高技术通讯 2022 年 第 32 卷 第 11 期:1112-1125

doi:10.3772/j.issn.1002-0470.2022.11.002

ORB-SLAM 系统特征分析研究①

薛 瑞②*** 李 易*** 李文明③* 安述倩* 叶笑春* 唐志敏***

(*处理器芯片全国重点实验室(中国科学院计算技术研究所) 北京 100190)

(**中国科学院大学计算机科学与技术学院 北京 100049)

摘 要 随着自动驾驶汽车、机器人、无人机、虚拟现实和增强现实等应用的飞速发展,其核心技术同步定位和建图(SLAM)成为目前热门研究方向之一。ORB-SLAM 系统作为典型的基于特征点法的 SLAM 系统,具有更好的鲁棒性和更高的计算效率,无论在系统优化层面还是底层硬件架构设计层面一直被广泛关注。然而目前学术界和工业界缺乏面向 ORB-SLAM 系统底层硬件架构设计的系统特征分析研究。本文从跟踪线程、地图构建线程和回环检测线程出发详细介绍 ORB-SLAM 系统,选取了 ORB-SLAM2 系统进行了性能分析实验,得到了 ORB 特征提取和块求解器 2 个热点函数,并分析了 2 个热点函数的执行特征。在 Intel i5-6500 和 ARM Neoverse-N1 处理器平台实验对比评估了 2 个热点函数的 IPC、分支预测失效率、一级数据缓存读失效率、最后一级缓存失效率和最后一级缓存 MPKI 等特征,并总结了对体系结构设计的需求,为面向 ORB-SLAM 系统的底层硬件架构设计提供了指导性建议。

关键词 ORB-SLAM;热点函数;特征分析;硬件架构设计

0 引 言

同步定位和建图(simultaneous localization and mapping, SLAM)作为自动驾驶汽车、机器人、无人机、虚拟现实和增强现实等自主系统的关键技术[1-3],能很好地应用在包括路径规划、导航、避障和3D 重建等高级任务中。在过去的几十年,无论在学术界还是工业界,SLAM 一直是研究的热点[4-7],其主要工作是设计自动导航系统、建立和更新周围 3D 环境同时估计自身传感器的当前位置。

近年来,基于特征点法的 SLAM 系统与其他基于直接法的 SLAM 系统相比,对运动较快的场景具有更好的鲁棒性,且系统处理较为稳定,因此受到了特别的关注[8]。在基于特征点法的 SLAM 系统中,ORB[9](oriented FAST and rotated BRIEF)特征计算方法相比于尺度不变特征变换(scale-invariant feature transform,SIFT)或加速鲁棒特征(speeded up robust features, SURF)算法具有良好的旋转和缩放不变性,计算效率和鲁棒性更高,因此被广泛采用。

由于 ORB-SLAM 系统应用于现实世界的真实场景中,主要输入数据为大量的连续图像帧,因此针对海量的图像帧数据快速处理的需求,提高 ORB-SLAM 系统的处理性能成为当前亟待解决的问题,因此对 ORB-SLAM 系统进行特征分析研究以指导面向 ORB-SLAM 系统的底层硬件架构的设计具有重要的意义。目前已有的底层硬件架构设计有 eSLAM[8]、HcveAcc[10] 和 PISCES[11] 等。然而这些加速结构没有对算法进行深入的分析,缺乏为后续底

① 国家自然科学基金(61732018,61872335,61802367),中国科学院战略性先导科技专项(C 类)项目(XDC05000000),中国科学院国际伙伴计划(171111KYSB20200002)和数学工程与先进计算国家重点实验室开放基金(2019A07)资助项目。

② 女,1993 年生,博士生;研究方向:计算机系统结构;E-mail:xuerui@ict.ac.cn。

③ 通信作者,E-mail:liwenming@ict.ac.cn。

(收稿日期:2021-08-30)

层硬件架构设计提供有效的指导。针对上述问题,本文对 ORB-SLAM 系统进行了较为全面的应用特征分析,以指导面向 ORB-SLAM 系统的底层硬件架构的设计。

本文首先从对 ORB-SLAM 系统介绍出发,分别详细阐述 ORB-SLAM 系统[12]3 个线程的工作原理及其 2 个优化版本 ORB-SLAM2[13] 和 ORB-SLAM3[14],并选取 ORB-SLAM2 作为本文实验的目标系统。然后通过实验 ORB-SLAM2 系统,得到并分析系统的热点函数特征。最后根据对热点函数特征的分析结果,提出设计面向 ORB-SLAM 系统的底层硬件架构的需求。

本文的主要贡献包括:(1)完整地介绍了 ORB-SLAM 系统跟踪、地图构建和回环检测 3 个线程的工作原理;(2)实验分析并重点阐述了 ORB-SLAM2 系统中的热点函数及执行特征;(3)通过对热点函数进行详细的特征分析实验,得到了面向 ORB-SLAM 系统对底层硬件架构的需求。

1　相关工作

当前主流的 SLAM 系统由 4 个阶段组成,分别为前端视觉里程计(visual odometry, VO)、后端优化(optimization)、回环检测(loop closing)和三维地图建立(mapping)[12-17]。前端视觉里程计阶段用来求解传感器在各个时刻的位置信息;后端优化阶段用来全局优化前端视觉里程计长时间的累计误差;回环检测阶段用来检测传感器是否到达之前到过的位置;三维地图建立阶段用来构建传感器在移动过程中周围的 3D 点云地图。

前端视觉里程计有特征点法和直接法 2 种求解方法,现有的基于特征点法的 SLAM 系统有 Mono-SLAM[18]、PTAM[15]、ORB-SLAM[12] 及其优化系统 ORB-SLAM2[13] 和 ORB-SLAM3[14] 等,基于直接法的有 DTAM[19] 和 LSD-SLAM[17] 等;只关注于前端视觉里程计部分的 VIO 设计方案有基于半直接法的 SVO[16]、基于特征点法的 OKVIS[20] 和基于直接法的 DSO[21] 等;只关注于 3D 重建部分的系统有 Kinect-Fusion[22]、InfiniTAM[23]、ElasticFusion[24] 等。表 1 总结了最具有代表性的各个 SLAM 系统。其中 Mono-SLAM 系统是第一个实时的单目视觉 SLAM 系统[18],扩展卡尔曼滤波(extended Kalman filter, EKF)作为后端优化的方法,追踪前端稀疏的特征点,以传感器当前状态和所有路标点为状态量,更新特征点的均值和协方差。该方法的缺点是场景窄、路标数有限、稀疏特征点易丢失。PTAM 系统[15] 提出并实现了跟踪和建图的并行化,首次区分出前后端概念,即把跟踪需要实时响应的图像数据部分作为前端,地图优化放在后端进行。PTAM 系统是第一个使用非线性优化作为后端的方案,并提出了关键帧机制,该方法的缺点是场景小、跟踪容易丢失。ORB-SLAM 系统[12] 围绕 ORB 特征计算,计算效率比 SIFT 和 SURF 算法高,且具有良好的旋转和缩放不变性,并创新地使用了 3 个线程来完成 SLAM 系统,分别为实时跟踪特征点线程、地图构建线程和回环检测线程,该方法的缺点是每幅图像都需要计算 ORB 特征,三线程结构给 CPU 带来了较重负担。ORB-SLAM2 系统[13] 在单目 ORB-SLAM 系统基础上增加了双目和 RGB-D 传感器的输入,并通过使用光束平差法(bundle adjustment, BA),获得了比基于迭代最近点或者光度和深度误差最小化的方法更高的精度;ORB-SLAM2 系统使用近距离和远距离的特征方法,双目效果比直接法更精确且定位时可以重用地图,是一种轻量级的方法,在 CPU 上可以实现实时性;ORB-SLAM2 系统增加的双目和 RGB-D 方法可以解决原来 ORB-SLAM 系统在单目回环检测时严重的漂移问题。ORB-SLAM3 系统[14] 是一种紧密集成的视觉惯性 SLAM 系统。对比于 ORB-SLAM 系统和 ORB-SLAM2 系统,ORB-SLAM3 系统在惯性测量单元初始化阶段引入最大后验概率估计思想,同时采用了多子地图系统。

随着 SLAM 系统需要的处理性能不断提高,近几年学术界和工业界涌现出一些面向 SLAM 系统的加速结构设计。对于使用广泛且本文重点关注的基于 ORB-SLAM 系统的加速结构设计如针对 ORB-SLAM 系统中耗时最高的前端视觉里程计的特征提取部分进行基于 FPGA 硬件结构设计[25]。文献[26]提出了一种基于CGRA结构的加速器设计,从输入

表 1 代表性的各个 SLAM 系统的总结

系统类型	使用方法	系统	传感器数据类型	输入稀疏性	提出年份
SLAM	特征点法	MonoSLAM[18]	单目	稀疏	2007
		PTAM[15]	单目	稀疏	2007
		ORB-SLAM[12]	单目	稀疏	2015
		ORB-SLAM2[13]	单目、双目、RGB-D	稀疏	2017
		ORB-SLAM3[14]	单目、双目、RGB-D、IMU	稀疏	2020
	直接法	DTAM[19]	RGB-D	稠密	2011
		LSD-SLAM[7]	单目、双目、RGB-D	半稠密	2014
VIO	半直接法	SVO[16]	单目	稀疏	2014
	特征点法	OKVIS[20]	双目、IMU	稀疏	2015
	直接法	DSO[21]	单目	半稠密	2018
3D 重建		KinectFusion[22]	RGB-D	稠密	2014
		InfiniTAM[23]	RGB-D	稠密	2015
		ElasticFusion[24]	RGB-D	稠密	2016

图像帧数据以及跟踪任务和列文伯格-马夸尔特算法(Levenberg-Marquardt,LM)的计算特点进行粗粒度可重构的硬件逻辑设计。文献[8]提出了 eSLAM,在 FPGA 平台上设计了基于 ORB 的特征提取和特征匹配 2 部分的高能效加速器。在 eSLAM 基础上,文献[10]提出了 HcveAcc,针对 ORB-SLAM 系统的跟踪任务中的高密集度的特征提取和高精度的描述子生成部分设计了基于 ASIC 的加速器结构,实现 ORB-SLAM 系统运行的高性能和高精度。文献[11]设计了一个基于特征点法 SLAM 系统的前端视觉里程计任务的 EKF 和 ORB 部分以及后端优化任务的 BA 部分的流水线结构 PISCES,利用其计算稀疏性降低访存功耗和计算延时,提高处理性能。表2总结了近几年最具代表性的ORB-SLAM

表 2 近几年最具有代表性的 ORB-SLAM 系统加速器

加速器	优化部分	硬件平台	参考文献	提出年份	数据集	存在的问题
PISCES	跟踪任务中扩展卡尔曼滤波器部分和 ORB 部分后端优化任务中光束法平差部分	FPGA	文献[11]	2020	EuRoC dataset	未量化阐明 SLAM 是计算密集型的原因;只用一组数据集,证明力度不充足
HcveAcc	跟踪任务中特征提取和描述子生成	ASIC	文献[10]	2020	EuRoC dataset TUM dataset	只加速跟踪任务的热点部分,未分析后端优化部分
eSLAM	跟踪任务中特征提取和特征匹配部分	FPGA	文献[8]	2019	TUM dataset	只加速跟踪任务的热点部分,未分析后端优化部分;只用一组数据集,证明力度不充足;只针对基于 ORB 的视觉 SLAM 进行实验,未针对 ORB-SLAM 系统进行系统特征分析和加速结构设计
CGRA	跟踪任务和 LM 算法	CGRA	文献[26]	2018	TUM dataset ICL-NUIM	未量化阐明 SLAM 是计算密集型的原因,只阐述跟踪任务是计算密集型和 LM 算法是耗时最多的部分
FPGA	跟踪任务中特征提取部分	FPGA	文献[25]	2017	无	只加速特征提取部分,未进行 ORB-SLAM 系统前后端系统特征分析和加速器设计

系统加速器,并分析各加速器存在的问题。针对表2中存在的问题,本文进行了面向 ORB-SLAM 系统的应用特征分析研究,为面向 ORB-SLAM 系统的底层硬件架构设计提供了指导性建议。

2　ORB-SLAM 系统介绍

ORB-SLAM 系统[12]是一种基于单目的、围绕 ORB 特征计算的 SLAM 系统,使用跟踪、地图构建和回环检测 3 个线程进行同步定位和建图工作。ORB-SLAM 系统流程图如图 1 所示。

2.1　跟踪线程

跟踪线程主要负责定位传感器并确定何时插入新的关键帧,主要由 ORB 特征提取、初始位姿估计、局部地图跟踪和关键帧选取 4 部分组成。

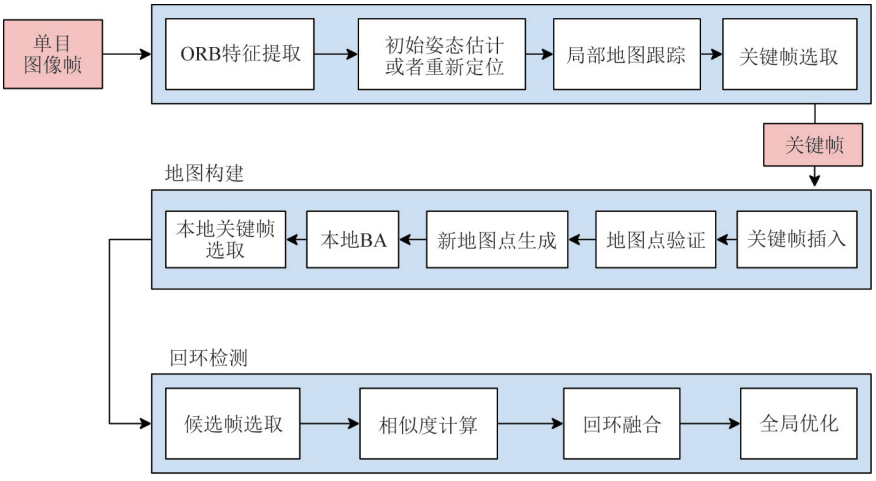

图 1　ORB-SLAM 系统流程图

ORB 特征提取[9]部分是由 FAST(features from accelerated segment test)算法[27]改进得到。ORB 利用 FAST 算法对插入的新图像帧进行 FAST 角点搜索,即搜索整个图像帧中所有与其周围邻域内足够多的像素点的灰度值相差较大的像素点。图 2 所示为图像帧中某一个角点和周围邻域内的像素点,给定比较的阈值以及圆圈大小(本文以圆圈大小 16 为例)。首先比较像素点 1 和 9,如果其和中心像素的灰度值差的绝对值均小于阈值,则该点不是角点,否则将它作为候选角点;再比较像素点 1、5、9 和 13 与中心像素点灰度值差的绝对值,如果其中有3个绝对值大于阈值,则该点继续作为候选点,否则舍弃;最后比较像素点 1 ~16 与中心像素的灰度值差,如果连续大于或小于阈值的像素点超过 8 个,则该点为特征点,否则舍弃。由于此时搜索出的角点过多,因此本文对检测出来的角点群利用非极大值抑制方法进行 FAST 角点的筛选,并在保留的 FAST 角点上计算方向,以此来实现特征点的旋转不变性,最后计算当前图像帧中角点的 BRIEF(binary robust independent elementary features)描述子[28]用于与上一帧中的角点的 BRIEF 描述子进行特征匹配。

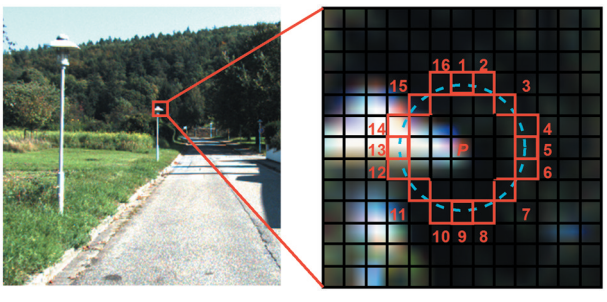

图 2　图像帧中某一角点与其邻域点

初始位姿估计部分首先判断当前帧与上一帧的特征匹配是否成功,若成功则假设传感器为恒速运动模型来预测传感器当前位姿,并搜索与上一帧观察到的地图点进行对应,最后与对应的地图点进行位姿优化。若没有足够的位姿与地图点的匹配对,则使用上一帧中更广泛的位姿附近的地图点的搜索范围。若当前帧与上一帧的特征匹配失败,则需要全局重定位,首先将当前帧转化为词袋向量[29],利用 ORB 特征词典与所有关键帧进行匹配,然后对每

个关键帧进行随机抽样一致性(random sample consensus,RANSAC)迭代,并使用 PnP 算法[30]进行传感器位姿求解。若传感器位姿有足够多可被使用的正确数据,则搜索与所有关键帧观察到的地图点进行对应,通过匹配对进行位姿优化。初始位姿估计部分的流程图如图 3 所示。

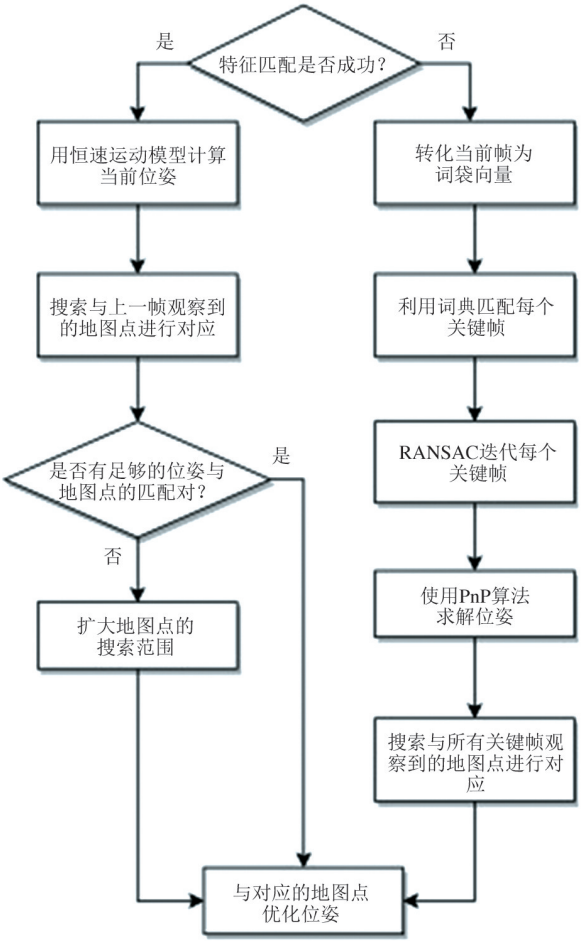

图 3　初始位姿估计部分流程图

局部地图跟踪部分首先更新局部关键帧和局部地图点,将局部地图点与当前帧进行匹配,最后使用最小化重投影误差方法实现对传感器位姿的进一步优化。

关键帧选取部分通过舍弃不满足条件的关键帧,保留满足以下条件的关键帧,保证系统具有鲁棒性。

(1)自上次全局重定位后,必须已经过大于 20 帧;

(2)地图构建部分处于空闲状态,或自上次插入关键帧后已经过大于 20 帧;

(3)当前帧至少跟踪到 50 个点;

(4)当前帧比与当前帧共享最多地图点的帧跟踪到少于 90% 的点。

2.2　地图构建

地图构建线程负责处理新的关键帧并执行 BA,以实现传感器周围地图的最佳重建。此部分主要由关键帧插入、地图点验证、新地图点生成、本地 BA 和本地关键帧选取 5 部分组成。

关键帧插入部分首先更新共视图,然后更新与跟关键帧中有大多数共同的点相连接在一起的生成树,最后计算关键帧的词袋向量。

地图点验证部分主要工作是需要地图点必须在创建后的前 3 个关键帧期间通过限制性测试,以确保它们是可进行特征匹配的并且不会错误地进行三角化计算。

新地图点生成部分负责从共视图中已连接的关键帧进行三角化计算 ORB 特征点来创建新地图点。对于关键帧中每个未匹配的 ORB 特征点,在另一个关键帧中搜索与其他不匹配点进行匹配,丢弃不满足对极约束的匹配。最后对 ORB 特征点匹配对进行三角化计算生成新地图点。

本地 BA 部分主要优化当前关键帧、在共视图中所有连接到当前关键帧的关键帧以及被关键帧观测到的所有地图点。所有其他观测到地图点但未连接到当前关键帧的关键帧均包含在优化中,同时保持固定不变。在优化中间和结束时,将标记为异常值的观测值丢弃。

本地关键帧选取部分主要检测和删除冗余的关键帧以维持重构过程的紧凑。

2.3　回环检测

回环检测线程主要由候选帧选取、相似度计算、回环融合和全局优化 4 部分组成。此线程采用了词袋模型来检测回环,若检测到回环,则进行相似度转换的计算,以告知回路中累积的漂移。然后通过建立当前帧和回环帧之间的匹配关系,融合重复的点,最后对相似性约束进行位姿图优化[31],来修正视觉里程计的局部误差,以实现全局一致性。

2.4　ORB-SLAM2 系统

ORB-SLAM2 系统[13]是在 ORB-SLAM 系统基础上提出来的一个支持单目、双目和 RGB-D 的完整

SLAM 系统方案。它能够实现地图重用、回环检测和重定位功能。同时 ORB-SLAM2 系统在后端优化采用了基于单目和双目的 BA 方法,此方法允许米制比例尺的轨迹精确度评估。此外,ORB-SLAM2 系统包含一个轻量级的定位模式,该模式能够在允许零点漂移的条件下,利用前端视觉里程计来追踪未建图的区域并匹配特征点。ORB-SLAM2 系统流程图如图 4 所示,其中线框部分,即跟踪线程的预处理输入部分和全局 BA 部分为 ORB-SLAM2 系统在 ORB-SLAM 系统的基础上的改进。

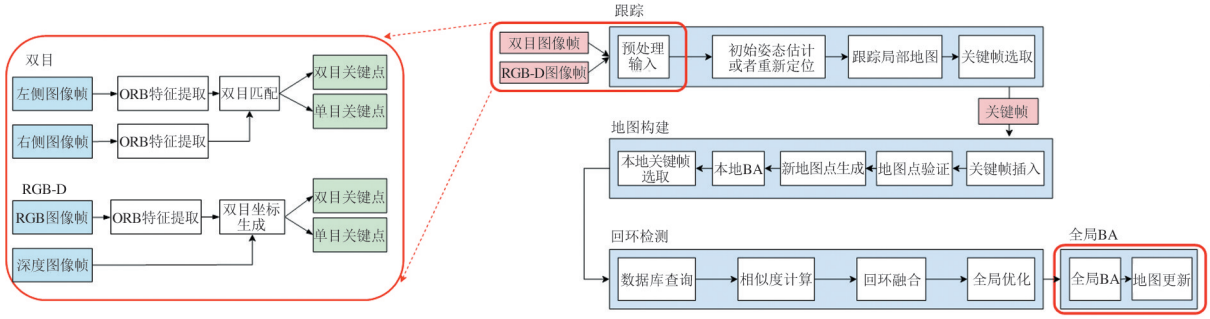

图 4　ORB-SLAM2 系统流程图

本文采用 ORB-SLAM2 系统进行系统特征分析研究的原因如下。

(1)从第 3 节可以看出,ORB-SLAM 系统时间占比最大部分为跟踪线程,而 ORB-SLAM3 系统相比于 ORB-SLAM2 系统的主要优化在于增加了惯性测量单元集成、初始化和缩放修正部分以及地图合并部分,未涉及跟踪线程部分的计算,对系统特征分析影响较小。因此 ORB-SLAM3 系统的改进部分并不是本文的研究目标。

(2)相比于 ORB-SLAM 系统和 ORB-SLAM3 系统,ORB-SLAM2 系统在 SLAM 系统优化及底层硬件架构设计等方面均具有更广泛的应用[8,10-11,25-26]。

因此本文实验部分将采用 ORB-SLAM2 系统进行系统热点函数及系统特征分析。

3　ORB-SLAM2 系统热点函数分析实验

本节将对 ORB-SLAM2 系统进行整体性能分析,并选取其中的热点函数。

实验平台:本文采用基于 Skylake 架构的 64 位 Intel i5-6500 桌面电脑处理器和基于 ARM v8 架构的 ARM Neoverse-N1 处理器作为实验平台进行对比实验。

数据集:本文选取 3 个典型开源数据集 KITTI[32]、EuRoC[33]和 TUM RGB-D[34]作为实验数据集。

KITTI 数据集由德国卡尔斯鲁厄理工学院和丰田美国技术研究院联合创办,是目前国际上最大的自动驾驶场景下的计算机视觉算法评测数据集[32],该数据集包含城镇、乡村等自动驾驶可覆盖到的道路及道路周边信息的图像帧序列。EuRoC 数据集是室内微型飞行器和惯性测量单元收集视觉惯性数据[33],包含苏黎世联邦理工学院的 machine hall 和一个普通房间 vicon room 的室内场景的图像帧序列。TUM RGB-D 数据集是由慕尼黑工业大学计算机视觉实验室公布的[34],该数据集均从实际环境中采集,包含针对纹理丰富的办公室场景以及不同帧数大小、不同运动速度、不同结构纹理等场景的图像帧序列。

处理器各详细参数及数据集分别如表 3 和表 4 所示。

本文使用 Linux 的性能评估软件 Perf 进行实验测试,针对 ORB-SLAM2 的单目、双目和 RGB-D 3 种传感器输入,分别在 KITTI、EuRoC 和 TUM 数据集上进行了单目实验,KITTI 和 EuRoC 数据集上进行了双目实验以及将 TUM 作为 RGB-D 输入数据集进行 ORB-SLAM2 系统实验,得到 160 个在表 3 所示的 Intel 和 ARM 处理器下测试 ORB-SLAM2 系统中各函数的时间占比。并对这 160 个时间占比结果中排名前 5 的函数进行了统计得到如图 5 所示的热点函数分析结果。

表 3　处理器各详细参数

项目	Intel	ARM
处理器型号	Core i5-6500 CPU	Neoverse-N1 CPU
处理器架构	Skylake	ARM v8
核数	4 核 4 线程	8 核 8 线程
译码宽度	单核 5 路[35]	单核 4 路[36]
调度端口个数	单核 8 个[35]	单核 8 个[36]
主频	3.2 GHz	2.5 GHz
操作系统版本	Ubuntu 16.04.1	Ubuntu 16.04.1
L1 DCache 大小	4×32 kB	8×64 kB
L1 ICache 大小	4×32 kB	8×64 kB
L2 Cache 大小	4×256 kB	8×1024 kB
LLC Cache 大小	6144 kB	32 MB
Cache line 大小	64 B	64 B

由图 5 所示的分析结果可以看出，由于 Perf 评估软件统计函数执行时间是统计函数中没有调用其他函数部分的执行时间，因此包含 FAST 角点提取函数 FAST _ t 和角点分数计算以用来进行非极大值抑制筛选角点的函数 CornerScore 的 ORB 特征提取函数 ExtractorORB 执行时间占比最高为 67%。由于 ExtractorORB 函数为前端视觉里程计部分的热点函数，对于后端非线性优化部分，由图 5 可以看出，优化求解器函数 Solver 执行时间占比最高为 17%，且由于 ORB 特征提取函数 ExtractorORB 和优化求解器函数 Solver 为学术界和工业界热门的研究部分[8,10-11,25-26]，因此本节将重点介绍和分析ORB特征提取函数 ExtractorORB 和优化求解器函数 Solver。

表 4　数据集

KITTI	Data _ odometry _ gray	sequences00-10
	Data _ odometry _ color	sequences00-10
EuRoC	Machine Hall	MH _ 01 _ easy; MH _ 02 _ easy; MH _ 03 _ medium; MH _ 04 _ difficult; MH _ 05 _ difficult
	Vicon Room1	V1 _ 01 _ easy; V1 _ 02 _ median; V1 _ 03 _ difficult
	Vicon Room2	V2 _ 01 _ easy; V2 _ 02 _ median; V2 _ 03 _ difficult
TUM	Freiburg1	360;desk; desk2; floor; plant; room; rpy; teddy; xyz
	Freiburg2	360 _ hemisphere;xyz; large _ no _ loop; 360 _ kidnap; coke; desk; desk _ with _ person; dishes; flowerbouquet; flowerbouquet _ brownbackground; rpy; large _ with _ loop; metallic _ sphere; metallic _ sphere2; pioneer _ 360; pioneer _ slam; pioneer _ slam2; pioneer _ slam3
	Freiburg3	cabinet; teddy;walking _ xyz; walking _ static; large _ cabinet; sitting _ rpy; long _ office _ household; nostructure _ notexture _ far; nostructure _ notexture _ near _ withloop; sitting _ static; sitting _ xyz; sitting _ halfsphere; walking _ rpy; nostructure _ texture _ far; nostructure _ texture _ near _ withloop; structure _ notexture _ far; structure _ notexture _ near; structure _ texture _ far; structure _ texture _ near; walking _ halfsphere;

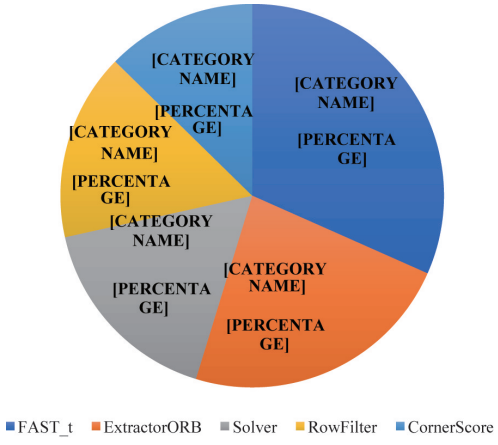

图 5　热点函数分析

作为前端视觉里程计阶段的热点函数 ExtractorORB，主要由跟踪线程调用，在 ORB-SLAM2 系统中，跟踪线程作为主线程首先调用热点函数 ExtractorORB 中 operator 函数进行 ORB 特征点的提取。其中包含 ComputePyramid 函数用来计算图像金字塔；ComputeKeyPointsOctTree 函数对图像金字塔进行角点检测，提取图像金字塔中各层图像的关键点；以及 ComputeDescriptors 函数对图像金字塔的每层计算 BRIEF 描述子。在 ComputeKeyPointsOctTree 函数中调用了时间占比最高的 FAST _ t 函数来进行

FAST 角点的提取。热点函数 ExtractorORB 的函数调用关系如图 6 所示。

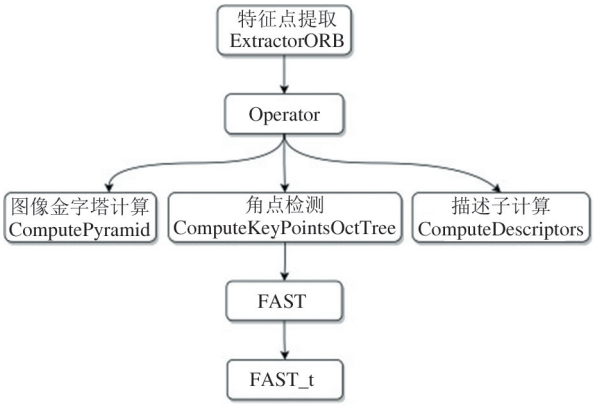

图 6　热点函数 ExtractorORB 的函数调用关系

作为后端优化阶段的热点函数优化求解器函数 Solver 分别由跟踪线程的 GlobalBundleAdjustement、BundleAdjustment 和 PoseOptimization 函数、地图构建线程中的 LocalBundleAdjustment 函数以及回环检测线程中的 OptimizeEssentialGraph 和 OptimizeSim3 函数调用。这 6 个函数均通过调用设置线性求解器 LinearSolver 为 Dense、PCG、Cholmod 之一的方法;设置块求解器 BlockSolver 为是否可变尺寸的方式;设置求解器求解算法 SetAlgorithm 为高斯牛顿(Gauss Newton, GN)、列文伯格-马夸尔特算法(LM)和信赖域(Dogleg)之一;初始化稀疏优化器 SparseOptimizer,并设置局部关键帧顶点以及地图点顶点和优化对应的边,加入优化器中;最后调用优化器的 optimize 函数中的热点函数 Solver 进行迭代优化求出传感器位置的最优解。Solver 函数的函数调用关系如图 7 所示。

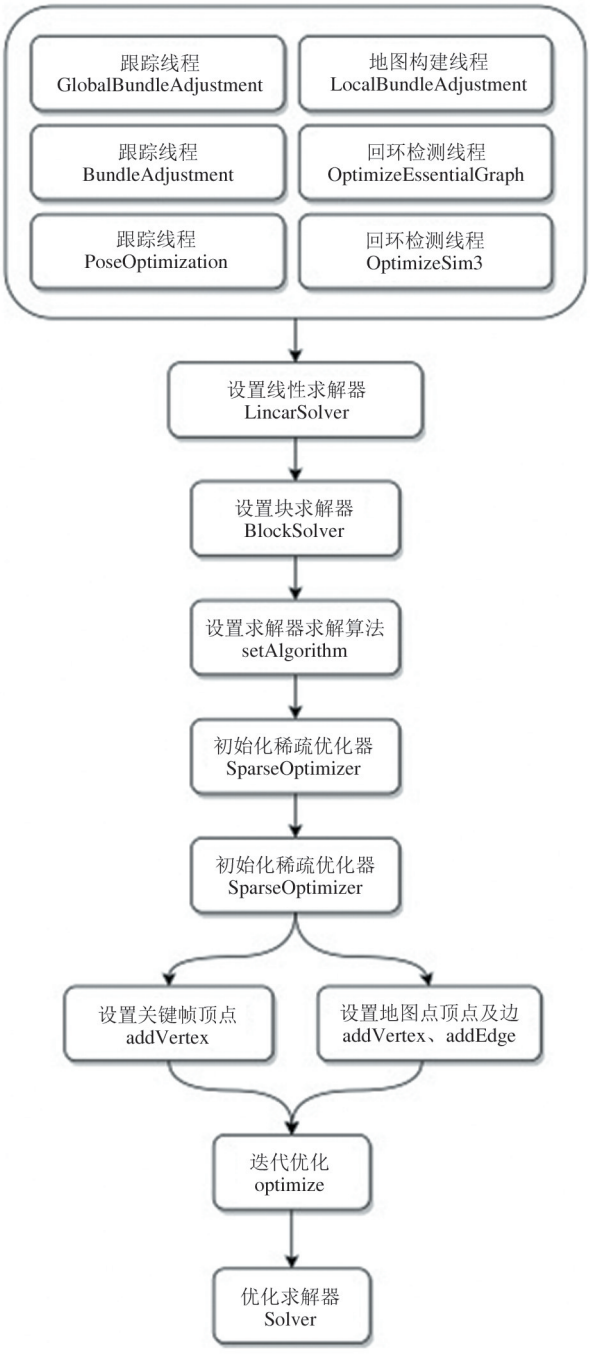

图 7　热点函数 Solver 的函数调用关系

4　ORB-SLAM2 系统特征分析实验

通过对 ORB-SLAM2 系统进行整体性能分析,本节将针对系统中时间占比较高的 ORB 特征提取函数 ExtractorORB 和优化求解器函数 Solver 进行进一步函数特征分析。本文主要从 IPC(instruction per clock)、分支预测失效率、一级数据缓存读失效率、最后一级缓存失效率和最后一级缓存 MPKI 等指标进行函数特征分析。

4.1　IPC

图 8 和图 9 为 2 个热点函数 ExtractorORB 和 Solver 分别在 Intel 和 ARM 处理器对 160 个数据集进行实验得到的 IPC 值对比图。图 10 所示为在图 8 和图 9 基础上计算得到的 2 个热点函数在所有数据集计算得到的 IPC 平均值对比图。

由图 8、图 9 和图 10 可以看出,在 Intel 处理器中ExtractorORB 函数的IPC明显高于Solver函数,是

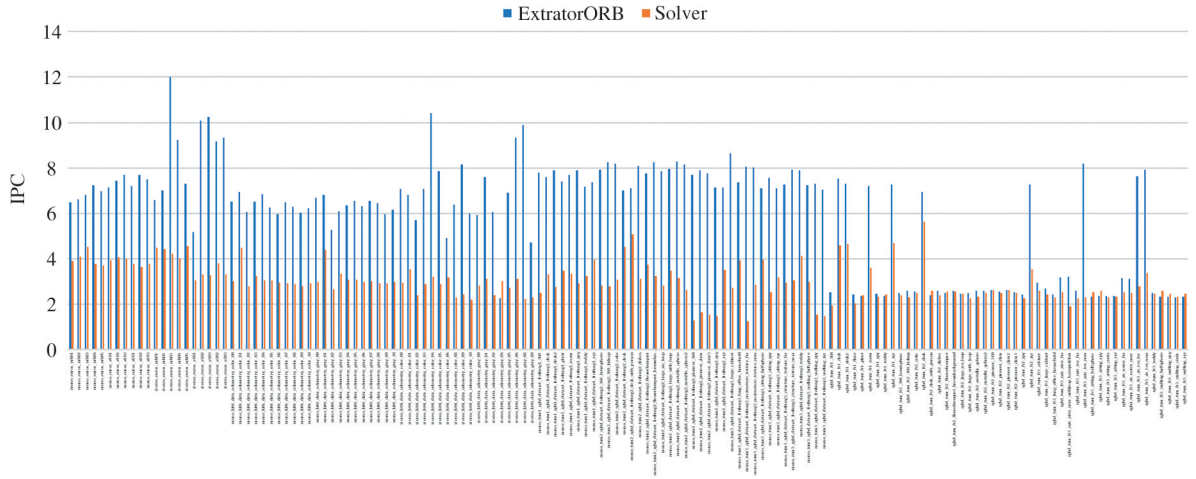

图 8　两个热点函数在 Intel 处理器运行所有数据集的 IPC 值对比图

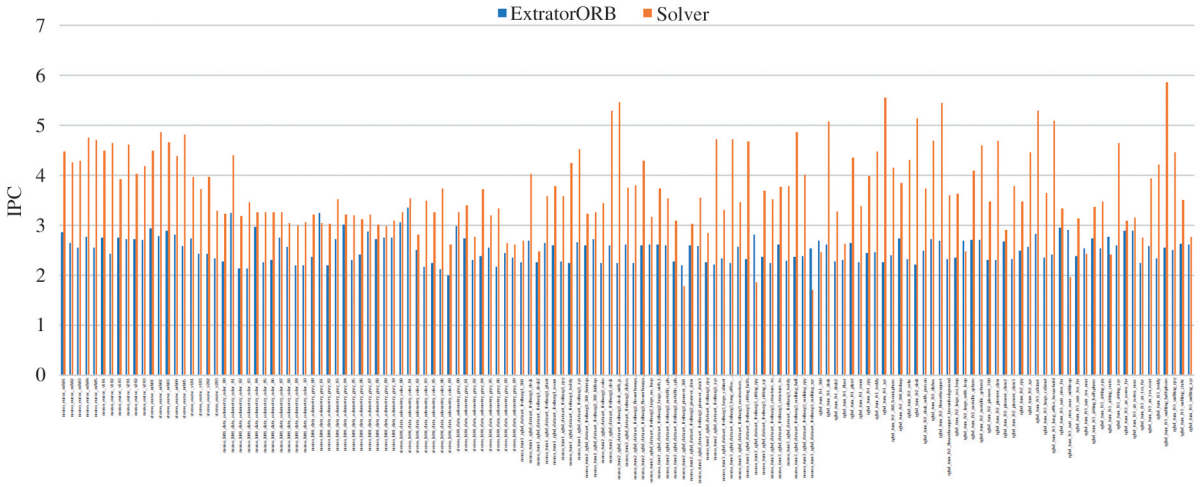

图 9　两个热点函数在 ARM 处理器运行所有数据集的 IPC 值对比图

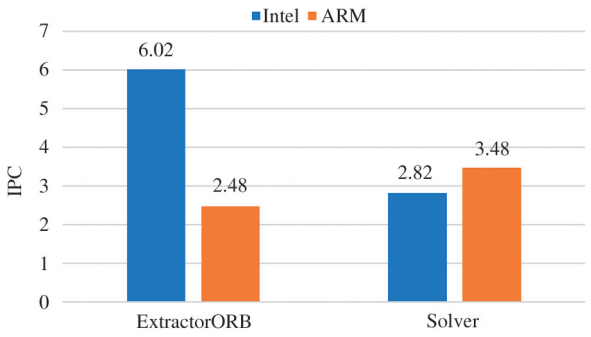

图 10　热点函数的 IPC 平均值对比图

Solver 函数的 2 倍多。由于 ExtractorORB 函数主要功能是进行 ORB 特征提取，在提取 FAST 角点和计算 BRIEF 描述子时主要指令为简单的位运算和比较指令，而 Solver 函数主要功能是进行位姿增量和地标增量的计算，主要指令为矩阵向量的加减乘除运算，相比于 ExtractorORB 函数，指令复杂度较高。且由后文 4.3 节可知，Solver 函数的一级读缓存失效率明显高于 ExtractorORB 函数，则每个 cycle 访存命中率低，每个 cycle 执行的指令要低于 ExtractorORB 函数。而在 ARM 处理器中，ExtractorORB 函数的 IPC 要低于 Solver 函数，主要由于 ARM 处理器使用的是精简指令集（reduced instruction set computer，RISC）系统，与 Intel 使用的复杂指令集（complex instruction set computer，CISC）系统相比每条指令较为简单。针对 Solver 函数中的矩阵向量操作，如乘加操作，ARM 处理器每个 cycle 需要执行乘法和加法 2 条指令，而 Intel 处理器只需要执行 1 条乘加指令，因此在 ARM 处理中，简单位运算操作较多的 ExtractorORB 函数的 IPC 要低于复杂的矩阵向量

操作的 Solver 函数。

4.2　分支预测失效率

图 11 和图 12 为 2 个热点函数 ExtractorORB 和 Solver 分别在 Intel 和 ARM 处理器对 160 个数据集实验得到的分支预测失效率对比图。图 13 和图 14 所示为在图 11 和图 12 基础上计算得到的 2 个热点函数分别在 Intel 和 ARM 2 台处理器执行所有数据集计算得到的分支预测失效率平均值对比图以及在 EuRoC、KITTI 和 TUM 3 个数据集的分支预测失效率平均值对比图。

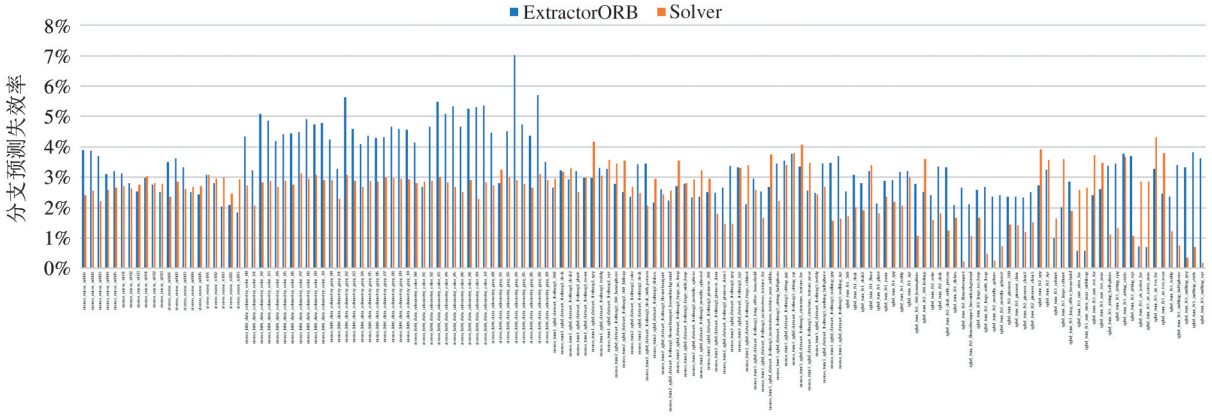

图 11　两个热点函数在 Intel 处理器运行所有数据集的分支预测失效率对比图

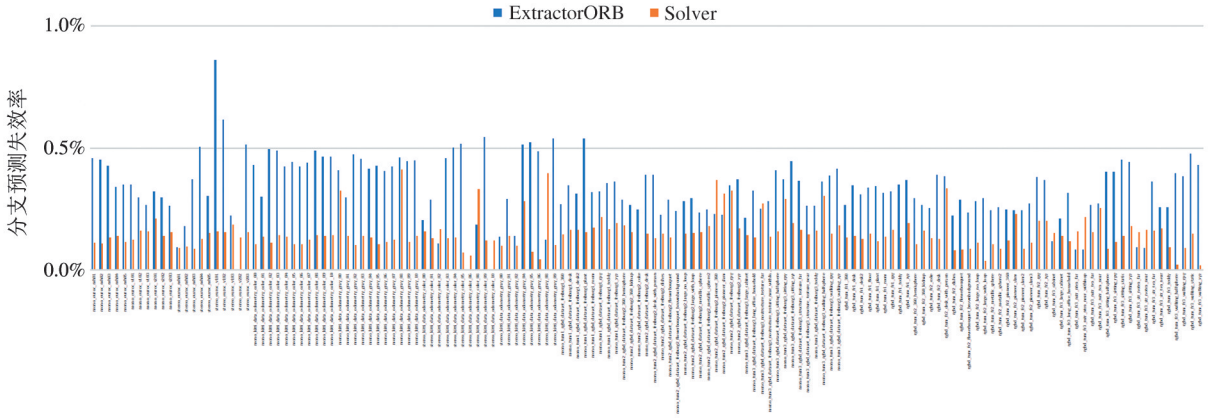

图 12　两个热点函数在 ARM 处理器运行所有数据集的分支预测失效率对比图

由图 11、图 12 和图 13、图 14 实验结果可以看出2个热点函数失效率相似，由于ExtractorORB函数的分支预测指令主要是在对每帧数据进行 FAST 角点提取时进行的对帧中每一个像素点进行的for

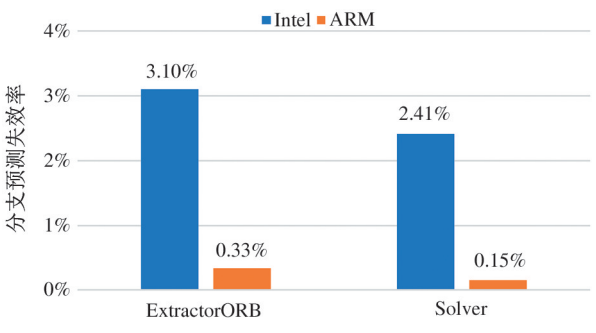

图 13　热点函数的分支预测失效率平均值对比图

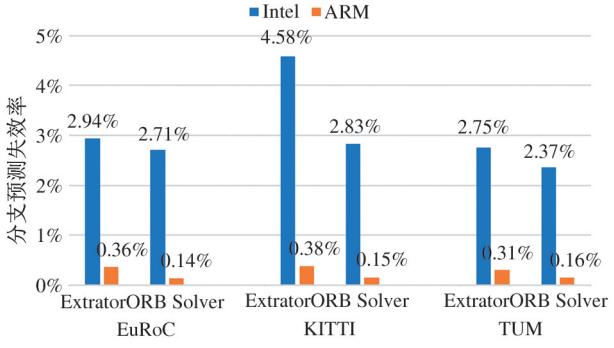

图 14　热点函数在 3 个数据集的分支预测失效率平均值对比图

循环遍历操作，Solver 函数的分支预测指令主要是对多组传感器位姿和路标点数据组成得到的雅克比矩阵进行的 for 循环遍历得到传感器位姿和路标点的增量值。因此 2 个函数的分支预测失效均是由 for 循环产生，因此失效率相似，且在 Intel 和 ARM 2 台处理器中执行表现出来的趋势相似。

4.3　一级数据读缓存读失效率

如图 15 所示为热点函数 ExtractorORB 和 Solver 分别在 Intel 和 ARM 处理器对 160 个数据集实验得到的一级数据缓存读失效率平均值对比图。

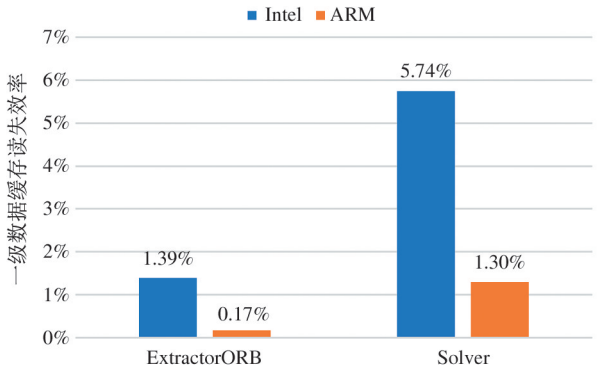

图 15　热点函数的一级数据缓存读失效率平均值对比图

由图 15 实验结果可以看出，ExtractorORB 和 Solver 函数的一级缓存数据命中率较高，但相比于 ExtractorORB 函数，Solver 函数命中率较低。主要由于 ORB-SLAM2 系统中的 ORB 特征提取函数和优化求解器函数整体均具有较好的数据规则访问特征，但由于 ExtractorORB 函数对于数据的访问局部性更好，且数据具有很好的流式读特征；而 Solver 函数的输入数据主要为矩阵、向量，在进行矩阵乘加等操作时，由于输入数据为行存储，但读取数据进行计算的方式有列读取，数据访问局部性相比 ExtractorORB 函数较差，因此 ExtractorORB 函数的一级数据缓存读访问失效率比 Solver 函数较高。相比于 Intel 处理器，2 个热点函数在 ARM 处理器上的一级数据缓存读失效率更低，主要由于 ARM 处理器具有更大的一级数据缓存。

4.4　最后一级缓存失效率

如图 16 所示为 2 个热点函数 ExtractorORB 和 Solver 分别在 Intel 和 ARM 处理器对 160 个数据集实验得到的最后一级缓存失效率平均值对比结果图。

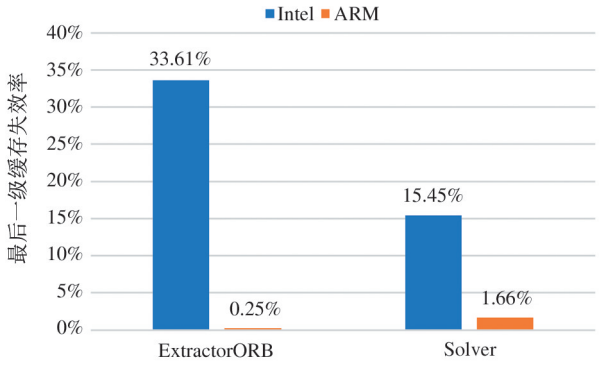

图 16　热点函数的最后一级缓存失效率平均值对比图

由图 16 实验结果可以看出，在 Intel 处理器中 ExtractorORB 函数的最后一级缓存失效率是 Solver 函数的 2 倍。由于 ExtractorORB 函数的数据局部性好，且一级缓存命中率高，因此最后一级缓存命中率低于 Solver 函数。而 ARM 处理器中 2 个函数的最后一级缓存失效率明显低于 Intel 处理器执行结果，主要由于 ARM 处理器的三级缓存大于 Intel 处理器的三级缓存。

4.5　最后一级缓存 MPKI

图 17 和图 18 所示为 2 个热点函数 ExtractorORB 和 Solver 分别在 Intel 和 ARM 处理器对 160 个数据集实验得到的最后一级缓存 MPKI 失效率对比图。

由图 17 和图 18 实验结果可以看出，无论在 Intel 处理器还是 ARM 处理器执行，Solver 函数的最后一级缓存 MPKI 值远大于 ExtractorORB 函数。ExtractorORB 函数指令简单，且数据访问规则、局部性好。由图 19 所示的 2 个热点函数执行所有数据集的访存指令占比的平均值可以看出，Solver 函数的访存指令占比较高，由于 Solver 函数中有大量的矩阵向量计算，因此需要对矩阵向量进行多次访存操作；而 ExtractorORB 函数主要访存和计算为位运算，因此访存指令占比较高，对于每千条指令的最后一级缓存的失效率相比于 Solver 函数较低。Solver 函数相比于 ExtractorORB 函数在进行底层硬件架构设计时需要针对 Solver 函数的矩阵向量数据设计更规则的数据处理结构，以及在对 Solver 函数数据进行运算时设计更有效的向量运算部件以提高Solver函

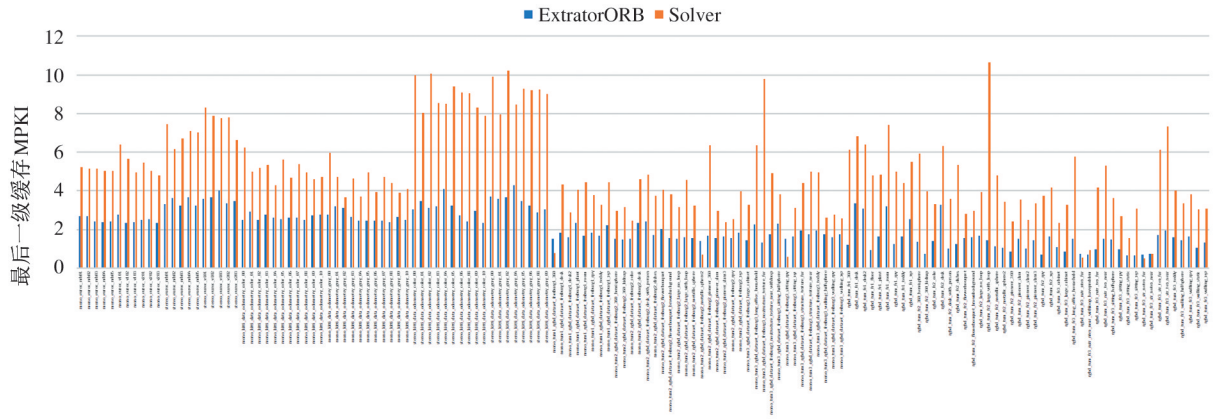

图 17　热点函数在 Intel 处理器执行所有数据集的最后一级缓存 MPKI 值对比图

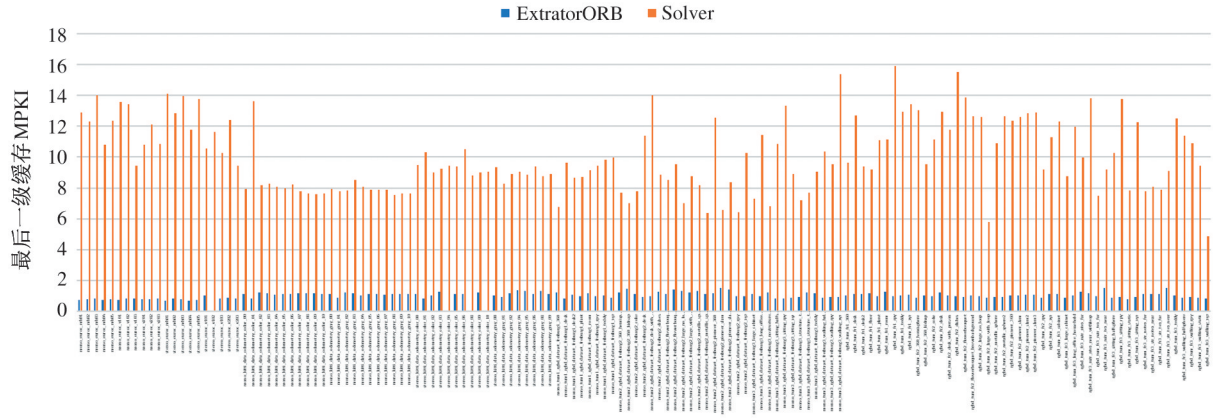

图 18　热点函数在 ARM 处理器执行所有数据集的最后一级缓存 MPKI 值对比图

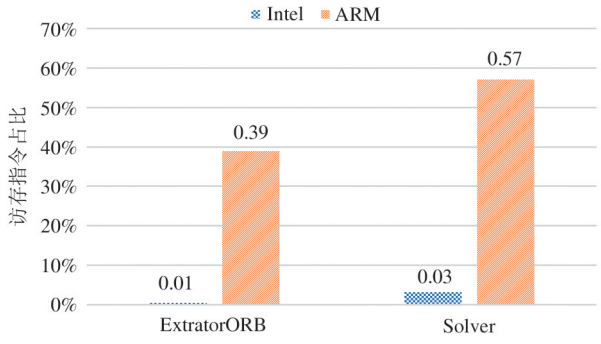

图 19　热点函数执行所有数据集的访存指令占比平均值

数的运算速度,进而提高 ORB-SLAM 系统的性能。

4.6　运算类型特征

通过算法分析可以得出 ORB-SLAM2 系统中的 2 个热点函数具有以下运算特点。

(1) 对于前端视觉里程计部分中的 ExtractorORB 热点函数,其主要为单比特计算,在进行底层硬件架构设计时可通过增加单比特计算逻辑来提高其处理速度。

(2) 对于后端非线性优化部分中的 Solver 热点函数,其主要为浮点计算,在进行底层硬件架构设计时可通过增加浮点乘加部件来提高其处理性能。其计算模式更适用于 Vector 或 SIMD 模式。

5　结　论

随着同步定位和建图技术的广泛应用,基于特征点法的同步定位和建图系统 ORB-SLAM 系统成为热门的研究方向。目前缺乏面向基于 ORB-SLAM 系统底层硬件架构设计指导的系统特征分析的研究。因此本文首先针对 ORB-SLAM 系统进行了详细的介绍;然后对 ORB-SLAM2 系统进行了热点函数实验分析,得到了 ExtractorORB 和 Solver 2 个热点函数;最后对 2 个热点函数进行了 IPC、分支预测失效率、一级数据缓存读失效率、最后一级缓存失效

率和最后一级缓存 MPKI 等特征的评估,为面向 ORB-SLAM 系统的底层硬件架构设计提供了指导性建议。

参考文献

[1] 史殿习, 杨卓越, 金松昌, 等. 面向数据共享的多无人机协同 SLAM 方法[J]. 计算机学报, 2019(5): 983-998

[2] 廖瑞杰, 杨绍发, 孟文霞, 等. SegGraph:室外场景三维点云闭环检测算法[J]. 计算机研究与发展, 2019, 56(2):338-348

[3] 王大伟, 王卓, 王鹏, 等. 基于边缘计算的云原生机器人系统[J]. 智能科学与技术学报, 2020, 2(3): 275-283

[4] DURRANT-WHYTE H, BAILEY T. Simultaneous localization and mapping: part I[J]. *IEEE Robotics and Automation Magazine*, 2006, 13(2):99-110

[5] 任金伟, 郑鑫, 李昱辰, 等. 基于新型多传感器融合策略的移动端双目视觉惯性 SLAM 闭环算法研究[J]. 高技术通讯, 2021, 31(7):681-691

[6] 王化友, 代波, 何玉庆. CFD-SLAM:融合特征与直接法的快速鲁棒 SLAM 系统[J]. 高技术通讯, 2019, 29(12):1224-1238

[7] 朱鸣镝, 陈婵. 基于多特征融合的高鲁棒性视觉 SLAM 改进算法[J]. 智能计算机与应用, 2020, 10(2): 23-28

[8] LIU R, YANG J, CHEN Y, et al. eSLAM: an energy-efficient accelerator for real-time ORB-SLAM on FPGA platform[C]// Proceedings of the 56th Annual Design Automation Conference, Las Vegas, USA, 2019:1-6

[9] RUBLEE E, RABAUD V, KONOLIGE K, et al. ORB: an efficient alternative to SIFT or SURF[C]// Proceedings of International Conference on Computer Vision, Barcelona, Spain, 2011:2564-2571

[10] LI R, WU J, LIU M, et al. HcveAcc: a high-performance and energy-efficient accelerator for tracking task in VSLAM system[C]// Design, Automation and Test in Europe Conference, Grenoble, France, 2020:198-203

[11] ASGANI B, HADIDI R, GHALESHAHI N S, et al. PISCES: power-aware implementation of SLAM by customizing efficient sparse algebra[C]// ACM/IEEE Design Automation Conference, San Francisco, USA, 2020:1-6

[12] MUR-ARTAL R, MONTIEL J M, TARDOS J D. ORB-SLAM: a versatile and accurate monocular SLAM system[J]. *IEEE Transactions on Robotics*, 2015, 31(5): 1147-1163

[13] MUR-ARTAL R, TARDÓS J D. ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras[J]. *IEEE Transactions on Robotics*, 2017, 33(5):1255-1262

[14] CAMPOS C, ELVIRA R, RODRÍGUEZ J J, et al. ORB-SLAM3: an accurate open-source library for visual, visual-inertial and multi-map SLAM[J]. *IEEE Transactions on Robotics*, 2021,37(6): 1874-1890

[15] KLEIN G, MURRAY D. Parallel tracking and mapping for small AR workspaces[C]// Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan, 2007:225-234

[16] FORSTER C, PIZZOLI M, SCARAMUZZA D. SVO: fast semi-direct monocular visual odometry[C]// IEEE International Conference on Robotics and Automation, Hong Kong, China, 2014:15-22

[17] ENGEL J, SCHÖPS T, CREMERS D. LSD-SLAM: large-scale direct monocular SLAM[C]// European Conference on Computer Vision, Zurich, Switzerland, 2014: 834-849

[18] DAVISON A J, REID I D, MOLTON N D, et al. MonoSLAM: real-time single camera SLAM[J]. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007, 29(6):1052-1067

[19] NEWCOMBE R A, LOVEGROVE S J , DAVISON A J. DTAM: dense tracking and mapping in real-time[C]// International Conference on Computer Vision, Barcelona, Spain, 2011:2320-2327

[20] LEUTENEGGER S, LYNEN S, BOSSE M, et al. Keyframe-based visual-inertial odometry using nonlinear optimization[J]. *The International Journal of Robotics Research*, 2015, 34(3):314-334

[21] ENGEL J, KOLTUN V, CREMERS D. Direct sparse odometry[J]. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018, 40(3):611-625

[22] NEWCOMBE R A, IZADI S, HILLIGES O, et al. Kinectfusion: real-time dense surface mapping and tracking[C]// IEEE International Symposium on Mixed and Augmented Reality, Basel, Switzerland, 2011:127-136

[23] KÄHLER O, PRISACARIU V A, REN C Y, et al. Very high frame rate volumetric integration of depth images on mobile devices[J]. *IEEE Transactions on Visualization and Computer Graphics*, 2015, 21(11):1241-1250

[24] WHELAN T, SALAS-MORENO R F, GLOCKER B, et al. Elasticfusion: real-time dense slam and light source estimßation[J]. *The International Journal of Robotics Research*, 2016, 35(14):1697-1716

[25] FANG W, ZHANG Y, YU B, et al. FPGA-based orb feature extraction for real-time visual SLAM[C]// Inter-

national Conference on Field Programmable Technology, Melbourne, Australia, 2017:275-278

[26] CHEN H, DAI Y, XUE R, et al. Towards efficient microarchitecture design of simultaneous localization and mapping in augmented reality era[C]//IEEE 36th International Conference on Computer Design, Orlando, USA, 2018:397-404

[27] ROSTEN E, DRUMMOND T. Machine learning for high-speed corner detection[C]// European Conference on Computer Vision, Berlin, Germany, 2006:430-443

[28] CALONDER M, LEPETIT V, STRECHA C, et al. Brief: binary robust independent elementary features[C]//European Conference on Computer Vision, Berlin, Germany, 2010:778-792

[29] GÓLVEZ-LÓPEZ D, TARDOS J D. Bags of binary words for fast place recognition in image sequences[J]. *IEEE Transactions on Robotics*, 2012, 28(5):1188-1197

[30] LEPETIT V, MORENO-NOGUER F, FUA P. EPnP: an accurate o(n) solution to the PnP problem[J]. *International Journal of Computer Vision*, 2009, 81(2):155

[31] STRASDAT H, MONTIEL J, DAVISON A J. Scale drift-aware large scale monocular SLAM[J]. *Robotics: Science and Systems VI*, 2010, 2(3):7

[32] GEIGER A, LENZ P, STILLER C, et al. Vision meets robotics: the KITTI dataset[J]. *The International Journal of Robotics Research*, 2013, 32(11):1231-1237

[33] STUM J, ENGELHARD N, ENDRES F, et al. A benchmark for the evaluation of RGB-D SLAM systems[C]// IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 2012:573-580

[34] BURRI M, NIKOLIC J, GOHL P, et al. The EuRoC micro aerial vehicle datasets[J]. *The International Journal of Robotics Research*, 2016, 35(10):1157-1163

[35] WikiChip. Skylake(client)-Microarchitectures-Intel[EB/OL]. https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(client):WikiChip LLC,[2021-08-30]

[36] WikiChip. NeoverseN1-Microarchitectures-ARM[EB/OL]. https://en.wikichip.org/wiki/arm_holdings/microarchitectures/neoverse_n1:WikiChip LLC,[2021-08-30]

Analysis of system characteristics of ORB-SLAM

XUE Rui[*,**], LI Yi[*,**], LI Wenming[*], AN Shuqian[*], YE Xiaochun[*], TANG Zhimin[*,**]

([*]State Key Laboratory of Processors, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**]School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049)

Abstract

With the rapid development of applications such as autonomous vehicles, robots, drones, virtual reality, and augmented reality, their core technology, simultaneous localization and mapping (SLAM), has become one of the current hot research fields. As a typical SLAM system based on the feature point method, ORB-SLAM has better robustness and higher computational efficiency. It has been widely concerned both at the system optimization level and the underlying hardware architecture design level. However, there is a lack of system characteristic analysis for the design of the underlying hardware architecture of the ORB-SLAM in academia and industry. This paper introduces the ORB-SLAM in detail from the tracking thread, local mapping thread and loop closing thread. The ORB-SLAM2 is selected for performance analysis experiments, two hotspot functions of ORB feature extraction and block solver are obtained, and the execution characteristics of the two hotspot functions are analyzed. On the Intel i5-6500 and ARM Neoverse-N1 processor platforms, the characteristics of IPC, branch miss rate, L1D read miss rate, LLC miss rate, and LLC MPKI of the two hotspot functions are experimentally evaluated, and the different requirements for architecture design are summarized, which provides guiding suggestions for the underlying hardware architecture design of ORB-SLAM.

Key words: ORB-SLAM, hotspot function, characteristic analysis, hardware architecture design