足角点间最小距离的要求，那么该标记将不被考虑。

如图 4(e)(f)所示，实验结果验证了该方法的有效性。图 4(a)为原始输入图像；图 4(e)展示了根据初步姿态估计投影得到的角点位置；图 4(f)则用红色标出了经

细化后的角点。可以清楚地看到，初始角点位置（蓝色）与细化后位置（红色）相比存在偏差，细化处理提高了角点定位的准确度，并进而改善了姿态估计结果。

在进行角点细化时，还需考虑遮挡情况。在细化过程中会对每个角点的位置进行局部图像分析。首先，判断其周围区域的图像对比度是否过低。由于标记图案为黑白结构，理想情况下角点应处于高对比度区域。如果该区域最亮与最暗像素值的差异低于设定阈值，则认为角点可能被遮挡。此外，若某个角点在细化过程中发生了严重位置漂移，也将被视为不可靠并予以排除。

2.2.3 特征点检测与匹配

章节 2.2.1 中提到的分形 ArUco 角点检测是基于传统 ArUco 的检测算法，而特征点检测与匹配是分形 ArUco 码检测原理的关键，也是它能相比传统 ArUco 鲁棒性有显著优势的关键。在传统 ArUco 码的检测过程中，边框之内的存储二进制 ID 的编码区域仅仅是利用编码信息与字典中的编码进行比对，内部大量的特征点的位置信息都被浪费。而分形 ArUco 利用了这些内部特征点信息用作位姿估计。

特征是一个数字图像中让人有兴趣的部分，它是许多计算机图像分析算法的起点，包括边缘，角，区域，脊等类别。图像中的兴趣点是指具有明确位置且可被鲁棒检测的像素点。这类点通常包含丰富的局部信息，并能在不同图像间实现理想的可重复性检测[6]。兴趣点检测在图像匹配[19]、物体识别[20]、目标跟踪[21]等领域具有重要应用。

兴趣点检测已经有很多成熟的算法，比如 Moravec 角点检测算法[7]，Harris-Stephens 角点检测算法[7,8]，SUSAN 角点检测器[8]。FAST（加速分段测试特征检测）算法最初由 Rosten 与 Drummond 提出[9]，用于识别图像中的兴趣点（interest points），以检测速度快而被移动机器人同步定位与建图（SLAM）这种实时帧率应用广泛使用[10]，如图 5 所示。

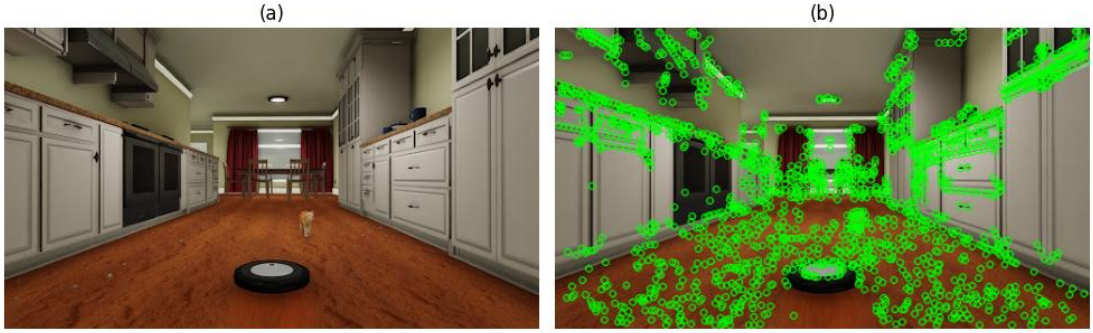

图 5 扫地机器人 SLAM 应用中利用 FAST 方法进行兴趣点检测，(a)为模拟家庭内部建模画面， (b)为在原属图像基础上检测并用绿色圆圈绘制出的兴趣点

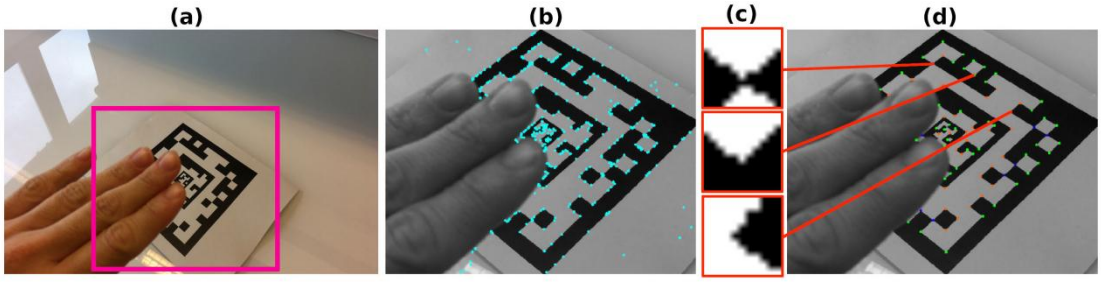

图 6 (a)原始图像展示有兴趣的区域 (b)用 FAST 方法对分形 ArUco 进行粗略特征点检测并用蓝点表示兴趣点 (c)边角分类的示例 (d)过滤并且分类之后的角点，不同的颜色（红色，蓝色，绿色）代表不同的类别[4]

在分形 ArUco 的检测过程中，FAST 同样被用于编码区域特征角点的粗略检测。当然，因为是速度极快的粗略检测，除了算法需要的有效角点之外，还有得到大量干扰的兴趣点，如图 6(a)(b)所示。

对 FAST 算法初步检测得到的特征点集，首先进行两级筛选：第一级基于响应值过滤，计算所有特征点响应值的 20%分位数阈值，仅保留响应值高于该阈值的特征点，确保所选点具有足够的显著性；第二级基于局部对比度过滤，对每个候选特征点提取 10×10 像素邻域，计算其灰度标准差，若标准差低于预设阈值则剔除该点，保证特征点位于纹理丰富的区域。

通过过滤的特征点随后进入分类阶段：对每个特征点的 10×10 邻域进行基于平均灰度的二值化处理，然后分析二值图像的连通域数量和非零像素占比——若连通域数为 2 且非零像素占比超过 50%则归为类别 1（绿色），连通域数为 2 但非零像素占比较少则归为类别 2（红色），连通域数大于 2 则归为类别 3（蓝

色），如图 6(c)(d)所示。该分类结果通过颜色编码可视化，并为后续的特征点匹配提供类型约束，从而在保持实时性的同时有效提升匹配准确率。

在特征点匹配阶段，系统首先基于特征点的分类结果建立匹配条件，仅允许相同类别的特征点之间进行匹配。单应矩阵[22]作为处理视角变化的核心数学工具，能够建立标记模板平面与图像像素平面之间的透视映射关系。当相机视角发生变化导致标记出现旋转、倾斜或部分遮挡时，算法首先利用检测到的外部四个角点计算单应矩阵，将模板坐标系中的二维点精确投影到当前图像空间，从而预测内部特征点应当出现的位置。基于这一预测，算法通过基于这些预测点建立 KDtree[17]索引然后半径搜索寻找预测点周围符合约束的特征点。该算法采用改进的 RANSAC 框架实现鲁棒位姿估计[11]：首先随机选取 4 组具有相同类别标识的特征匹配对计算初始单应性矩阵，评估该变换下所有特征点的投影误差，统计满足误差阈值的 inlier 数量。经过多次迭代后，选择 inlier 数量最多的变换作为最优解，最后使用所有 inlier 重新计算精确的变换矩阵。该方法通过单应性预测和分类约束的双重保障，显著减少了误匹配的可能性，同时利用 RANSAC 机制确保了算法对噪声和异常值的鲁棒性，即使在复杂视角变化或部分遮挡情况下，仍能为后续的位姿估计提供可靠的 2D-3D 对应关系。这种融合了类别约束和统计优化的匹配策略，使算法能够同时保证计算效率和姿态估计精度[16]。

2.3 分形 ArUco 库技术实现分析

分形 ArUco 功能目前有两个官方实现版本：一个是 ROS[23]集成的 ArUco 库，支持分形标记检测，但依赖 ROS 生态；另一个是轻量级头文件 nanofractal.h[12]，独立于 ROS，适合嵌入式和快速应用场景。通过调研得知，两个版本的理论原理一致，核心算法相同，具体实现以及主要特征高度相似。相比之下，nanofractal.h 更轻量，代码结构清晰，无 ROS 依赖，适合作为 OpenCV 集成的模板，nanofractal.h 的实现架构如图 7 所示。因此在后续代码分析过程中，本研究也将 nanofractal.h 作为主要参考模板。

分形 ArUco 的检测方法的实现中，阈值分割，轮廓定位，矩阵计算，FAST 特征点检测，图像绘制等功能都使用了 OpenCV 自带的方法。值得注意的是，FLANN[13]的 KD 树索引在这里用于将检测到的特征点与投影到图像上的标记

点进行匹配。然而这里的 FLANN 没有使用 OpenCV 自带的模块，而是使用了一个名为picoflann.h的头文件中定义的KD树索引类[5]。picoflann.h同样也是Rafael Muñoz Salinas 教授在多次改进后实现的专门针对类似任务的轻量头文件。区别于 OpenCV 集成的 FLANN 模块的 KD 树索引，picoflann.h 通过轻量化的实现和适配器机制，专注于特定任务的优化，尤其是在低维数据的特征点匹配中表现出色。它提供了更高的灵活性和可定制性，允许用户根据具体需求调整数据访问方式和索引行为，同时避免了 OpenCV 模块中可能存在的额外性能开销。

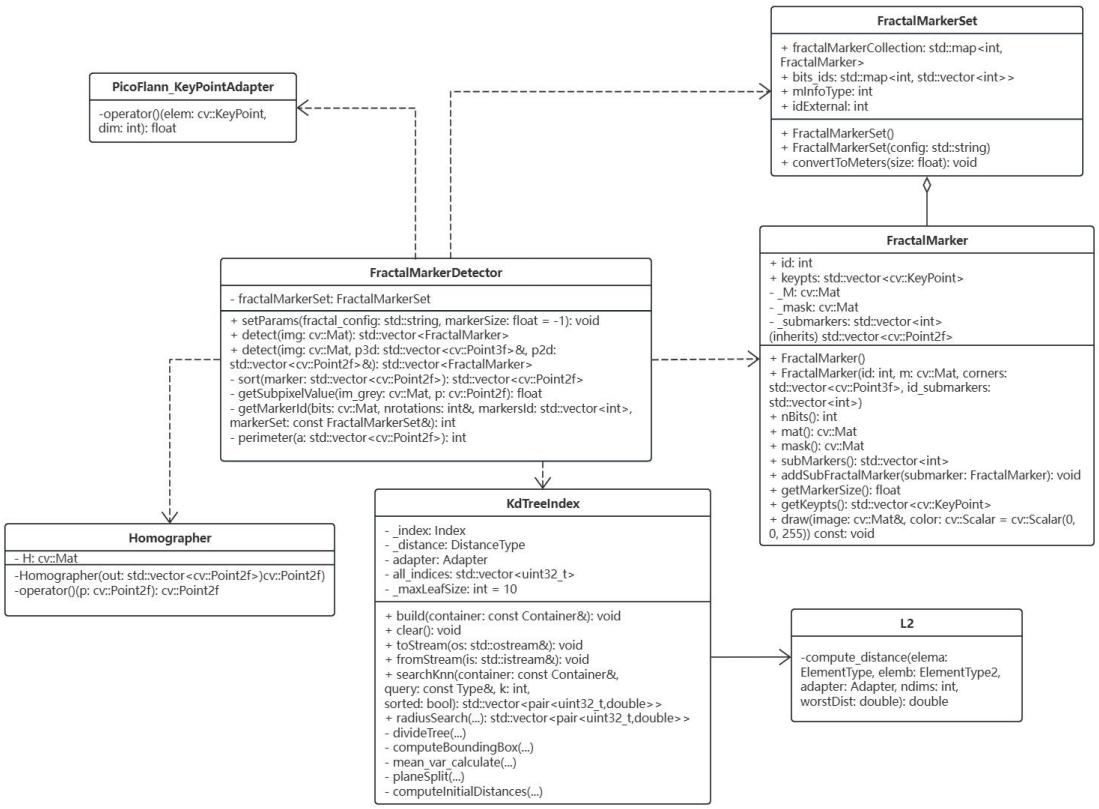

图 7 原 nanofractal.h 头文件分形 ArUco 实现类的依赖 UML 图

3. OpenCV 集成方案设计与实现

3.1 模块架构设计

分形 ArUco 的功能实现主要依赖于三个类：FractalMarker，FractalMarkerSet，FractalMarkerDetector。参考 OpenCV 库中 objdetect 模块的接口习惯，传统的 ArUco 模块设计主要依赖两个接口，aruco_detector.hpp 和 aruco_dictionary.hpp。字典是相同大小的各种独一无二的 ArUco 码的集合，aruco_dictionary 是用户用来调取预存的字典的接口。但由于分形 ArUco 码设计的特殊性，它与普通 ArUco 标记在存储结构上存在显著差异。普通 ArUco 标记通常仅需存储标记 ID 和基础位姿信息，而分形 ArUco 则需要记录多层级标记系统的完整拓扑结构。其配置数据包含：层级数量、子标记数量、各标记的 3D 空间坐标、标记间的连接关系矩阵（二进制位映射），以及几何约束参数。此外，预定义的分形 ArUco 每个层级数（2~5 层）只有一个固定的码，并不像传统 ArUco 有一整个字典。因此，本研究认为分形 ArUco 的预存储信息的调取接口不适合插入 arcuo_dictionary 接口中。最终设计方案为 FractalMarker 和 FractalMarkerSet 被定义在 fractal_marker.hpp 文件中，FractalDetector 类被定义在 fractal_detector.hpp 文件中。这样的设计和传统 ArUco 模块接口为平行关系，比较适合用户理解和调用，不易造成混淆，而且 objdetect 模块也维持了简洁。

3.2 FLANN 模块替换方案

如图所示，在原有设计中，FractalMarkerDetector 模块集成了过多的外部依赖。尤其在图像处理与特征匹配方面，多个功能性类如 KdTreeIndex、Homographer、L2 等直接暴露，造成了接口冗长、维护困难，如图所示。其中 KdTreeIndex、L2 都是 picoflann 命名空间内的类和结构，而 Homographer、Picoflann_KeyPointAdapter 则是为了处理该任务专门设计的类和结构。

根据 picoflann 作者 Rafael Muñoz Salinas 教授的实验数据表明，picoflann 在 10000 个 2D 点的数据集上同 OpenCV 的 FLANN 模块相比，索引建立速度提升 1.73 倍，KNN 搜索速度提升 1.8 倍，半径搜索速度提升 3.21 倍[5]。在这种情况下，这两种实现方案要在关键点匹配的表现和代码的简洁，用户友好性之间进行权衡。因此本实验需要在去除分形 ArUco 模块对 picoflann 的依赖，用 OpenCV

自带 FLANN 模块进行替换之后，对两种关键点匹配的实现进行性能对比和代码结构和复杂性进行对比。

在去除 picoflann 的依赖后，KdTreeIndex、L2 以及专门为处理该任务设计的 Picoflann_KeyPointAdapter 类均被去除，用于单应矩阵计算操作的 Homographer 类也可以通过方法内置而去除。此外，原先实现针对关键点过滤和分类实现的 kfilter 和 assignClass 方法经研究分析，只有在 FractalMarkerDetector 的检测算法实现中收到调用。因此计划将这两个方法移植为 FractalMarkerDetector 类的内置方法。最终的项目结构如图 8 所示。

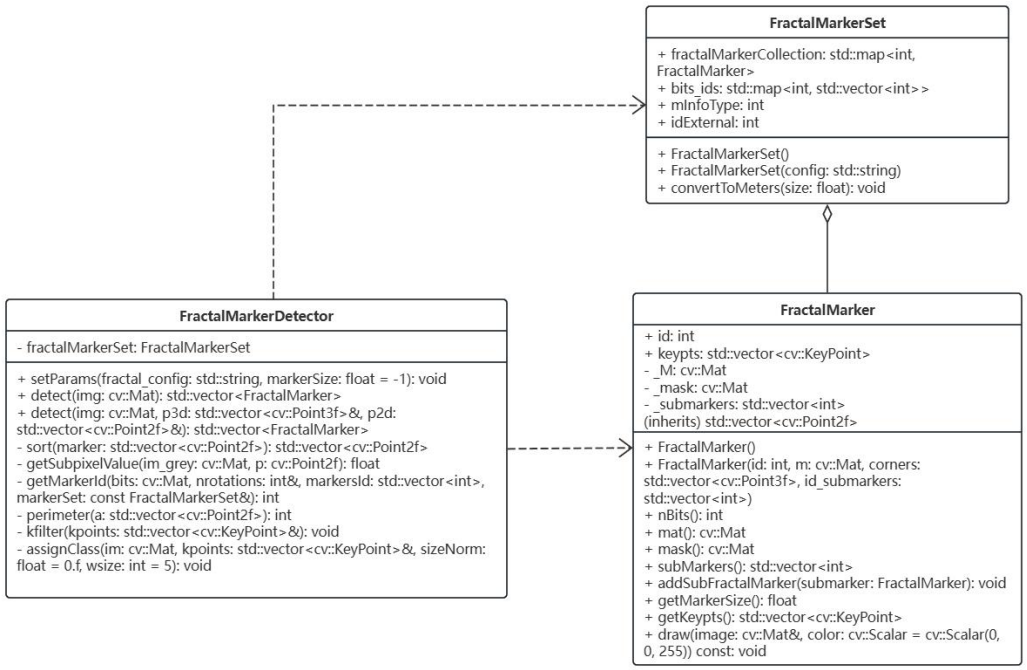

图 8 经过优化和模块替换后的分形 ArUco 项目结构 UML 图

3.3 重复特征点异常修复

在实验过程中，研究发现最终得到的匹配到的特征点集有重复现象。更详细的来讲，特征点匹配算法的理想解集应该是一个 2D 点集和一个 3D 点集，两个点集一一对应且每个点独一无二。其中 2D 点集代表在筛选且分类之后的特征点集中找到的能够与平面标准分形 ArUco 码上的角点匹配上的点的集合，3D 点集则代表特征点集在模板平面上的对应点的集合。首先,由于在原始图像进行关键点绘制时，重复的点会覆盖，因此人眼无法直观的从图像上观测到重复的点。此外，一张分形 ArUco 码的特征点数量极多，对于关键点是否能正确匹配标准分

形 ArUco 码上的角点，也有一定的难度进行判断和统计。

原始 nanofractal.h 的算法经过测试，在某些数据集上有出现特征 2D 点重复的情况。经过分析，重复的 2D 点对应的 3D 点是不同的。如果 3D 点集不在同一平面，经过通过初步检测计算出的单应矩阵投影计算出的 2D 点可能出现重合的情况，这是遮挡造成的结果。然而在本实验中，算法依赖的标准分形 ArUco 码上的所有角点都在同一平面，因此视觉变换后理论上不会出现重复点的情况。

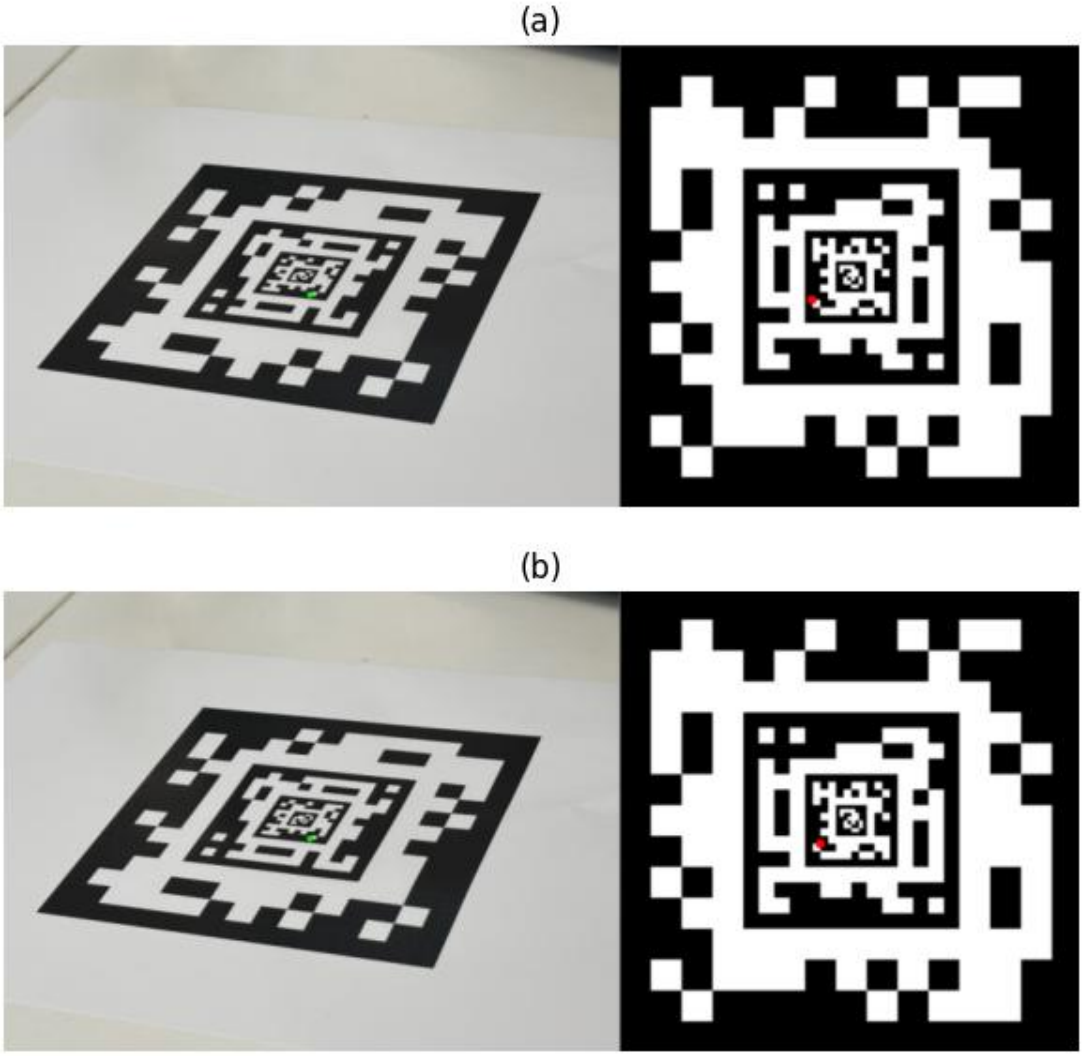

图 9 nanofractal.h 的计算结果，模板平面上(右图)不同的点匹配到相同的特征点(左图)，其中(a)是正确的匹配，(b)是错误的匹配

在这种情况下，出现重复点的原因只有可能是在 FAST 特征点检测时没有正确检测出真正匹配的特征点，或者在后续分类过滤的过程中错误将需要的特征点

抹除，导致在特征点匹配的阶段，基于根据单应矩阵视角转换计算出的理论 2D 点集利用 FLANN 进行半径搜索时找不到正确的特征点，于是只能找一个相对最近的类别信息相同的角点进行替代，如图 9 所示。与此同时，被错误匹配的角点在可能之前已经与其他理论 2D 点进行了匹配，这也就造成了出现重复特征点的现象。

在使用 OpenCV 自带 FLANN 模块替换后的版本中，这种现象同样存在。这些点集一一对应的关系是要在未来算法用于位姿估计，因此这些点匹配的精确性是尤为重要的。用户期待使用的检测算法的输出应该是保证正确的情况下尽可能多的匹配检测到的关键点和模板样点。由于特征点检测所用的 FAST 方法是 OpenCV 自带方法，没有办法进行简单的优化，本实验把工作重点放在匹配过程中对重复点的过滤筛选。通过记录之前加入 2D 点集的关键点，在添加新的特征点时检查是否重复，如果重复则比较重复的点与匹配到的 3D 点集中的欧几里得距离，并且保留距离最近的点作为正确的匹配点。算法流程图如图 10 所示。

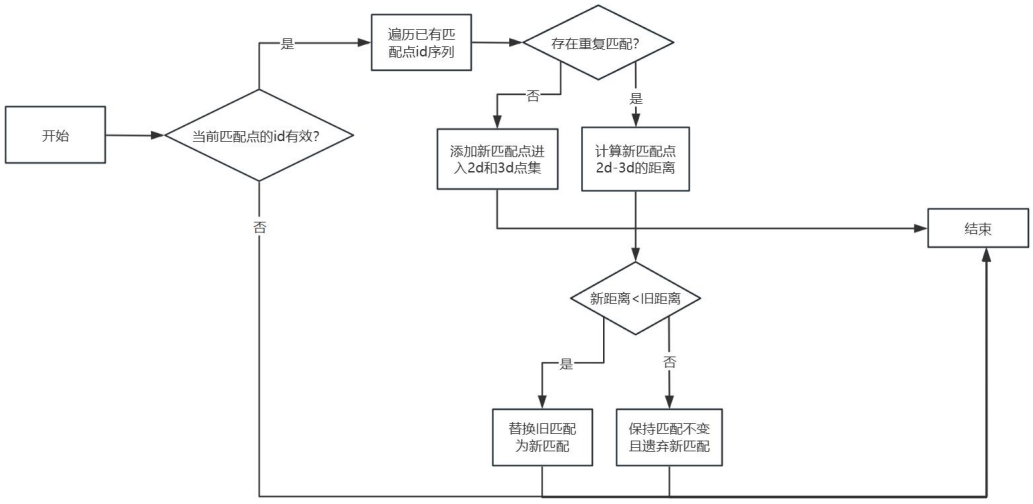

图 10 消除重复特征点匹配的算法流程图

OpenCV 集成的 FLANN 在此情景下通过使用半径搜索来匹配 3D 点集的投影在一定半径距离 R 内和图像中的最近的 2D 关键点。然而，如果匹配到的最近的相同几何约束的点到查询点距离超过了搜索半径距离 R，返回的匹配点 dist 距离为默认值 0，按照图 10 的算法计算会导致结果全部保留为默认值为 0 的异常点（outliers）。因此未达到避免错误匹配的目的，这里应该添加判断逻辑来避免

算法收敛到 dist 为 0 的异常点。此外，搜索半径距离 R 的调整也不应该被忽略。

通过这样的过滤和替换处理，实验期望的结果理论上能够有效地提高关键点匹配的性能。

4. 实验分析与验证

这一章节通过实验对比和分析原版 nanofractal.h 的分形 ArUco 码检测算法和在去除 picoflann 依赖，特征点匹配模块替换和优化之后的新版算法，来证明新版算法实现的有效性。实验从计算效率，特征点匹配数量，特征点匹配准确率三个标准进行衡量。本实验之后统一称前者算法为 nano 版本，后者为 OpenCV 版本，以便描述和理解。

4.1 实验环境与数据集

本实验实现设备为 Intel(R) Xeon(R) Platinum 8352V 2.1GHz 处理器，采用双路 CPU 配置，共 128 个逻辑核心，支持 NUMA 架构。所有实验均在 Ubuntu 22.04.1 LTS 操作系统环境下完成，系统内核版本为 5.4.0-155-generic。实验依赖的 OpenCV 库为 5.x 开发版本。

对于数据集的拍摄，摄像机选用 iPhone15 后置摄像头。nano 版本和 OpenCV 版本的算法实现差别主要在特征点匹配阶段，由于比对特征点是否正确匹配模板样点没有办法通过算法处理进行有效的判断。尤其是在特征点密集的情况下，无法通过简单的算法来正确判断所有的边缘案例。因此本实验的数据匹配更依赖人为判断，这导致无法处理规模太大的数据集。

在数据集的设计中，本实验采用了 FRACTAL_4L_6 分形码作为拍摄对象（除此以外还有 FRACTAL_2L_6，FRACTAL_3L_6 和 FRACTAL_5L_6,其中 L 代表分形码嵌套的层数，实验采用的分形码由 4 层 ArUco 码嵌套组成）。数据拍摄方法为将打印好的 FRACTAL_4L_6 分形码平放在水平面，摄像机以一定的倾斜角度的方向来拍摄。一组数据是无遮挡状态下的分形码，另一组数据是有遮挡状态下的分形码，专门遮挡住外围边框的角点以测试算法专门处理应对遮挡情况的检测机制。每一组数据图像的原始分辨率均为 4032*3024，通过 OpenCV 自带方法分别将像素压缩到 2016*1512，1344*1008，1008*756，672*504，如图 11(a)(b) 所示。

在本实验中，因为算法大多数依赖像素级的阈值长度，图像分辨率是影响算法输出的极其重要的因素。也正因为 nano 版本和 OpenCV 版本之间的检测算法完全一致，实验不需要其他的一些复杂的拍摄条件作为对照，比如模糊的程度，复杂的光照环境，极限的倾斜角等。本实验的数据集只是针对特定任务定制，如

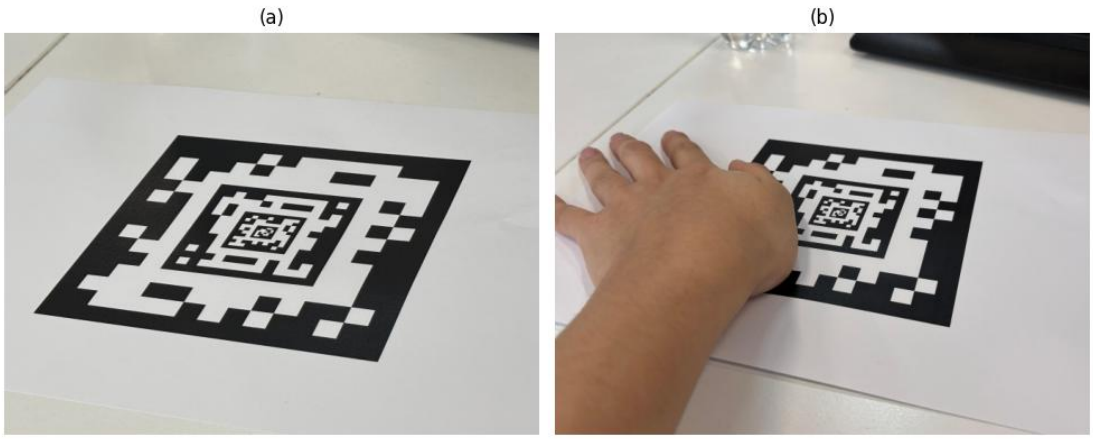

图 11 (a)为无遮挡的分形码图像，(b)为有遮挡的分形码图像

果未来有检测方法优化的测评需求，本实验的数据集测评无法完全覆盖各项指标。

4.2 性能对比测试

4.2.1 计算效率测试

首先是计算效率测试。本实验不只是对于 nano 版本和 OpenCV 版本之间的特征点匹配算法进行计算效率比较，也要对整个检测算法其他阶段进行测试。因为假设某个版本的实现在特征点匹配过程中花费了比另一个版本多数倍的时间，但是如果在整个检测算法中特征点匹配的耗时占比很低，整体算法的耗时实际上区别并不明显，那么系统完全可以采纳匹配准确率更高和匹配数量更多的算法。不同版本在不同数据集上的关键点匹配在算法总时间的时间占比如图 12 所示。

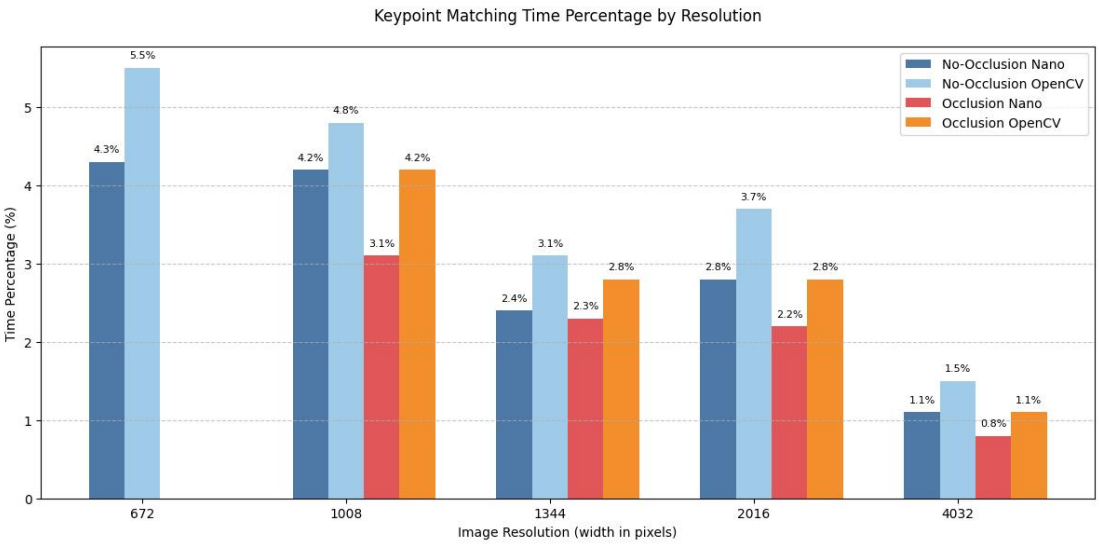

图 12 图中纵坐标为关键点匹配时间占比，横坐标为图像分辨率的宽度；No-Occlusion 代表无遮挡的数据集，Occlusion 代表有遮挡的数据集

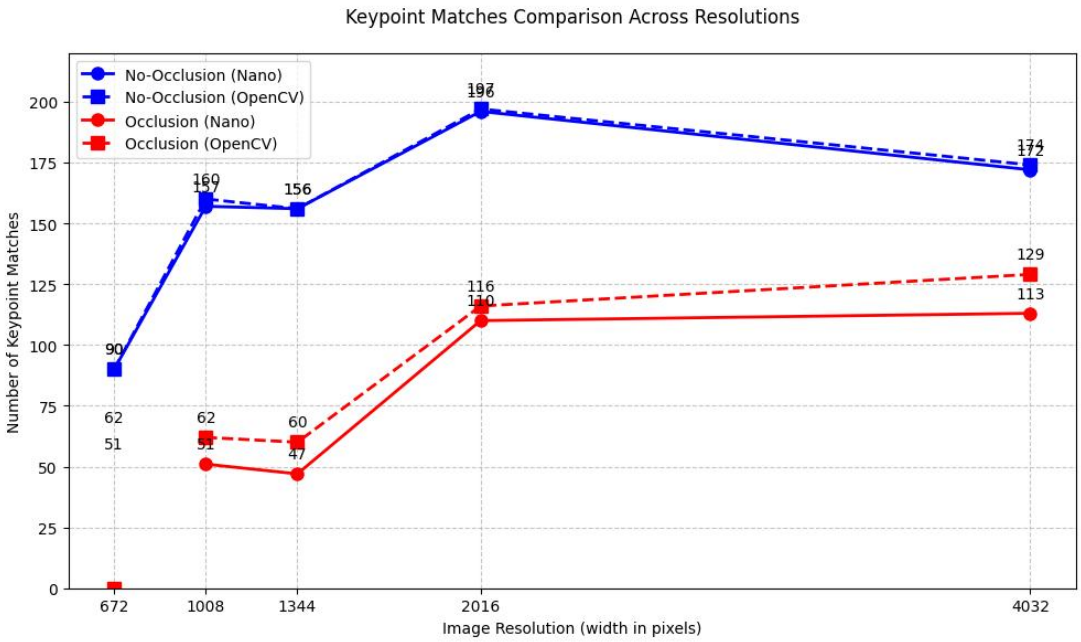

图 13 图中纵坐标为关键点匹配数量，横坐标为图像分辨率的宽度；

No-Occlusion 代表无遮挡的数据集，Occlusion 代表有遮挡的数据集

分析图像可知，像素高低和关键点匹配时间占比呈负相关，在有遮挡的数据集上时间占比普遍低于同像素的无遮挡数据集，这是因为检测算法和过滤算法在高分辨率和有遮挡情况下耗时增加的更多。值得注意的是，672 像素且有遮挡的情况下由于两种算法都无法检测出完整的外部边框，没有触发之后的关键点检测算法，因此图像中并没有关于它们的时间占比数据。

从数值上进行分析，可以发现关键点检测的时间占比是普遍较少的，OpenCV 版本的关键点匹配算法要比 nano 版本的多耗时 0.1 ~ 0.3ms，这导致算法的总耗时在有遮挡数据集上提高了 0.28% ~ 1.2%，在无遮挡数据集上提高了 0.8%~1.7%。换句话说，这是可以忽略不计的效率损耗。

4.2.2 特征匹配数量测试

其次是关键点匹配的数量的对比。不同版本在不同数据集上的关键点匹配数量如图 13 所示。从总体上分析，分辨率越高，关键点越多，能够匹配到模板分形 ArUco 码的关键点就越多，因此图像中的四条曲线整体都呈上升趋势。同时，分辨率越高，在相同数据集上的表现 OpenCV 版本相对于 nano 版本的优势越明显。而且由于遮挡数据集中有大量特征点被遮挡，能够检测和匹配的特征点数量相比没有遮挡的数据集要大幅度减少，这也是符合实验预期的结果。同样地，分辨率宽度 672 像素的遮挡数据集由于没法定位到外部边框无法进行单应矩阵的

计算因此无法通过模板分形 ArUco 的特征点坐标来估计现实的特征点坐标进而无法进行特征点匹配，因此图 13 中的数据为 0。

4.2.3 真阳率测试

然而想要验证 OpenCV 版本的特征点匹配算法的有效性，单独使用匹配数量作为衡量标准是不够的，因为很有可能出现假阳样本。因此接下来本实验的重点放在如何统计大量匹配点的真阳率。通过计算预测点位和实际的特征点位之间的欧几里得距离，如果大于某一设定的阈值，系统可以粗略判定出现了错误匹配。然而这种方式有很大的局限性，因为当特征点十分密集时很有可能出现有误匹配但是计算出的欧几里得距离并不大的情况。因此实验最终选择用人眼观测评判并统计出现错误匹配的特征点对数量，最终对实验结果进行分析统计。Nano 版本和 OpenCV 的实验结果分别如表 1 和表 2 所示。

表 1　nano 版本错误匹配数量统计表

分辨率（像素）	无遮挡数据集	有遮挡数据集
672	0	-
1008	0	0
1344	0	0
2016	1	0
4032	0	0

表 2　OpenCV 版本错误匹配数量统计表

分辨率（像素）	无遮挡数据集	有遮挡数据集
672	0	-
1008	0	2
1344	0	0
2016	0	0
4032	0	0

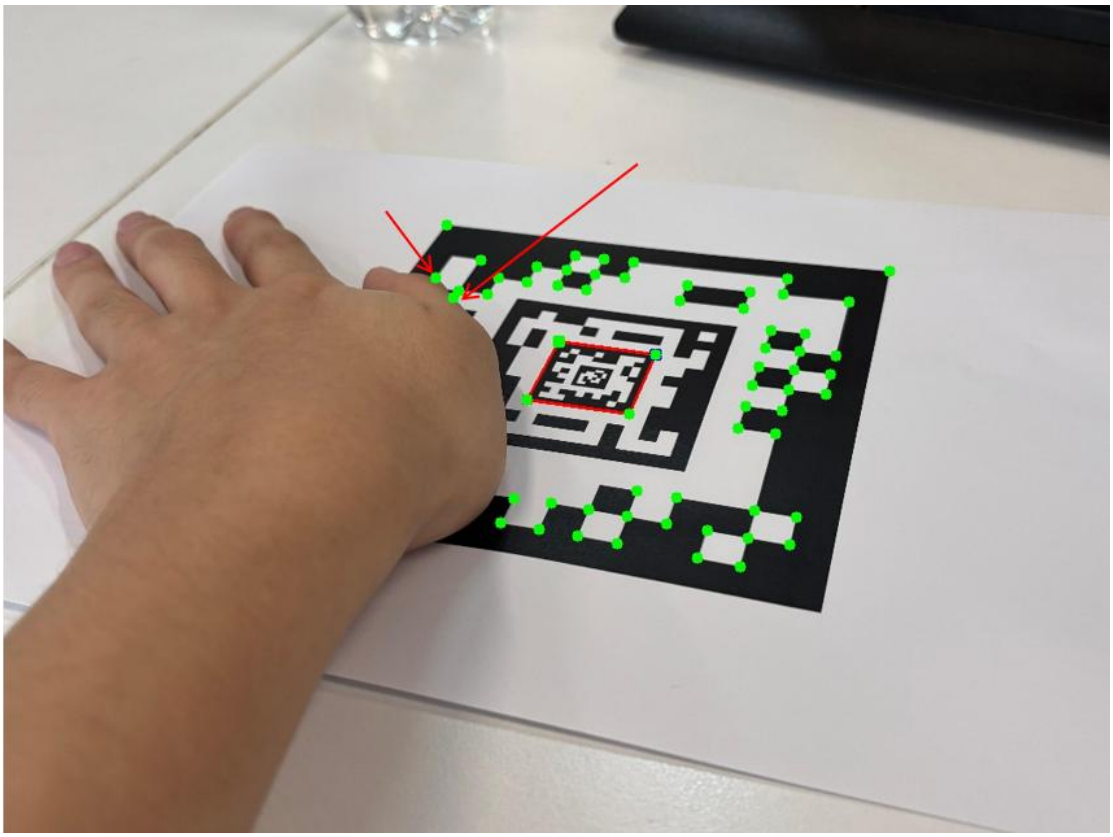

图 14 图中绿色点代表被匹配的关键点，红色箭头标注误匹配的关键点

通过分析数据，nano 版本出现的错误匹配仅有一例，即图 9(b)所展示的情况。通过第三章中提到的算法修复，OpenCV 版本在无遮挡数据集上可以达到 100%的真阳率，较 nano 版本增加的匹配点对都是正确匹配。在有遮挡数据集上，如图 14 所示，出现的误匹配都是由于类似的遮挡物体边缘的特征点被误检。造成这种情况出现有两个原因，一是因为特征点过滤和分类算法的不完善，导致这样的错误的特征点在过滤算法之后仍然被保留甚至给以分类，二是因为后续特征点匹配算法处理与预期点位距离较近的点的情况不够完善。

总体来说，OpenCV 版本相比于 nano 版本性能上计算效率几乎无差别（总耗时增加 0.28%～1.7%），在关键点匹配任务中能正确匹配的关键点数量有大幅提升（0%～28%），但是目前由于过滤算法不够完善，结果包含少数假阳案例。对于位姿估计任务，虽然后续通过 RANSAC 算法的鲁棒机制可以抽样计算出最佳的模型，但是 outliers 的数量越少算法就可以以越少的抽样次数确定最佳的模型进而计算单应矩阵，因此解决错误匹配的问题在该检测任务中十分重要，需要从关键点过滤分类算法和调试距离阈值参数两个层面分别优化。

5. 结论与展望

5.1 研究成果总结

本研究通过深入挖掘分形 ArUco 码的实现原理和 OpenCV 库的使用，成功将分形 ArUco 码的功能通过简洁有效的架构设计，用户友好的接口很好地集成进了本地的 OpenCV 库中。具体实现的代码行数从原先的 1500 行缩减到 800 行左右，大大提高了面向用户和开发者的可读性。通过实验不断调整优化，OpenCV 版本的实现相比于原版 nano 版本的实现已经在计算效率几乎无差别的情况下，能够做到在关键点匹配任务中匹配效果显著上升，但是仍然有少数错误的关键点匹配情况，还有继续优化的空间。

5.2 未来工作方向

未来的工作重心为在更多不同拍摄条件（包括不同的距离，角度，光线条件，遮挡物）的数据集上进行测试，排除潜在程序错误的影响。同时，目前存在的少量关键点错误匹配问题需要通过调试距离阈值参数来解决。实现能在多场景稳定输出正确的特征点匹配测试后，会向 OpenCV 社区提交合并申请。此外，未来研究方向将持续对检测方法尝试优化，比如尝试使用深度学习方法代替传统的视觉检测方法，进一步提高计算效率和检测准确率，并且持续同步 OpenCV 库对应模块的更新。

参考文献

[1] Roos-Hoefgeest, S., Garcia, I. A., & Gonzalez, R. C. (2021, October). Mobile robot localization in industrial environments using a ring of cameras and ArUco markers. In IECON 2021–47th Annual Conference of the IEEE Industrial Electronics Society (pp. 1-6). IEEE.

[2] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., & Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognition, 47(6), 2280-2292.

[3] Mráz, E., Rodina, J., & Babinec, A. (2020, October). Using fiducial markers to improve localization of a drone. In 2020 23rd International Symposium on Measurement and Control in Robotics (ISMCR) (pp. 1-5). IEEE.

[4] Romero-Ramire, F. J., Muñoz-Salinas, R., & Medina-Carnicer, R. (2019). Fractal markers: A new approach for long-range marker pose estimation under occlusion. IEEE Access, 7, 169908-169919.

[5] https://github.com/rmsalinas/picoflann

[6] Shi, J. (1994, June). Good features to track. In 1994 Proceedings of IEEE conference on computer vision and pattern recognition (pp. 593-600). IEEE.

[7] Dey, N., Nandi, P., Barman, N., Das, D., & Chakraborty, S. (2012). A comparative study between Moravec and Harris corner detection of noisy images using adaptive wavelet thresholding technique. arXiv preprint arXiv:1209.1558.

[8] Chen, J., Zou, L. H., Zhang, J., & Dou, L. H. (2009). The Comparison and Application of Corner Detection Algorithms. Journal of multimedia, 4(6).

[9] Viswanathan, D. G. (2009, May). Features from accelerated segment test (fast). In Proceedings of the 10th workshop on image analysis for multimedia interactive services, London, UK (pp. 6-8).

[10] Durrant-Whyte, H., & Bailey, T. (2006). Simultaneous localization and mapping: part I. IEEE robotics & automation magazine, 13(2), 99-110.

[11] Derpanis, K. G. (2010). Overview of the RANSAC Algorithm. Image Rochester NY, 4(1), 2-3.

[12] https://sourceforge.net/projects/aruco/files/ArucoNano-v5/nanofractal.h/

[13] Muja, M., & Lowe, D. (2009). FLANN-fast library for approximate nearest neighbors user manual. Computer Science Department, University of British Columbia, Vancouver, BC, Canada, 5(6).

[14] Fiala, M. (2009). Designing highly reliable fiducial markers. IEEE Transactions on Pattern analysis and machine intelligence, 32(7), 1317-1324.

[15] Stone, D., Jarrett, C., Woodroffe, M., & Minocha, S. (2005). User interface design and evaluation. Elsevier.

[16] Lepetit, V., Moreno-Noguer, F., & Fua, P. (2009). EP n P: An accurate O (n) solution to the P n P problem. International journal of computer vision, 81, 155-166.

[17] Ram, P., & Sinha, K. (2019, July). Revisiting kd-tree for nearest neighbor search. In Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining (pp. 1378-1388).

[18] Issac, A., Sarathi, M. P., & Dutta, M. K. (2015). An adaptive threshold based image processing technique for improved glaucoma detection and classification. Computer methods and programs in biomedicine, 122(2), 229-244.

[19] Lindeberg, T. (2015). Image matching using generalized scale-space interest points. Journal of mathematical Imaging and Vision, 52, 3-36.

[20] Zou, X. (2019, August). A review of object detection techniques. In 2019 International conference on smart grid and electrical automation (ICSGEA) (pp. 251-254). IEEE.

[21] Serby, D., Meier, E. K., & Van Gool, L. (2004, August). Probabilistic object tracking using multiple features. In Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004. (Vol. 2, pp. 184-187). IEEE.

[22] Dubrofsky, E. (2009). Homography estimation. Diplomová práce. Vancouver: Univerzita Britské Kolumbie, 5.

[23] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., ... & Ng, A. Y. (2009, May). ROS: an open-source Robot Operating System. In ICRA workshop on open source software (Vol. 3, No. 3.2, p. 5).

致谢

在论文交付之际，我谨向在本项目过程中给予我帮助的人致以诚挚的感谢。

首先，衷心感谢我的导师于仕琪老师。从选题到实现方向，于老师以其深厚的学术积累和项目经验不遗余力地指导我的研究进程。无论是实验设计还是论文撰写，他总能一针见血地指出问题并给出建议，这些宝贵的指导让我受益匪浅。此外，得益于于老师的引荐，我有幸与 ArUco 及分形 ArUco 的发明人——西班牙科尔多瓦大学的 Rafael Muñoz-Salinas 教授进行交流，他的权威建议对本项目的改进起到了重要作用。在此，我也向 Muñoz-Salinas 教授表达深深的感激。

最后，感谢课题组学长学姐们的支持，他们为我耐心解答了许多工程问题，并帮助我更好地理解项目细节。