中图分类号: 学校代码: 10055
UDC: 密级: 公开

南開大學

硕 士 学 位 论 文

结构指导的实时鲁棒高效三维重建系统

Real-time robust and efficient 3D reconstruction system with structural guidance

论文作者 李仕杰 指导教师 程明明 教授

申请学位 工学硕士 培养单位 计算机学院

学科专业 计算机科学与技术 研究方向 计算机视觉

答辩委员会主席 卫金茂 教授 评 阅 人 杨巨峰、卢少平 副教授

南开大学研究生院

二〇一九年五月

南开大学学位论文使用授权书

本人完全了解《南开大学关于研究生学位论文收藏和利用管理办法》关于南开大学 (简称“学校”) 研究生学位论文收藏和利用的管理规定，同意向南开大学提交本人的学位论文电子版及相应的纸质本。

本人了解南开大学拥有在《中华人民共和国著作权法》规定范围内的学位论文使用权，同意在以下几方面向学校授权。即：

1. 学校将学位论文编入《南开大学博硕士学位论文全文数据库》，并作为资料在学校图书馆等场所提供阅览，在校园网上提供论文目录检索、文摘及前 16 页的浏览等信息服务；

2. 学校可以采用影印、缩印或其他复制手段保存学位论文；学校根据规定向教育部指定的收藏和存档单位提交学位论文；

3. 非公开学位论文在解密后的使用权同公开论文。

本人承诺：本人的学位论文是在南开大学学习期间创作完成的作品，并已通过论文答辩；提交的学位论文电子版与纸质本论文的内容一致，如因不同造成不良后果由本人自负。

本人签署本授权书一份（此授权书为论文中一页），交图书馆留存。

学位论文作者暨授权人 (亲笔) 签字：_____

20　　年　　月　　日

南开大学研究生学位论文作者信息

论 文 题 目	结构指导的实时鲁棒高效三维重建系统				
姓　　名	李仕杰	学号	2120160415	答辩日期	2019 年 5 月 30 日
论 文 类 别	博士 ☐　学历硕士 ☑　专业学位硕士 ☐　同等学力硕士 ☐　划☑选择				
学院（单位）	计算机学院		学科/专业 (专业学位) 名称		计算机科学与技术
联 系 电 话	18222916255		电子邮箱		jason.li.nku@gmail.com
通讯地址 (邮编)：天津市津南区同砚路 38 号南开大学信息栋楼 425(300350)					
非公开论文编号			备注		

注：本授权书适用我校授予的所有博士、硕士的学位论文。如已批准为非公开学位论文，须向图书馆提供批准通过的《南开大学研究生申请非公开学位论文审批表》复印件和“非公开学位论文标注说明”页原件。

南开大学学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师指导下进行研究工作所取得的研究成果。除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人创作的、已公开发表或者没有公开发表的作品的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。本学位论文原创性声明的法律责任由本人承担。

学位论文作者签名：_____ 年 月 日

非公开学位论文标注说明

(本页表中填写内容须打印)

根据南开大学有关规定，非公开学位论文须经指导教师同意、作者本人申请和相关部门批准方能标注。未经批准的均为公开学位论文，公开学位论文本说明为空白。

论文题目			
申请密级	☐ 限制 (⩽2 年) ☐ 秘密 (⩽10 年) ☐ 机密 (⩽20 年)		
保密期限	20 年 月 日至 20 年 月 日		
审批表编号		**批准日期**	20 年 月 日

南开大学学位评定委员会办公室盖章 (有效)

注：限制 ★2 年 (可少于 2 年); 秘密 ★10 年 (可少于 10 年); 机密 ★20 年 (可少于 20 年)

摘要

近些年来，随着深度传感器的发展，采用深度传感器的三维重建算法取得了极大的进步。因为深度传感器能够直接获得稠密的深度图像，因而能够重构出更为精准的三维结构。所以三维重建算法被广泛的应用于各个领域。

尽管三维重建算法与过去相比有了显著的提升。仍然有许多问题制约了其在实际生活当中的应用：

1. 位姿估计模块是三维重建算法的重要组成模块。然而由于噪声的存在，位姿估计的结果存在很大的误差。更为严重的是，在一些极端场景下（如少纹理场景），位姿估计算法经常会失败，进而导致整个三维重建系统的失效。
2. 重建的尺度严重制约了三维重建算法在实际场景中的应用，然而现有的数据管理结构并不是十分的高效。
3. 现实世界中，通常存在许多几何结构，如线结构、平面结构，但是现有工作并未很好的利用这些几何结构对算法进行提升。

针对这些问题，我们在本文中提出了以下的改进方法，构成了我们结构指导的实时鲁棒高效三维重建系统：

1. 因为直线结构广泛存在于自然场景当中，且在少纹理场景中具有足够的辨识度。对于位姿估计模块，我们引入了直线结构，提升了位姿估计的精度与鲁棒性。
2. 对于存储效率，我们利用数据分布特性设计了一种半顺序结构来提升系统的存储效率。这种结构既能够只存储有效的数据，又能够减少所需的索引数据。
3. 由于平面结构不仅仅存在于大多数场景中，而且采用体素对于平面结构进行存储往往会消耗很多空间。为了提升系统的存储效率，我们将平面结构引入到我们的模型表达中来。

我们在许多数据集上测试了我们的方法，发现我们的方法均优于传统方法。这证明了我们设计的有效性。

关键词： 三维重建；几何结构；存储效率

Abstract

In recent years, with the development of depth sensors, the three-dimensional reconstruction algorithm using depth sensors has made great progress. Because the depth sensor can directly obtain dense depth images, it can reconstruct a more accurate three-dimensional structure. Therefore, the three-dimensional reconstruction algorithm is widely used in various fields.

Although the 3D reconstruction algorithm has been significantly improved compared to the past. There are still many problems that restrict its application in real life:

1. The pose estimation module is an important component of the 3D reconstruction algorithm. However, due to the presence of noise, there is a large error in the results of the pose estimation. More seriously, in some extreme scenarios (such as less texture scenes), the pose estimation algorithm often fails, which leads to the failure of the entire 3D reconstruction system.
2. The scale of reconstruction severely restricts the application of 3D reconstruction algorithms in practical scenarios. However, the existing data management structure is not very efficient.
3. In the real world, there are usually many geometric structures, such as line structures and planar structures, but the existing work does not make good use of these geometric structures to improve the algorithm.

In response to these problems, we propose the following improvements in this paper, which constitute our robust and efficient real-time 3D reconstruction system:

1. Because linear structures are widely found in natural scenes and have sufficient recognition in less textured scenes. For the pose estimation module, we introduce a linear structure that improves the accuracy and robustness of pose estimation.
2. For storage efficiency, we designed a semi-sequential structure to improve the storage efficiency of the system by using data distribution characteristics. This structure can store only valid data and reduce the required index data.

3. Since the planar structure does not only exist in most scenes, the use of voxels to store planar structures tends to consume a lot of space. In order to improve the storage efficiency of the system, we introduce the planar structure into our model expression.

We tested our methods on many datasets and found that our methods are superior to traditional ones. This proves the effectiveness of our design.

Key Words: 3D Reconstruction; Geometry Structure; Storage Efficiency

目录

第一章 引言

同时定位与建图（Simultaneous Localization And Mapping，SLAM）技术[1–5]对于实现移动机器人自主导航与定位至关重要。近些年来，随着移动机器人技术的迅猛发展，同时定位与建图技术被广泛的应用于增强现实（augmented reality，AR）、无人驾驶汽车（self-driving car），四旋翼无人机（Unmanned Aerial Vehicle，UAV）以及移动机器人（mobile robot）等领域（如图1.1所示）。

图 1.1 同时定位与建图技术的应用场景：(a) 四旋翼无人机 (图片来源[6])，(b) 无人驾驶汽车 (图片来源[7])，(c) 移动机器人 (图片来源[8])，(d) 增强现实 (图片来源[9])

根据使用场景以及使用传感器的类型不同，同时定位与建图技术可以分为不同的种类。如图1.2所示，在水下，由于声波在水中的传播受水的影响较小，因而声呐（sonar）作为主要的传感器被广泛的采用[10]。而在地面上，激光雷达（LiDAR）[11, 12] 和视觉传感器（Camera）[13, 14] 则被广泛的采用。激光雷达通过测

量激光发射与接收之间的时间差来获得测量值。因为测量精准，故而广泛应用于无人驾驶等对于精度要求较高的领域。然而激光雷达过于昂贵，因而采用视觉传感器的廉价方案便应运而生。尽管视觉传感器的测量值不如激光雷达精准，但视觉传感器价格低廉，而且提供了更为丰富的信息，如颜色。因而也被广泛的应用。根据摄像头个数的不同，视觉同时定位与建图方案可以分为单目摄像头（monocular camera）[13, 14] 与双目摄像头（stereo camera）[15] 两种。他们的主要区别是单目摄像头仅仅能获得相对尺度，而双目摄像头可以获得绝对尺度。

图 1.2 视觉同时定位与建图技术常用的传感器（图片来源[16]）：双目相机（左上），双目全向相机（左下），单目相机（右上），单目全向相机（右下）

近些年来，一种新的相机引起了人们的广泛关注——深度相机。与传统的相机仅能得到彩色图像不同，深度相机能同时得到彩色图像和深度图像。随着深度相机的普及，采用深度相机的同时定位与建图方案[17, 18] 也得到了极大的发展。传统的单目或双目视觉同时定位与建图方案得到的地图经常是由稀疏的特征点构成的[11–14]。这是因为单目或双目相机很难得到稠密的深度信息。因为重建的地图过于稀疏，因而这些地图一般只能用于定位，人们很难辨识出场景的具体结构。与之相比，深度相机能够直接获得稠密的深度图像，因而能够重构

出更为准确的三维结构。（如图1.3所示）所以采用深度相机方案的同时定位与建图技术又被称为三维重建，并被广泛应用于环境感知，室内建模等诸多领域。

尽管三维重建算法与过去相比有了显著的提升，仍然有许多问题制约了其在实际生活当中的应用。在本文中，我们提出这些问题。为了解决这些问题，我们提出了一系列的解决方案。这些方案的集合就构成了我们的结构指导的实时鲁棒高效三维重建系统。

位姿估计模块是三维重建算法的重要组成模块，它根据输入图像与重建模型计算出传感器当前的位置和朝向。然而由于噪声的存在，位姿估算的结果存在很大的误差。更为糟糕的是，在一些少纹理场景（如墙壁），由于缺少信息，位姿估计算法经常会失败，进而导致整个三维重建系统的失效。由于直线结构广泛存在于自然场景当中，并且在低纹理场景中具有足够的辨别度。为了克服上述困难，我们在位姿估计模块当中引入直线结构，以提升位姿估计的精度与鲁棒性。这一部分内容在第三章进行详述。

如何存储和表示重建好的三维模型是一个难题。在大多数三维重建算法中，整个重建场景会被划分为均匀的小立方体。这些小立方体称作体素（Voxel）。一个场景通常会由成千上万个体素所构成。存储如此众多的体素会消耗大量的空间。因而存储效率严重制约了重建场景的大小。如果想要进行大尺度的重建，存储效率的提升势在必行。过去有一些方法对提升存储效率做出了很多努力。在这些工作的基础之上我们看见了进一步提升存储效率的可能性。在三维重建系统当中，现有的模型存储结构可以分为顺序结构和散列结构。顺序结构不需要额外的索引数据，但会存储空的体素数据导致存储效率的下降。与之相对，散列结构不会存储空的体素数据，但需要存储额外的索引数据，因而也会导致存储效率的下降。因而我们结合顺序结构和散列结构的优点，设计一种新的数据结构，既能够只存储非空的体素数据，又能够利用结构化的信息来减少所需的索引数据。这一部分内容在第四章进行详述。

除此之外，主流的三维重建算法只采用同一种结构来表示整个模型，如空间点，体素。在现实世界中，通常存在许多结构，如线，平面等。如果这些结构被利用，则三维重建算法的性能就会得到提升。在众多结构当中，平面结构不仅仅存在于大多数场景中，而且采用体素对于平面结构进行存储往往会消耗很多空间。为了提升系统的存储效率，平面结构被引入到我们的系统，进行模型表示。通过以上的设计，三维重建系统中的存储效率得到了一定的提升。这一

部分内容在第五章进行详述。

因为采用深度相机的三维重建方法是近年来新兴的研究领域，在本章的剩余部分我们将回顾过去几年间该领域的发展历史以及一些新兴的研究热点。本文剩余章节的组织结构如下：第二章中我们介绍三维重建系统的基础知识以及具体流程，我们的方法就是构建于此的。在第三章中，我们介绍了我们如何将直线结构融入位姿估计模块，以提升位姿估计的精度和鲁棒性。在第四章中我们设计了一个全新的数据结构——结构化跳表。采用该数据结构可以有效地提升三维重建模型的存储效率。第五章中，我们探索了平面结构对于提升三维重建模型存储效率的可能性。我们发现有效的利用平面结构对于提升三维重建模型存储效率有很大的帮助。第六章我们总结了全文并提出了对于未来工作的展望。

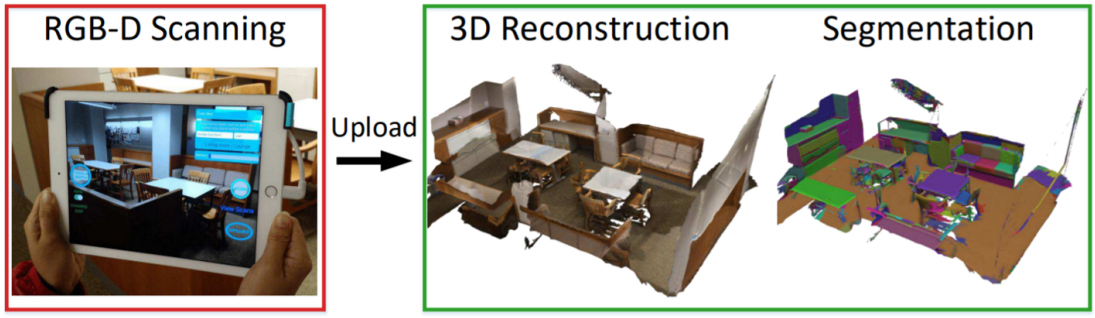

图 1.3 深度传感器及其重建出的三维模型（图片来源[19]）

本章剩余部分我们将回顾过去几年间该领域的发展历史以及一些新兴的研究热点。采用深度相机的三维重建方法是近年来新兴的研究领域，该领域有一些综述性文献[20–22]。在所有方法之中，KinectFusion[17, 18] 是第一个实时的三维重建系统。该方法大力推进了相关领域的发展。在大多数三维重建方法当中，整个场景由体素所表示，每个体素内存储截断符号距离函数的值[23, 24]。这种表示方法也被一些不采用深度相机的三维重建算法所采用，如文献[25, 26] 中采用单目传感器重建整个场景。然而，原始的 KinectFusion 系统仅能用于小范围场景的重建，这是由它过低的存储效率所限制。具体来说，体素的增长速度与重建场景的尺度相关，而不是与重建表面的大小相关。为了缓解这些问题，一些相关的工作被提出，Kintinuous[27, 28] 通过 CPU 内存和 GPU 内存之间的数据交换，将原有系统扩展到大规模重建。在该方法当中，GPU 上使用的体素数量是固定的，称为云切片（cloud slice）。超出云切片范围的场景将被换出。这种方法可行的原因是 CPU 内存通常远大于 GPU 内存。文献 [[29, 30] 为 Kintinuous 的改进版，在

文献[29] 中改进了位姿估计模块，使得位姿跟踪的更为精准。在文献[30] 中，引入了形变这一概念，可以对稠密的地图表示形式进行优化。另一种策略是仅存储重建表面的体素块。通常通过树结构[31, 32] 或散列函数[33, 34] 管理这些数据。

最近，几何结构已经被用于一些单目同时定位与建图体统当中以提升系统的性能。线结构可提高跟踪精度和稳健性。文献[35, 36] 为将特征点法和直线结构相结合的方法。其中文献[35] 为双目系统，文献[36] 为单目系统。文献[37, 38] 为将直线结构和直接法相结合的方法。以上方法中由于对于直线结构的处理方式与点特征类似，因而运行效率不是很高。与以上方法不同，文献[39] 将直线结构在跟踪过程当中看作的一种弱的约束信息，因而得到了很好的运行效率。文献[40, 41] 为深度传感器与直线结构相结合的系统，文献[40] 采用边缘（edge）而非直线结构，同样取得了很好的效果。文献[41] 利用直线结构解决了光照变化条件下的跟踪，得到了很好的效果。从这些方法的实验当中，我们可以看到这些算法在低纹理场景和光照变化环境中相比于传统的基于点特征的方法，具有更好的性能。因而线结构在这些场景当中可以看作是点特征很好的补充。

与线结构相比，平面结构更适合用于三维模型的描述。近些年来也涌现了很多结合平面的工作，如文献[42, 43] 中采用平面结构来简化地图的表示形式。文献[44] 用平面结构来进行去噪，用以提升系统的建模精度。文献[45, 46] 将平面结构用于跟踪，以提升系统的运行效率或跟踪精度。文献[47, 48] 在低纹理场景中应用平面结构，用以提升系统的跟踪精度。从这些工作中，我们可以看到平面模型广泛应用于模型化简和位姿跟踪过程当中。

然而最近的三维重建通常只采用一个结构来描述整个模型。只有少数工作引入了不同的结构[49, 50]。但是这些工作对于多结构模型通常不采取统一的数据管理方法。因而对于不同结构管理的问题非常值得研究。这个问题将在本文后续部分进行描述。

近些年来对于动态非刚体场景重建的研究逐渐增多。DynamicFusio[51] 为 KinectFusion 的进阶版，解决了如何在被测物体也能同时运动的情况下，进行实时的表面重建。由于该算法的性质，也决定了物体表面即使有形变也能准确的实时还原物体表面形状。类似的工作还有文献[52, 53]。

一些工作探索额外传感器的使用对于重建精度的提升，即多传感器的融合。为了提升建图的精度 GravityFusion[54] 将重力信息加入地图表示当中，有效的消除了竖直方向的偏移。文献[55] 采用预积分的方式，融合了惯性测量单元的信息，

提升了位姿估计的精度。

除了基于体素的重建，还有一些工作探索采用新的数据结构表示重建场景。比如基于面元的重建（Surfel），如 ElasticFusion[56]。文献[57] 在文献[56] 的基础之上进一步对光源进行估计。

由于当今计算机视觉的研究主要针对 2D 图像，因而亟需 3D 数据集。近些年来产生了越来越多的 3D 数据集，如 ScanNet[19]、Matterport3d[58]、TUM RGBD Dataset[59] 等等。

近些年来，随着深度学习的发展，将语义信息融入三维重建系统称为了一个热门的研究方向，如语义建图[60–62] 和建立物体级别的地图[63–67]。

第二章 背景知识

我们的方法构建于一个开源的实时三维重建系统——InfiniTAM[68–70]。而InfiniTAM 又以 KinectFusion[17, 18] 为基础。事实上，作为第一个成功的三维重建系统，KinectFusion 中的许多技术细节已经成为三维重建系统中的规范。为了更好的理解我们的系统，我们首先详细的介绍 KinectFusion 的技术细节，在此基础之上对 InfiniTAM 中的改进之处进行概述。

第一节 符号约定

为了保证文章剩余部分符号的统一与规范，在这一小节中，我们将介绍本文的符号约定。

在三维重建系统中，有三个坐标系系统：图像平面（image plane）、相机坐标系（camera frame）和世界坐标系（global frame）。图像平面即相机成像的相平面，相机坐标系即以相机当前所处位置为原点的坐标系系统，世界坐标系可以任意设置，一般设置成以相机初始位置为原点的坐标系系统。与之对应，定义在世界坐标系下的空间点为 $\mathbf{p}_g$，相机坐标系下的空间点为 $\mathbf{p}_k$。而在图像平面上，像素点的坐标表示为 $\mathbf{u} = (u, v)^T$，对应的齐次化坐标为 $\dot{\mathbf{u}} = (\mathbf{u}^T|1)^T \in \mathbb{R}^3$。

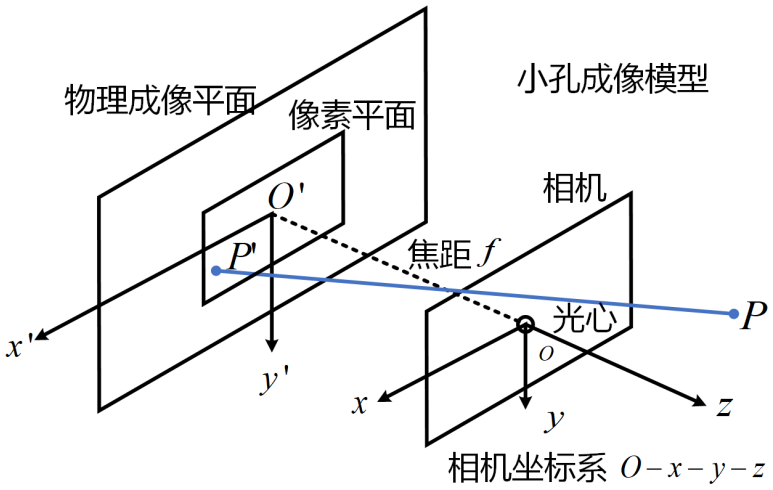

图 2.1 相机模型

KinectFusion 中采用的相机模型为单孔模型（pinhole camera）（如图2.1）[71]。由单孔模型可知，相机的内参矩阵表示为 **K**:

$$\mathbf{K}=\begin{pmatrix} f_x & 0 & p_x & 0 \\ 0 & f_y & f_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.1}$$

其中，f_x，f_y 为相机的焦距，p_x，p_y 为主点偏置，即相机平面的中心点坐标。有了内参矩阵 **K**，则从相机坐标系中一点到像素平面的转换方程为

$$\mathbf{u}=\pi(\mathbf{K}\mathbf{p}_k) \tag{2.2}$$

其中，$\mathbf{q}=\pi(\mathbf{p})\in\mathbb{R}^2$ 为投影变换，对于 $\mathbf{p}=(x,y,z)^T$，有 $\pi(\mathbf{p})=(x/z,y/z)^T$。

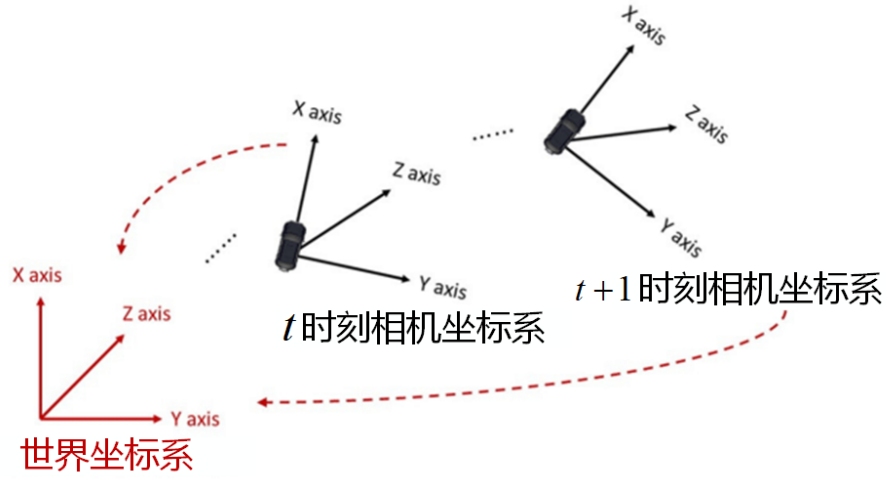

图 2.2　世界坐标与相机坐标系之间的变换关系

为了将相机坐标下的空间点变换到世界坐标下，定义从相机坐标系到世界坐标系的变化矩阵[71]（如图2.2）为：

$$\mathbf{T}_{g,k}=\begin{pmatrix} \mathbf{R}_{g,k} & \mathbf{t}_{g,k} \\ \mathbf{0} & \mathbf{1} \end{pmatrix} \tag{2.3}$$

其中：

$$\mathbf{R}_{g,k}=\begin{pmatrix} cos\alpha cos\gamma-cos\beta sin\alpha sin\gamma & -cos\beta cos\gamma sin\alpha-cos\alpha sin\gamma & sin\alpha sin\beta \\ cos\gamma sin\alpha+cos\alpha cos\beta sin\gamma & cos\alpha cos\beta cos\gamma-sin\alpha sin\gamma & -cos\alpha sin \\ sin\beta sin\gamma & cos\gamma sin\beta & cos\beta \end{pmatrix} \tag{2.4}$$

代表旋转矩阵，α、β、γ 代表绕三轴的旋转角度。$\mathbf{t}_{g,k}=(t_x,t_y,t_z)$ 代表平移分量，t_x，t_y，t_z 分别代表沿三轴的平移量。

第二节 模型表示形式

在 KinectFusion 中，对于重建好的三维模型，采用截断符号距离函数（Truncated Signed Distance Function，TSDF）[23] 进行表示。截断符号距离函数是一种三维地图的表示形式，是三维重建算法中的主流模型表示形式。整个三维模型被表示成一个可以容纳整个重建模型的大立方体（volume）。这个大立方体被均匀的划分为小立方体——体素。每一个体素对应空间中的一个点。

体素中存储的值即为截断距离函数的值。用 $S_k(\mathbf{p})$ 表示存储在体素中的两个量：

- $F_k(\mathbf{p})$：该体素到最近的平面（一般称作零交叉（zero crossing））的距离。
- $W_k(\mathbf{p})$：融合新的测量值时的权重。

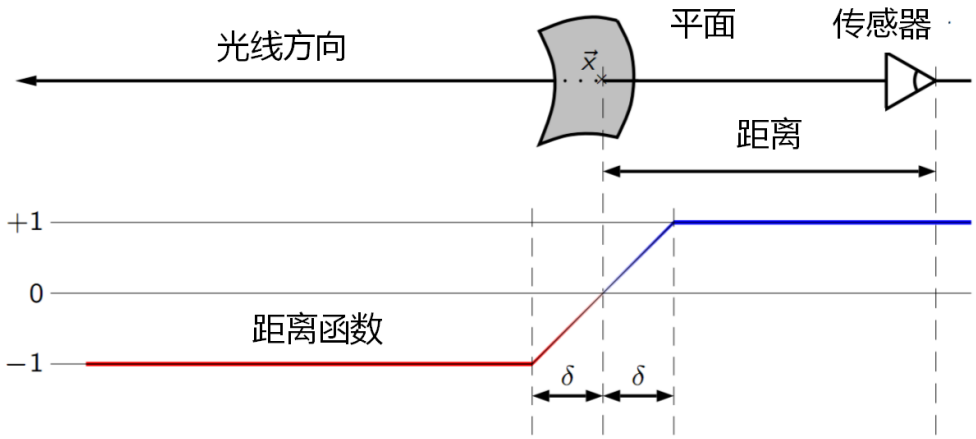

图 2.3 截断符号距离函数：空间点存储沿光线方向到平面的最近距离，可见部分的函数值为正（平面前），不可见部分的函数值为负（平面后），超出 $[-\delta,+\delta]$ 区域范围的函数值被截断为 $\pm\delta$。

截断符号距离函数如图2.3所示。距离函数代表空间点距离最近平面的距离。符号代表函数值有正负之分：可见部分的函数值为正（图2.3中蓝色区域），不可见部分的函数值为负（图2.3中红色区域）。截断意味着线性变化区域被限制在一定范围之内（如图2.3中所示，函数值被限定在 $[-\delta,+\delta]$ 之间），即所谓的截断。假设深度图像提供的深度信息存在的不确定度为 $\pm\mu$（相当于上图的 δ）。换言之，某像素点的深度测量值为 d，则我们可以认为 $d\pm\mu$ 的范围内的点都可能在

平面上。设 r 是距离相机光心的范围，则光线方向上满足 $|r-d|<\mu$ 的 μ 都可能是平面的测量值。相比于其他三维模型的表示方法（如点云），截断符号距离函数的表示方法有利于对不同帧的测量值进行融合。对于测量值的融合在接下来的重建平面的更新部分进行介绍。

第三节 KinectFusion

整个算法的流程共分为四个步骤：

1. 平面测量：首先，从深度相机读入一帧深度图，计算出测量平面（measured surface）：角点图（vertex map）和法向量图（normal map）备用。
2. 传感器位姿估计：然后读入前一帧计算出的平面预测（surface prediction），结合当前帧的测量平面（measured surface），执行迭代最近点算法（iterative closest point，ICP）[72, 73]，得到当前帧对应的相机位姿 $\mathbf{T}_{g,k}$。
3. 平面重建更新：有了相机位姿 $\mathbf{T}_{gk}$，就可以将本帧的测量数据融合到此前保存的全局截断符号距离函数 $[F_{k-1}(\mathbf{p}),W_{k-1}(\mathbf{p})]$ 中，更新全局截断符号距离函数为 $[F_k(\mathbf{p}),W_k(\mathbf{p})]$。
4. 平面预测：最后，根据重建得到的截断符号距离函数 $[F_k(\mathbf{p}),W_k(\mathbf{p})]$ 以及 $\mathbf{T}_{g,k}$，进行该帧的平面预测（反馈到下一个循环的第 2 步）。

接下来，我们分小节对以上步骤进行解析。

2.3.1 平面测量

在该步骤中，输入是深度相机获得的原始深度图，输出的是该深度图对应的角点图（vertex map）和法向量图（normal map）。

在 k 时刻，从深度相机传来原始的深度图 $\mathbf{R}_k$。$\mathbf{R}_k$ 中像素 $\mathbf{u}=(u,v)^T$ 对应的深度为 $\mathbf{R}_k((u))$。因为原始的深度图存在大量的噪声（noise）。为了减少噪声的影响，对原始的深度图 $\mathbf{R}_k$ 采用双边滤波（Bilateral Filtering）[74] 进行预处理，得到降噪后的深度图 $\mathbf{D}_k$。为了得到角点图，只需要将深度图 $\mathbf{D}_k$ 中的像素点投影到相机坐标系下。从像素坐标到相机坐标的转换方程为：

$$\mathbf{V}_k(\mathbf{u})=\mathbf{D}_k(\mathbf{u})\mathbf{K}^{-1}(u) \tag{2.5}$$

对深度图 $\mathbf{R}_k$ 逐像素进行计算，就可以得到角点图 $\mathbf{V}_k$，也就是相机坐标系下的点云图。有了角点图 $\mathbf{V}_k$，就可以通过方向向量的叉乘计算出相应的法向量图 $\mathbf{N}_k$，

对应的公式计算如下：

$$\mathbf{N}_k(\mathbf{u}) = \mathbf{v}[(\mathbf{V}_k(u+1,v) - \mathbf{V}_k(u,v)) \times (\mathbf{V}_k(u,v+1) - \mathbf{V}_k(u,v))] \tag{2.6}$$

上式计算出来的角点图和法向量图都是在相机坐标系下，为了将其转换至世界坐标系，需要进行空间变换：

$$\mathbf{V}_k^g(\mathbf{u}) = \mathbf{T}_{g,k}\mathbf{V}_k(\mathbf{u}) \tag{2.7}$$

$$\mathbf{N}_k^g(\mathbf{u}) = \mathbf{R}_{g,k}\mathbf{N}_k(\mathbf{u}) \tag{2.8}$$

其中 $\mathbf{V}_k^g$ 和 $\mathbf{N}_k^g$ 表示在世界坐标系下的角点图和法向量图。$\mathbf{T}_{g,k}$ 和 $\mathbf{R}_{g,k}$ 代表从相机坐标系到世界坐标系的变换矩阵和旋转矩阵。

通过以上公式，就由原始深度图 $\mathbf{R}_k$ 得到了对应的角点图 $\mathbf{V}_k$ 和法向量图 $\mathbf{N}_k$。

2.3.2 平面重建更新

在这一节中，我们先介绍平面重建更新这一步。在平面重建更新中，假设相机的位姿估计 $\mathbf{T}_{g,k}$ 已经计算出来，要做的就是把当前的测量结果融合到全局地图（global model）中去。

当新来一帧深度图时，需要将其转换为对应的截断符号距离函数：

1. 将从平面到相机方向超过 μ 距离的体素中记录的 $F_k(\mathbf{p})$ 一律截断到 μ，因为我们认为在该空间范围之外的空间是没有物体存在的。
2. 从平面沿着光线方向（对应相机视角下不可见区域）超过 μ 距离的体素，它们记录的 $F_k(\mathbf{p})$ 一律记录为空。因为此时观察不到这些区域，本次测量针对它们的 $F_k(\mathbf{p})$ 应视作无效，不参与接下来的全局截断符号距离函数融合的操作。
3. 其余体素记录相应的截断距离函数值，即该点沿光线方向到平面的距离。将原始深度图 $\mathbf{R}_k$ 转换到世界坐标系下的截断符号距离函数用以下公式表示：

$$F_{R_k}(\mathbf{p}) = \Psi(\lambda^{-1}||\mathbf{t}_{g,k} - \mathbf{p}||_2 - \mathbf{R}_k(x)) \tag{2.9}$$

$$\lambda = ||\mathbf{K}^{-1}\dot{\mathbf{x}}||_2 \tag{2.10}$$

$$\mathbf{x} = \lfloor \boldsymbol{\pi}((K)(t)_{g,k}(p)_k) \rfloor \tag{2.11}$$

$$\Psi(\eta) \begin{cases} min(1, \frac{\eta}{\mu} sgn(\eta)) & iff \quad \eta > \mu \\ null & others \end{cases} \tag{2.12}$$

其中，$\lfloor \rfloor$ 表示的是最邻近查找函数，因为通过透射变换计算得到的 $\boldsymbol{\pi}(\mathbf{K}\mathbf{t}_{g,k}\mathbf{p}_k)$ 不能够保证是整数，但是像素坐标必须为整数，因而我们需要查找 $\boldsymbol{\pi}(\mathbf{K}\mathbf{t}_{g,k}\mathbf{p}_k)$ 最邻近的整数坐标，再根据 $\mathbf{R}_k(\mathbf{x})$ 得到对应的深度。注意，这里 $||\mathbf{t}_{g,k} - \mathbf{p}||$ 前面除了一个系数 λ，是因为 $\mathbf{R}_k(\mathbf{x})$ 是深度值，所以需要把相机到点的距离转为深度。

我们采用加权融合的方式进行模型的更新。假设截止到第 $k-1$ 帧深度图融合得到的截断符号距离函数的值为 $[F_{k-1}(\mathbf{p}), W_{k-1}(\mathbf{p})]$，则更新公式为：

$$F_k(\mathbf{p}) = \frac{W_{k-1}(\mathbf{p})F_{k-1}(\mathbf{p}) + W_{R_k}(\mathbf{p})F_{R_k}(\mathbf{p})}{W_{k-1}(\mathbf{p}) + W_{R_k}(\mathbf{p})} \tag{2.13}$$

$$W_k(\mathbf{p}) = W_{k-1}(\mathbf{p}) + W_{R_k}(\mathbf{p}) \tag{2.14}$$

其中，$F_k(\mathbf{p})$ 和 $W_K(\mathbf{p})$ 对应的是当前帧计算值，$F_{R_k}(\mathbf{p})$ 和 $W_{R_k}(\mathbf{p})$ 对应累积更新量。

总的来说，在这一步里，我们使用式 (2.13) 和式 (2.14) 遍历每个体素，使用当前帧计算它们的 $F_{R_k}(\mathbf{p})$ 和 $W_{R_k}(\mathbf{p})$，并融合到全局截断距离函数中得到更新后的 $F_k(\mathbf{p})$ 和 $W_K(\mathbf{p})$。

2.3.3 平面预测

在之前的平面重建更新中我们进行平面重建的前提是已知相机位姿的估计值 $\mathbf{T}_{g,k}$。在本小节和下一小节中主要解决的问题就是估计相机位姿 $\mathbf{T}_{g,k}$。

三维重建中位姿估计问题的定义如下：已知上一时刻 $k-1$ 的全局模型，位姿估计值 $\mathbf{T}_{g,k-1}$ 和 k 时刻的深度信息 $\mathbf{R}_k$。我们想要估计当前时刻 k 的相机位姿 $\mathbf{T}_{g,k}$。为了得到当前时刻相机位姿 $\mathbf{T}_{g,k}$ 的估计值，我们分为两步进行：

1. 根据 $k-1$ 时刻的全局模型，得到当前时刻的平面预测。
2. 利用 k 时刻的平面测量值和上一步中的平面预测值，使用迭代最近点法预测 k 时刻的相机位姿 $\mathbf{T}_{g,k}$。

这一小节解决的就是第一步中平面预测的问题，也就是根据当前的模型 $[F_k(\mathbf{p}), W_k(\mathbf{p})]$ 与相机位姿 $\mathbf{T}_{g,k}$，预测此时相机能够看到的平面的位置（$F_k = 0$）。并将预测的平面位置存储在角点图 $\hat{\mathbf{V}}_k$ 和法向量图 $\hat{\mathbf{N}}_k$ 中。注意，这里的相机是虚拟的（virtual camera），因为相机的实际测量结果已经在上一节中融合到全局模型中去了，所以我们让一个相机在相同的位置 $\mathbf{T}_{g,k}$ 去观察更新后的模型，计算此时得到的角点图 $\hat{\mathbf{V}}_k$ 和法向量图 $\hat{\mathbf{N}}_k$，所以才称作平面预测。之所以不直接用相机的实际测量结果，而使用重建的模型，是因为：与单次测量结果相比，模型融合了多次的测量结果，因而更为精准。

为得到平面预测值，KinectFusion 使用的是光线透射法（raycasting）。注意光线透射法是在世界坐标下进行的。光线从像素 $\mathbf{u}$ 可以测量的最小深度（受限于深度传感器，由于深度传感器的深度测量是有一定范围的，所以在光线投影的时候还要加上这个限制得到的预测结果才是合理的。）出发。沿着光线方向 $\mathbf{T}_{g,k}\mathbf{K}^{-1}\mathbf{u}$ 行进，直到遇见零交叉（截断符号距离函数从正变到负，$+ve \rightarrow -ve$）认为找到可见平面，加入到角点图中。有两种情况被认为是没有找到平面：

1. 遇到背面（back face）的情况（截断距离函数值从负变到正，$-ve \rightarrow +ve$）。
2. 直到沿着光线搜索完有效的体素还没有遇到零交叉或背面。

这是原理上寻找平面预测值的方法，从效率和准确度的角度考虑，实践中还用到了插值（interpolation）以及增加沿光线搜索步进距离的方式。因为一次前进一个体素的效率比较低，可以使用一个小于 μ 的步长加速寻找过程）。对每个像素点进行一次光线投影，我们就完成了平面预测，结果存储在世界坐标系下的预测角点图和预测法向量图中。

2.3.4 传感器位姿估计

和上一节结合起来，我们接下来就要实现传感器位姿估计了。传感器位姿估计是通过迭代最近点（Iterative Closest Point, ICP）算法[72, 73] 得到的，所以首先我们先简要介绍迭代最近点算法。

迭代最近点算法是一种点云匹配算法。（如图2.4所示）假设我们通过深度相机得到了第一组点云 $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n\}$，深度相机经过位姿变换（旋转加平移）后又得到了第二组点云 $\mathbf{Q} = \{\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_n\}$。注意这里的 $\mathbf{P}$ 和 $\mathbf{Q}$ 的坐标分别对应移动前和移动后的坐标系（即坐标原点始终为相机光心，这里我们有移动前、移动后两个坐标系），并且我们通过相关算法筛选和调整了点云存储的顺序，使得 $\mathbf{P}$，$\mathbf{Q}$ 中的点一一对应，如 $(\mathbf{p}_i, \mathbf{q}_i)$ 在三维空间中对应同一个点。

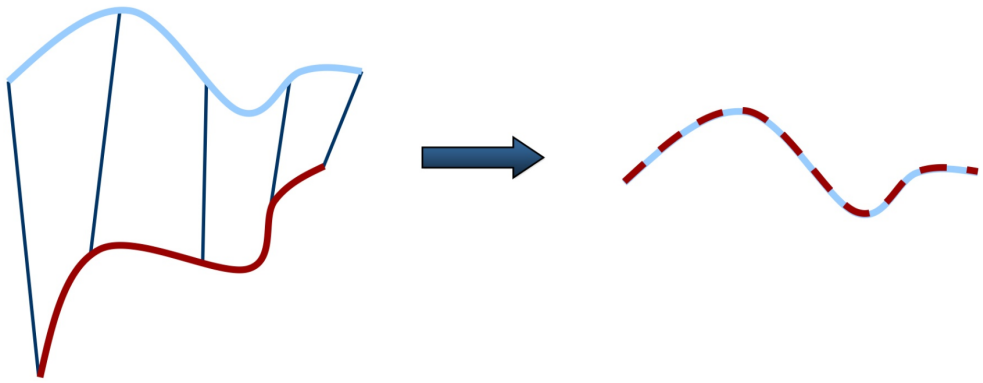

图 2.4 迭代最近点法示意图

现在我们要解决的问题是：计算相机的旋转 $\mathbf{R}$ 和平移 $\mathbf{t}$，在没有误差的情况下，从 $\mathbf{P}$ 坐标系转换到 $\mathbf{Q}$ 的公式为：$\mathbf{q}_i = \mathbf{R}\mathbf{p}_i + \mathbf{t}$。但由于噪声及错误匹配（如 $(\mathbf{p}_i, \mathbf{q}_i)$ 其实并不对应空间中同一点，但特征匹配算法错误地认为二者是同一点）的存在，上式不总是成立，所以我们要最小化的目标函数为。

$$\frac{1}{2}\sum_{i=1}^{n}||\mathbf{q}_i - \mathbf{R}\mathbf{p}_i - \mathbf{t}||^2 \tag{2.15}$$

常用的求解 $\mathbf{R}$ 和 $\mathbf{t}$ 的方法有：

1. 奇异值分解 (Singular Value Decomposition，SVD)
2. 非线性优化

KinectFusion 采用了非线性优化方法的方法进行求解。非线性优化方法通常采用高斯牛顿方法迭代的对误差函数进行最小化。在 KinectFusion 中，对于迭代最近点法，做了一些改进。在 KinectFusion 中输入为上一帧计算出的的平面预测值 $[\hat{\mathbf{V}}_{k-1}, \hat{\mathbf{N}}_{k-1}]$，当前帧深度图计算出的平面测量值 $[\hat{\mathbf{V}}_k, \hat{\mathbf{N}}_k]$。输出为当前相机的位姿估计（pose estimation）$\mathbf{T}_{g,k}$。为了使算法更加鲁棒，在这里，误差函数定义为点到平面的距离（如图2.5），而不是点到点的距离（如图 2.5 所示）：

$$E(\mathbf{T_{g,k}}) = \sum||(\mathbf{T}_{g,k}\dot{\mathbf{V}}_k(u) - \hat{\mathbf{V}}^g_{k-1}(\hat{\mathbf{u}}))^T\hat{\mathbf{N}}^g_{k-1}(\hat{\mathbf{u}})||^2 \tag{2.16}$$

接下来我们只需要对式 (2.16) 进行非线性优化（论文中使用的高斯牛顿法），就可以得到最后的最佳的位姿估计值 $\mathbf{T}_{g,k}$ 了。在高斯牛顿法中需要给 $\mathbf{T}_{g,k}$ 设定一个初始值，一般直接设为上一帧的位姿估计值 $\mathbf{T}_{g,k-1}$。每次迭代得到的解都向着最小化误差函数的方向前进。

迭代最近点法直接配准了两个点云。除了迭代最近点法，在三维重建中直接对相邻彩色图进行帧到帧的配准也是位姿估计的一种常见的作法。称作直接

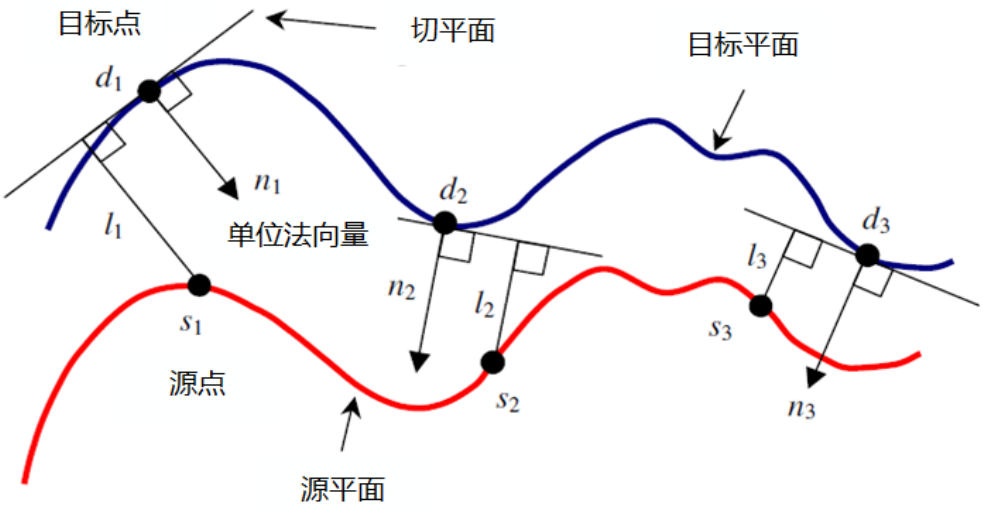

图 2.5 采用点到平面方法的对应关系

法[13, 75]。直接法基于一个假设——灰度不变假设。灰度不变假设是说对于同一个空间点的观测，在各个图像中测量值应该是相同的。换言之，在不同图像中对同一空间点的观测值应该是不变的。然而这一假设由于光照，噪声等因素的影响，并不是完全成立的。因而在直接法中，我们就要最小化这些因素带来的影响。这是通过最小化光度误差（Photometric Error）达到的（示意图见图2.6）。光度误差定义为：

$$\mathbf{e} = \mathbf{I}_1(\mathbf{p}_1) - \mathbf{I}_2(\mathbf{p}_2) \tag{2.17}$$

$\mathbf{P}$ 的世界坐标为 (X, Y, Z)。$\mathbf{p}_1$ 和 $\mathbf{p}_2$ 是图像 $\mathbf{I}_1$ 和 $\mathbf{I}_2$ 对空间点 $\mathbf{P}$ 的观测值。设第一个相机旋转、平移分别为 $\mathbf{I}$、$\mathbf{0}$。第二个相机相对于第一个相机的位姿变换为 $\mathbf{T}$。同时，两相机的内参相同，记为 $\mathbf{K}$。为清楚起见，我们列写完整的投影方程：

$$\mathbf{P}_1 = \begin{bmatrix} u \\ v \end{bmatrix}_1 = \mathbf{D}\frac{1}{Z_1}\mathbf{K}\mathbf{P} \tag{2.18}$$

$$\mathbf{P}_2 = \begin{bmatrix} u \\ v \end{bmatrix}_2 = \mathbf{D}\frac{1}{Z_2}\mathbf{K}\mathbf{T}\mathbf{P} \tag{2.19}$$

其中 Z_1，Z_2 是 $\mathbf{P}$ 在两个相机坐标系下的深度坐标值。$\mathbf{D}$ 为齐次坐标到非齐次坐

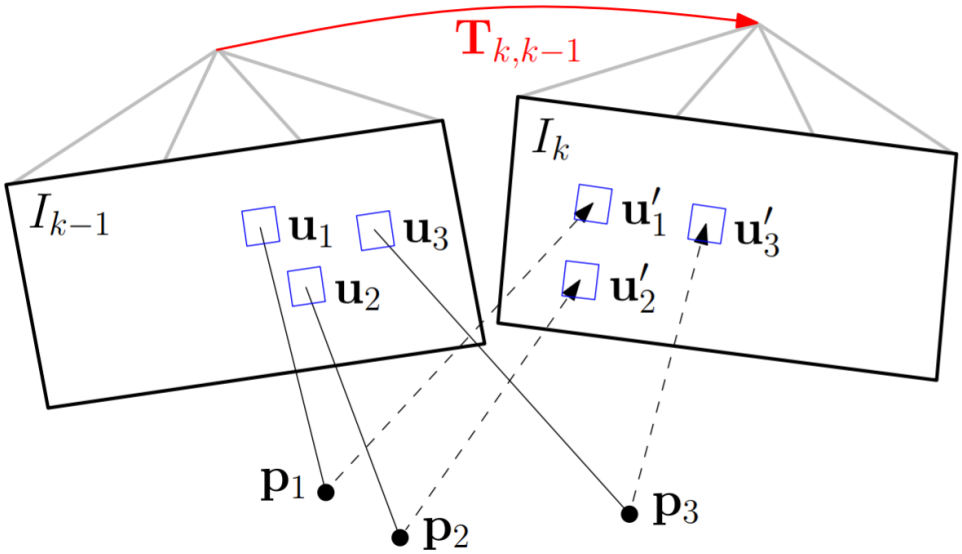

图 2.6 光度误差示意图（图片来源[6]）。同一个空间点（如 $\mathbf{p}_1$），在不同帧上的观测值（灰度值 u_1，u_2）应该相同。

标的转换矩阵：

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.20}$$

我们优化该误差的二范数：

$$J = \mathbf{e}^T \mathbf{e} \tag{2.21}$$

由于一帧图像中有许多个像素（假设为 N），我们将这些点的误差累计，则整个位姿估计问题的目标函数变为：

$$J = \sum_{i=1}^{N} \mathbf{e}^T \mathbf{e} \tag{2.22}$$

$$\mathbf{e}_i = \mathbf{I}_1(\mathbf{p}_{1,j}) - \mathbf{I}_2(\mathbf{p}_{2,j}) \tag{2.23}$$

对其采用高斯牛顿法进行优化即可得到最终的结果。

第四节 InfiniTAM

在上一节中我们详细的介绍了 KinectFusion。InfiniTAM 的基本流程与 KinectFusion 相同，主要的区别有两点:

- 更为高效的系统实现
- 更为高效的模型管理方式

对于第一点，为了更为高效的实现，InfiniTAM 将整个系统分为运行在 CPU 和 GPU 两部分。高并行的算法均运行在 GPU 上，并与 CPU 通过通信相连接。通过这种方法，InfiniTAM 能够以极高的效率运行。整个系统的如图2.7示意。

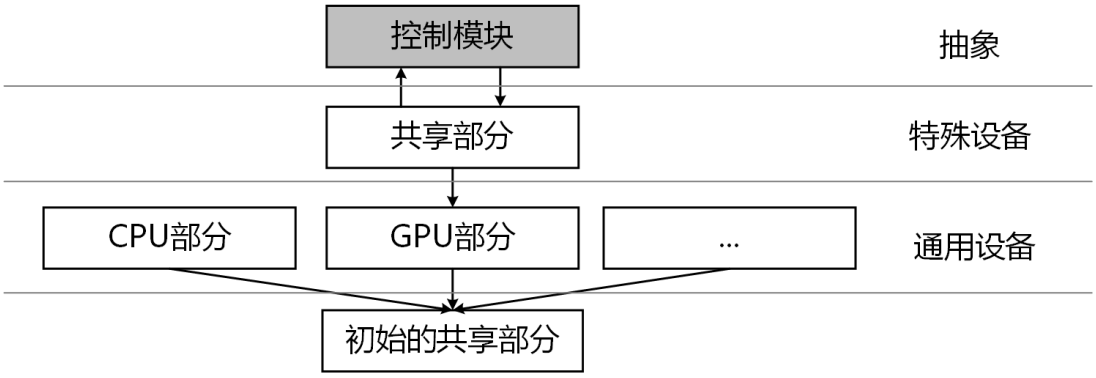

图 2.7 InfiniTAM 系统架构

对于第二点，InfiniTAM 采用了一种新的模型存储与管理方式——Voxel-Hashing[33]。因为 KinectFusion 重建时，整个空间被划分成了等大小的体素，存储数以千计的体素消耗了大量的显存，因而难以重建大尺度场景。除此之外大量体素的更新需要大量的 GPU 资源，因而整个系统的运行效率也不是很高。为了解决这个问题，VoxelHashing 算法应运而生。与 KinectFusion 算法相比，VoxelHashing 只存储在场景表面附近的体素，而不是存储整个空间中所有的体素，因而能够节省显存。为了管理存储的体素，VoxelHashing 采用散列表的形式来存储和管理场景表面划分的体素块（voxel block）。体素块是由 $8 \times 8 \times 8$ 个体素所构成，方便管理和查询。下面我们就来介绍一下 VoxelHashing 算法。

2.4.1 VoxelHashing

VoxelHashing 示意图如图2.8所示。待重建空间可以划分为无数个网格，每个网格代表一个体素块，即个体素。与 KinectFusion 一样，每个体素存储截断符号距离函数值，颜色值和权重值等信息。所有的体素块存储在一个数组当中，由一个散列表统一进行管理。如图所示，散列表由表块所构成，每一个表块 n 个散列项所构成（图中为 4 个）。每一个散列项都存储着对应体素块的空间坐标，为解决散列碰撞（hash collision）的偏移（offset）值，和指向体素块实际存储的空间地址。

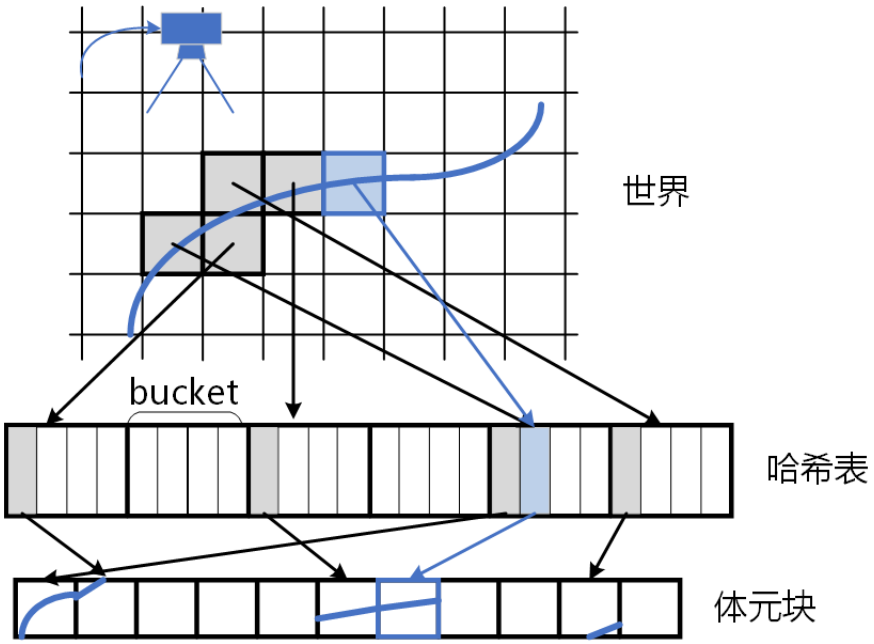

图 2.8 VoxelHashing 示意图

散列值通过体素块的空间坐标计算得到，公式如下：

$$H(x,y,z) = (x \cdot p_1 \oplus y \cdot p_2 \oplus z \cdot p_3) \quad mod \quad n \tag{2.24}$$

采用散列表管理体素块一个很严重的问题是存在不同的体素块会映射到同一个散列值的情况，这一现象也称作散列冲突（hash collision）. 为了解决这一问题，VoxelHashing 采用了表块中的下一个散列项进行存储。详细来说，解决散列冲突的方法是，将散列表划分为表块。每个表块有 n 个散列项（图中为 4 项）。当不同的体素块映射到同一个散列值的时侯，采用表块内一个空的散列项进行存储。极端的情况是表块内所有的散列项都被占满了。对于极端情况的解决，我们在下面的插入操作中进行详解。

在使用散列表进行数据管理时，会涉及到几个操作，下面一一进行说明。

插入操作

在将当前帧信息融合到模型之前，需要在显存中分配在相机视场内但不在显存中的体素块。首先要根据体素块的空间坐标，计算出散列值，再在散列表中对应的表块中查找该体素块是否已经分配，如果该体素块已经分配了，则返回。否则就在表块中查找空闲的散列项，如果在表块中存在空余的散列项，则将体素块的散列项插入，如果表块已经被占满了，则在下一个表块中查找空闲

的散列项。如果在下一个表块中查找的散列项不是表块末尾的散列项，则找到插入的位置。

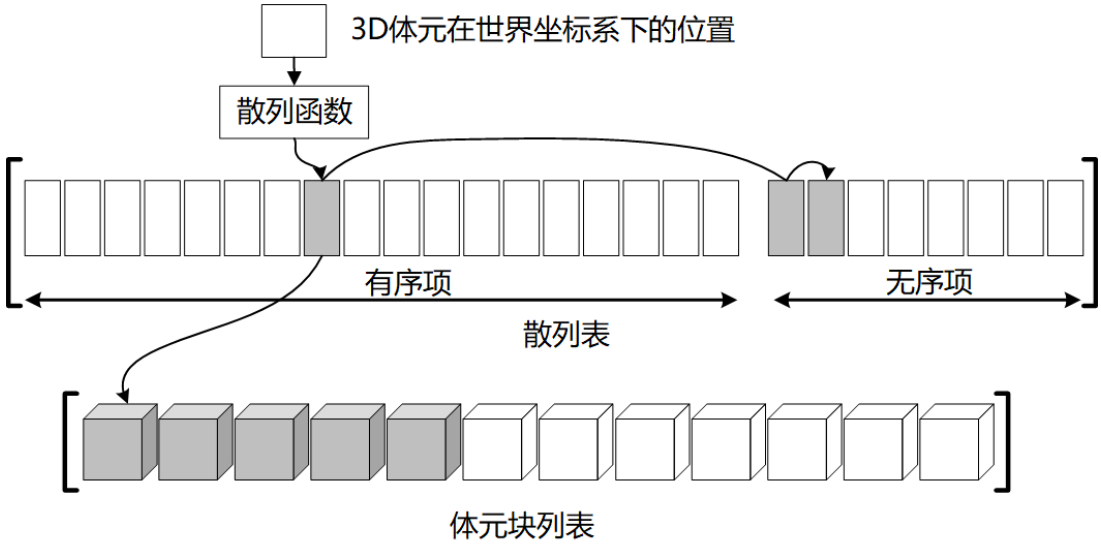

图 2.9 InfiniTAM 数据管理方式

索引操作

根据体素块的世界坐标计算散列值，进而在散列表中得到表块的位置，通过比较散列项中存储的体素块的坐标值，和待索引的体素块的坐标值，可以判断当前散列项存储的是否是待索引的体素块的信息。值得注意的是为了解决散列冲突，每个表块中的散列项存储的可能是上面的表块对应散列值的信息，并且，在表块中遍历每个散列项和待索取的散列块的坐标做对比时，碰到空的散列项需要继续向下寻找。因为由于散列删除，表块会产生空的散列项。

删除操作

散列项的删除操作分为三种情况：

- 如果待删除的散列项在散列值对应的表块中，并且不是末尾项，或者是末尾项，但是末尾项的偏置为空，则在表块中直接将散列项删除。
- 如果待删除的散列项是散列值对应的表块中的末尾项，并且末尾项的偏置不为空，则需要根据偏置值找到下一个散列项，并将下一个散列项移动到表块末尾的位置，因为表块末尾的项存储的是表块满时链接的别的表块中散列项的地址，所以这里必须做移动。
- 如果待删除的散列项在别的表块中，则删除该散列项并且修改偏置的值。

与 VoxelHashing 相比，InfiniTAM 对其做了一些调整。InfiniTAM 中取消了表块，而是将索引数据分为了有序部分与无需部分两部分。通过散列表，先寻找在有序部分的存储位置。如果该位置已经存在值，则在无序部分寻找一个空项进行存储，并用索引值将二者相关联。无序项间也是如此进行关联。对应的插入、索引和删除操作也会做细微的改变。因为改变不大，在此不做详述。对应的 InfiniTAM 数据管理的示意图如图2.9所示。

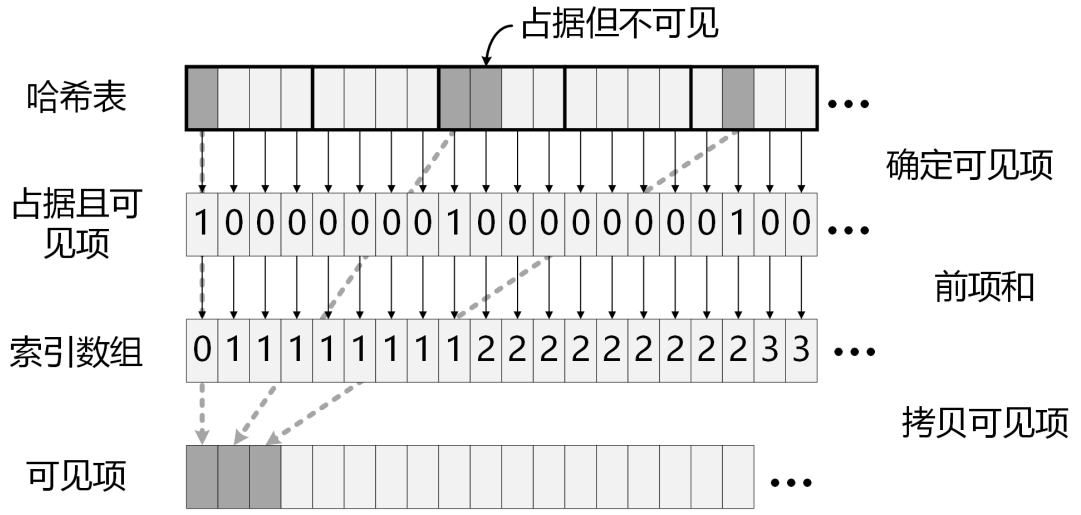

图 2.10 可见列表的更新

2.4.2 数据融合

在数据融合阶段，由于仅存储了重建表面附近的体素。为了高效的执行效率，InfiniTAM 对于 KinectFusion 进行了一些改进。

体素块的分配

在相机的位姿估计得到相机位姿后，我们就要将新的深度图像融合到模型当中去。在 KinectFusion 中，系统存储了整个场景中的体素。然而 InfiniTAM 并未存储整个场景，而是只存储了重建好的平面附近的体素块。因而当新一帧到来之时，我们需要计算在当前相机视角内可见但还未分配的体素块。为此，我们使用在光线投影算法计算当前场景中在重建表面附近可见的体素块，如果体素块没有分配，则在散列表中插入新的散列项，并且分配体素块所需的存储空间。

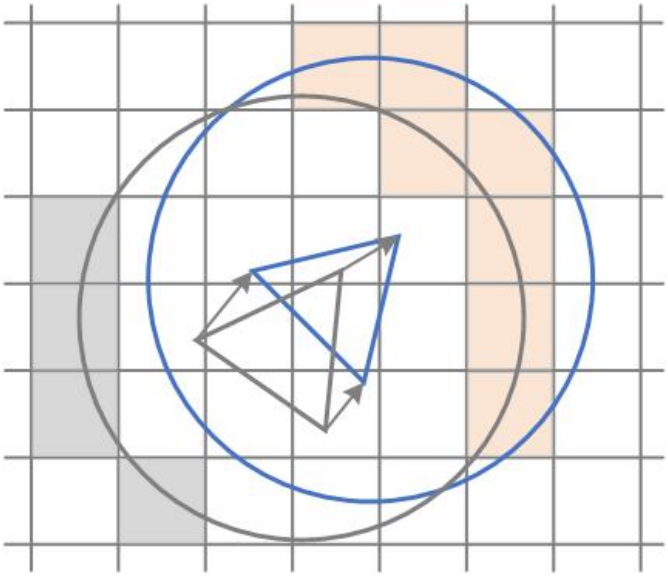

图 2.11 数据的换入换出，橙色体素块被换入设备内存，灰色体素块被换出到主机内存

可见列表的更新

此时在相机视场内所有可见的体素块都存在内存当中。为了更高的执行效率，并不是所有在内存中的体素块都进行更新。因为散列表大部分都是空值，并且体素块许多不在相机的视场内。因而我们先用并行前项和（parallel prefix sum）算法挑选出已经分配并且在相机视场内的体素块，之后仅仅对这些需要更新的体素块进行更新。算法流程如下：首先计算已经有值的并且在对应的体素块在当前帧可见的散列项，然后对获取的 0-1 数组做累加得到索引数组，在索引数组中数字增加的地方就是散列项有值，并且对应的体素块在当前帧可见的，这类散列项拷贝出来，在更新模型时，只更新这类散列项对应的体素块。流程图如图2.10所示。

相机数据融合

相机数据融合的方法于 KinectFusion 类似，都采用了加权融合的方法。唯一不同的地方在于 KinectFusion 的相机位姿估计方法不够鲁棒，因而 InfiniTAM 采用了更为鲁棒的位姿估计方法。基于 KinectFusion 的几何估计存在一些问题，比如对一些走廊等区域，空洞大的区域处理不好。InfiniTAM 则采用几何信息和图像信息融合的方式，利用 GPU 达到了实时位置跟踪。

- 几何估计: 几何估计的基础依旧是 KinectFusion 中的阶段符号距离函数模型，核心依旧是基于点到平面距离的迭代最近点算法。

- 直接法: 在拥有深度信息的前提下，文章利用光流法的思想，通过最小化光测误差（Photometic Error）的方式，获取最优的姿态。

InfiniTAM 采用联合姿态估计的方法，即将两个误差项进行加权叠加:

$$E = E_{icp} + w_{color}E_{color} \tag{2.25}$$

其中 w_{color} 是权重，根据经验设为 0.1。颜色的权重是帧到帧（frame-to-frame）的对齐，而几何权重是帧到模型（frame-to-model）的对齐。

主机/设备内存交换

对于距离相机位置较远且体素块位于设备内存中的情况，InfiniTAM 将其换出到主机内存。相似的，对于距离相机较近但体素块只存在于主机内存的情况，InfiniTAM 将其换入到设备内存。相关示意图间见图2.11。

第三章　直线结构指导的直接法位姿估计

位姿估计算法为了节省计算量，通常利用稀疏点特征进行计算。目前最先进的方法分为从图像中提取特征点的间接法和直接处理像素强度的直接法。然而这些方法在少纹理场景当中（如墙壁）由于信息的缺失经常失败。因为直线结构广泛存在于自然场景当中。近年来，融合直线结构的方法在位姿估计中得到了广泛的应用。虽然在这些方法中，位姿估计的精度和鲁棒性有了一定的提高，但是这些方法大大增加了计算量。这是因为这些方法将直线结构看作独立于点特征的另一种特征，进行了复杂的计算。在这一章中，我们设计了一种全新的方法。它使用直线结构来指导关键点的选择，而不是作为特征。因为直线上的点不仅梯度更大，而且具有共线这一特性，因而被视为比图像其他部分特征点更强的关键点。我们将这些点加入系统当中，使得跟踪精度与鲁棒性有了显著的提升。实验结果表明我们的方法在保证运行效率的前提下跟踪精度和鲁棒性有了一定的提升。

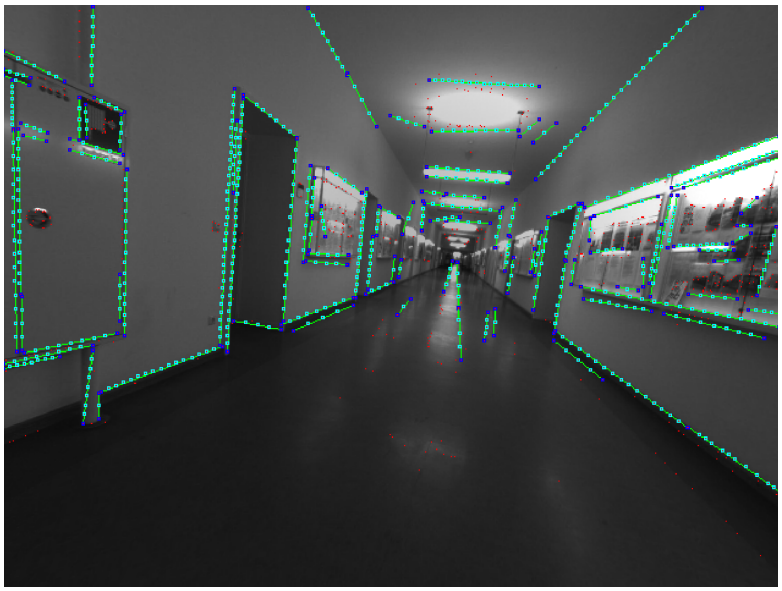

图 3.1　跟踪过程当中的一帧关键帧。我们提取关键帧上的点和线。提取的点和线的位置使用传入帧进行更新。红点表示未激活点 (未激活点)，绿线表示未激活线 (未激活线)。在每一行中，深蓝色的点表示未激活的线端点 (未激活线端点)，淡蓝色的点表示未激活的线公共点 (未激活线普通点)。

我们的方法属于直接法。由于直接法依赖于图像梯度，因而很容易受到光照变化的影响。为了缓解这种影响，我们的方法采用了 DSO[13] 提出的光度标定。光度标定考虑了光照变化的影响，并采用了一个更为复杂的标定模型来缓解这一情况。

整个系统由跟踪和建图两部分构成。在跟踪线程，新的帧与最新的关键帧相匹配。接下来跟踪点和跟踪线被调整。这一部分在第二小节进行描述。在建图线程，首先新的关键帧被创建。之后滑窗滤波器（Sliding Window Filter[76]）对激活关键帧中的所有变量做局部优化。最后新的跟踪点和跟踪线从新创建的关键帧中被提取。这一部分在第三小节进行描述。

第一节 数据管理

在该方法中，逆深度[77] 表示被采用作为点的参数化表示方法。跟踪点根据局部邻域的梯度进行提取和描述。为了提升系统效率，我们不采用复杂的直线描述子，而是从提取到的直线上均匀采样点作为跟踪点。因而直线在系统中由在其上采样得到的跟踪点进行表示。为了描述直线，我们采用其两个端点。根据是否参与优化流程，跟踪点和跟踪线被分为两部分。未激活部分不参与优化过程，激活部分与之相反。对于未激活部分，跟踪点称作未激活点，跟踪线称作未激活线。未激活线的两个端点被称为未激活线端点。未激活线上其余点称为未激活线普通点。与之相对的有，激活点, 激活线, 激活线端点，和激活线普通点。示意图如3.2所示。

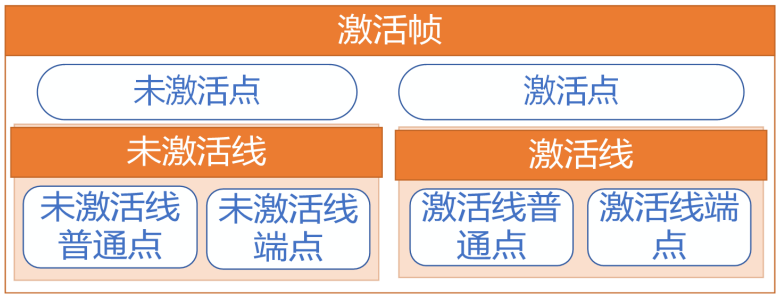

图 3.2 数据管理示意图

第二节 跟踪线程

3.2.1 初始帧跟踪

在这一步中，对新来的帧采用传统的两帧图像直接对齐、多尺度图像金字塔和恒运动模型对输入帧进行初始化，跟踪到最新的关键帧。如果直接图像对齐失败，系统会在最上层的图像金字塔尝试恢复跟踪，方法是使用多达 27 个不同方向的小旋转进行尝试。

3.2.2 线结构指导的调整

未激活点调整

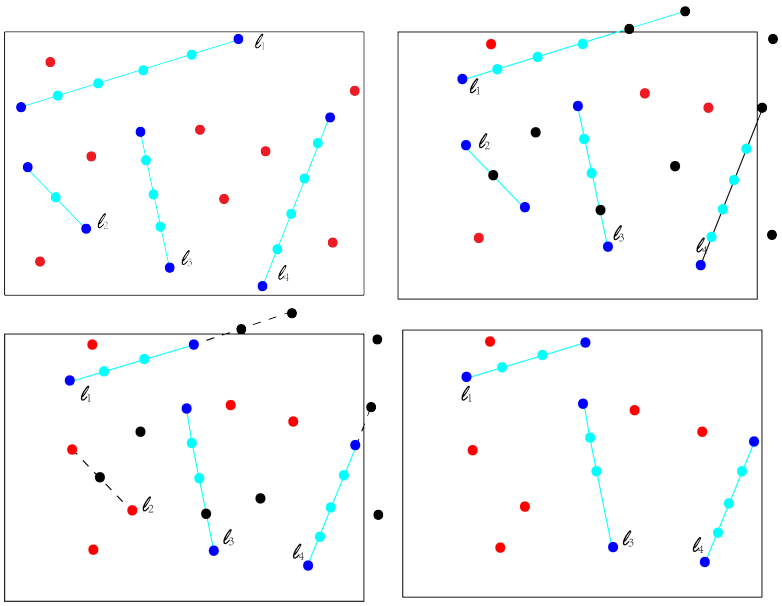

图 3.3 未激活点调整过程。红点代表未激活点，浅蓝色线代表未激活线。在浅蓝色线上，深蓝色点代表未激活端点，浅蓝色点代表未激活普通点。黑色点代表在当前帧跟踪失败。左上图代表关键帧中的未激活点和未激活线。右上图代表匹配帧中的未激活点和未激活线。左下图为未激活点调整过程。l_1 移除一个未激活线端点和未激活线普通点。l_2 移除一个未激活普通点并且只有两个未激活端点剩下。所以这条未激活线被释放，未激活线端点转换为未激活点。余下的直线过程类似。右下图显示最终的结果。

在这一步中，激活关键帧中未成熟点和未成熟线的位置被调整。对激活关键帧中的未激活点，在当前帧中沿极线方向搜索匹配点，以便将光度误差最小化。利用当前帧中的匹配点更新激活关键帧中对应未激活点的深度。如果与未激活点的匹配点不在视图中，或者更新深度的不确定性太大，则将此未激活点标记为调整失败点。

对于未激活线，其上的未激活线普通点和未激活线端点首先按照与未激活点相同的方式进行调整。如果未激活线普通点失败了，我们直接将其从未激活线中移除。对于一个调整失败的未激活线端点，系统会检查同一未激活线上的所有未激活线端点和未激活线普通点的调整状态，并移除调整失败点。新的未激活线由剩余的点组成。当未激活线包含的点数小于 3 时，未激活线就会被释放，未激活线上剩余的点就会转化为未激活点。这是因为任意两点共线。整个未激活点调整过程如3.3所示。

未激活线调整

这一步是在上一步的基础之上更进一步的调整未激活线的位置。上一步留下来的未激活线首先通过将所有未激活线端点和未激活线普通点反投影到三维空间。然后根据这些投影点拟合出一条新的三维直线。之后将这条新生成的直线作为新的未激活线投影到图像平面上。最后，通过将旧的未激活线端点投影到新的未激活线上来得到新的未激活线端点和未激活线普通点。与传统的点独立处理方法不同，我们考虑了同一直线上点之间的关系。在这里，我们使用未激活线信息来指导未激活线端点和未激活线普通点在其上的位置。通过使这些点在它们的生命周期内保持共线，未激活线普通点和未激活线端点结合了梯度和几何信息。随着信息量的增加，系统的精度得到了提高。整个未激活线调整过程如3.4所示。

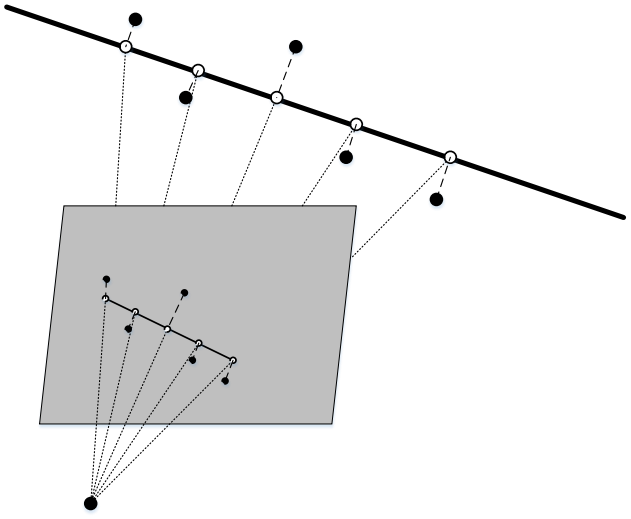

图 3.4　未激活线调整过程

3.2.3 新关键帧的判断

在跟踪线程的最后一步中，系统将会创建一个新的关键帧。新关键帧的创建取决于视图更改、摄像机平移和摄像机曝光时间的变化。这些因素的加权和被作为用于创建一个新的关键帧的标准。

第三节 建图线程

3.3.1 滑窗优化

在该工作中，我们采用滑窗滤波器的方法进行局部优化。在优化之前，所有关键帧中的未激活线和未激活点首先被激活为激活点和激活线。未激活线和未激活点的位置作为激活点和激活线中位置的初始值。然后，激活点和激活线参与优化。在优化中，激活线端点和激活线普通点的处理方法与激活点相同。所有帧、点和线的光度误差定义为:

$$E_{photometric} = E_p + E_l \tag{3.1}$$

其中 E_P 代表所有激活点的误差。E_l 代表所有激活线的误差。整个式子用高斯牛顿算法进行优化。

激活点的误差定义为：

$$E_{pj} := \sum_{\boldsymbol{p}\in\mathcal{N}_p} w_{\boldsymbol{p}} ||(I_j[\boldsymbol{p'}] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}}(I_i[\boldsymbol{p}] - b_i)||_\gamma, \tag{3.2}$$

其中 $\mathcal{N}_{\boldsymbol{p}}$ 是模板中的像素（参加文献[13]）, t_i 和 t_j 是图像的曝光时间，I_i, I_j 和 $||*||_\gamma$ 是 Huber 范数。$w_{\boldsymbol{p}}$ 是依赖于梯度的权重:

$$w_{\boldsymbol{p}} := \frac{c^2}{c^2 + ||\nabla I_i(\boldsymbol{p})||_2^2}. \tag{3.3}$$

c 是一个固定的值，$\boldsymbol{p'}$ 代表 $\boldsymbol{p}$ 的投影点，其中 $\boldsymbol{p}$ 的逆深度为 $d_{\boldsymbol{p}}$:

$$\boldsymbol{p'} = \pi_{\boldsymbol{C}}(\boldsymbol{R}\pi_{\boldsymbol{C}}^{-1}(\boldsymbol{p}, d_p) + \boldsymbol{t}). \tag{3.4}$$

$\boldsymbol{R}$ 和 $\boldsymbol{t}$ 代表图像 I_i 和 I_j 之间的旋转与平移。$\boldsymbol{C}$ 表示内参矩阵. $\pi_C : \mathbb{R}^3 \to \Omega$ 是使用内参矩阵从相机坐标向图像平面的投影，$\boldsymbol{C}$ 和 $\pi_C^{-1} : \Omega \times \mathbb{R} \to \mathbb{R}^3$ 是逆过程。a_i, a_j, b_i, b_j 为文献[13] 中的光度参数.

直线 $\boldsymbol{l}$ 的光度误差为它上面的点误差的和:

$$E_{lj} := \sum_{e \in \mathcal{E}} E_{ej} + \sum_{p \in \mathcal{P}} E_{pj}, \tag{3.5}$$

其中 $e \in \mathcal{E}$ 代表端点，$p \in \mathcal{P}$ 代表直线上的其他点.

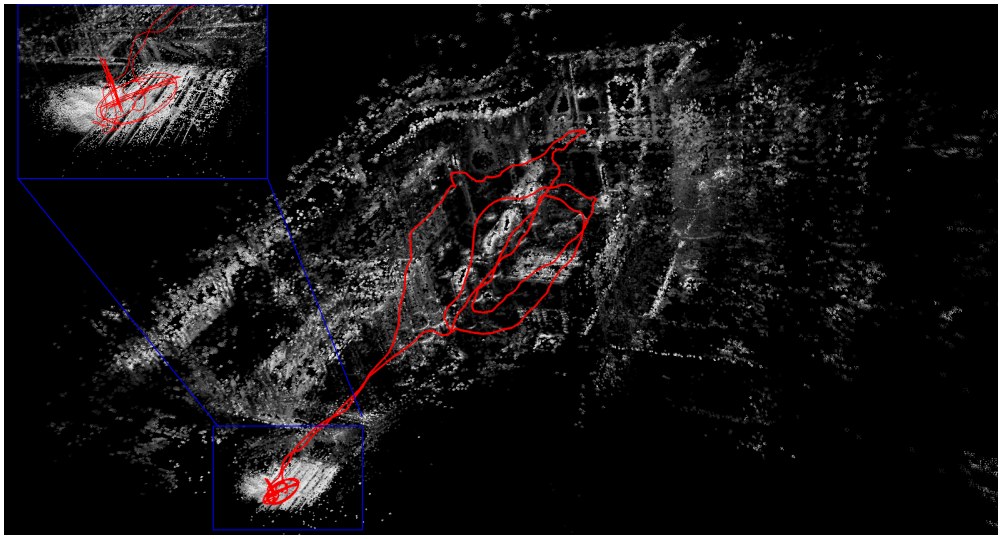

图 3.5 Euroc 数据集上运行结果

3.3.2 激活线调整

在优化过程中，每次迭代后，对激活线进行调整。该过程类似于未激活线调整。在高斯-牛顿算法中，通常每个变量被视作是相互独立的。然而，在我们的情况下，激活线普通点和激活线端点是共线的。在优化过程中，这一特性会发生改变，这是不希望的。因此，每次迭代之后，都会进行激活线调整以保持该特性。最后, 远离当前帧的激活点、激活线和激活帧将被 Schur 补[78] 边缘化。该操作可以使优化中的变量数量保持在一个固定的范围内。因此，计算效率不会随着帧数的增加而降低。

3.3.3 未激活线和未激活点生成

在这个过程中，在新的关键帧上创建未激活线和未激活点。直接法跟踪精度差的主要原因是直接法采用的描述符 (即梯度) 描述能力不是很强。在我们的方法中，我们从线结构中提取点。其中的一个优势是提取的点不仅梯度较大，同时含有线结构信息。除此之外，在直接法中，跟踪点大多存在于直线附近。因

此，我们通过从中选择点的子集来降低计算成本。

未激活点生成

对于未激活点生成，对图像构造了一个三层的图像金字塔。然后在每一层，图像被分割成小块。最后，选择每个小块中梯度值最大的点作为关键点。通过这种方法，所选的点在图像中均匀分布。

未激活线生成

对于线段提取，我们使用了公共 LSD 检测算法[79, 80]。对输入图像首先使用系数为 0.5 的高斯滤波器进行滤波。然后进行直线检测。为了考虑效率，过于接近直线的点将被过滤掉。这里我们对每条线采用自适应距离阈值。可能会有一些线相互重叠，这将导致冗余计算。这里我们使用基于角网格的方法来合并类似的线。我们选择 5° 作为网格大小。所有直线的方向都归一化在 0° 到 180° 度，然后放入相应的网格。在每个网格中，距离最小的线被合并。选择直线的两个端点作为对应的未激活线端点，未激活线普通点从未激活线上均匀采样得到。

第四节 实验结果

3.4.1 TUM RGBD 数据集

我们将我们的方法与现今最好的直接法方法 DSO 与间接法方法 ORB-SLAM 在 TUM RGBD 数据集上进行了对比。实验结果见表3.1。在此实验中，我们选择绝对轨迹误差（Absolute Trajectory Error (ATE)）和相对位姿误差（Relative Pose Error (RPE)）作为我们的评价标准。从表中我们可以看出 ORB-SLAM 得到了最差的结果，并且在很多场景中跟踪失败。DSO 比 ORB-SLAM 结果稍好。我们的方法相比这两个方法取得了更好的结果。

3.4.2 Euroc 数据集[83]

除了跟踪精度，我们还在 Euroc 数据集[83] 上测试了我们的系统的运行时间。（图3.5给出了在该数据集上运行的效果图。）我们选取了不同的方法进行对比，我们的方法取得了最好的运行效率。具体的结果见表3.2。

表 3.1 TUM RGBD 数据集上的实验结果

	绝对轨迹误差 (m)			相对位姿误差 (m)		
序列	Ours	DSO	ORB-SLAM	Ours	DSO	ORB-SLAM
fr1_xyz	0.0538	0.069	0.0836	0.0674	0.0882	0.106
fr2_360_kidnap	0.805	0.799	0.951	0.971	0.973	1.144
fr2_desk	1.33	1.65	1.207	1.64	1.81	1.266
fr2_desk_person	0.412	0.7	0.623	0.434	0.842	1.277
fr2_xyz	0.0653	0.0619	0.119	0.0808	0.0772	0.109
fr3_cabinet	1.05	1.08	-	1.57	1.56	-
fr3_large_cabinet	1.742	1.731	-	2.167	2.1553	2.93
fr3_long_office	1.168	1.18	1.233	1.464	1.512	1.577
fr3_nstr_ntex_far	0.504	0.677	-	0.74	0.876	-
fr3_nstr_tex_far	0.575	0.677	-	0.662	0.837	-
fr3_nstr_tex_loop	0.06	0.0597	0.403	0.667	0.08328	0.507
fr3_nstr_tex_sit_half	0.16	0.21	0.269	0.178	0.259	0.339
fr3_sit_xyz	0.174	0.187	0.343	0.221	0.24	0.413
fr3_str_notex_far	0.865	0.903	-	0.946	1.052	-
fr3_walk_xyz	0.275	0.232	-	0.345	0.282	-

表 3.2 EuRoc 数据集上的跟踪运行时间（毫秒）

	Method	Tracking
EuRoc 数据集	LSD-SLAM[75]	27.6ms
	ORB-SLAM[14]	17.9ms
	S-PTAM[81]	47.3ms
	LIBVISO2[82]	24.9ms
	DSO[13]	13.267ms
	Ours	5.88ms

第四章 结构化跳表

从上述章节的描述中，我们可以总结如下: 现有的三维重建算法的数据管理形式通常可以分为顺序结构和散列结构。顺序结构，例如 KinectFusion 按空间位置存储数据，因此不需要存储额外的索引数据。然而，这种方法需要存储整个空间中的所有体素数据。由于三维空间中非空的体素数据分布是非常稀疏的，因而整个空间中存在大量的空体素，导致了顺序结构的存储效率非常低。为了解决这个问题，散列结构，例如 VoxelHashing，应运而生。散列结构仅仅存储重建表面附近的体素，因而避免了存储大量的空体素。然而由于体素数据在内存中的排列是与其空间位置无关的，所以如何对数据进行管理就成为了一个难题，如索引、插入、删除。为此，一个设计良好的散列表被引入用来管理存储的数据。

通过减少空体素数据的贮存，散列结构在存储效率方面的性能优于有序结构。然而，索引数据通常需要在整个存储中占用大量空间，这会导致存储效率的下降。我们可以看到，顺序结构不需要额外的索引数据，但会存储空体素数据导致存储效率的下降。与之相对，散列结构不会存储空体素数据，但需要额外的索引数据，因而也会导致存储效率的下降。因而我们想结合顺序结构和散列结构的优点，设计一种新的数据结构，既能够只存储非空体素数据，又能够利用结构化的信息来减少所需的索引数据，也就是结构化跳表（Structured Skip List, SSL）。

现实世界中的数据分布通常沿着一个主方向（principal direction），也就是沿着主方向的数据分布比其他方向的数据分布更为均匀。通常来说，因为物体都分布在地面上，所以水平方向通常是主方向。结构化跳表类似于散列结构，仅仅存储重建表面附近的体块，因而与顺序结构相比有较高的存储效率。体素沿着主方向的垂直方向一个接一个地链接起来，形成一个无序的跳表。在主方向上，使用有序索引管理所有跳过列表。因为利用了结构化信息，与散列结构相比，结构化跳表可以减少所需的索引数据。更重要的是，因为利用了数据的分布规律，索引中存在的空项更少。这是因为索引数据是沿着主方向分布的。通过这种设计，结构化跳表可以获得良好的存储性能效率。然而这种设计中存在两个潜在的降低性能的风险：

- 数据碰撞
- 不平衡长度的跳表

对于体素块分配中的数据冲突，散列分配表（Hash Allocation List，HAL）被设计来解决分配冲突密集的问题。为了保持跳表长度的平衡，惯性测量单元（Inertial Measurement Unit，IMU）的测量信息被引入系统。因此，我们的系统在保持了高存储效率的前提下可以以较高的效率运行。

第一节 结构设计

在本节中，我们将给出结构化跳表详尽的描述。我们都知道，数据通常沿主方向分布。为了叙述的方便，在接下来的章节中我们在定义世界坐标系如下: 地面表示为 xy 平面和 z 轴是向上的。为了适应数据的分布规律，结构化跳表被设计成半顺序化的结构：索引数据沿着 xy 平面，是结构化的，然而体素块数据的存储沿着 z 轴，是非结构化的。

结构化跳表由两个列表组成: 结构化索引列表和非结构化体素块列表。相同 xy 值的体素块被链接成一个链表，存储在非结构化体素块列表中。因为同一个跳表中的元素在内存中的存储是不连续的，通过偏移量链接，故称为跳表。结构化索引列表管理所有跳表。索引列表中的每一项存储指向对应跳表起始节点的指针。

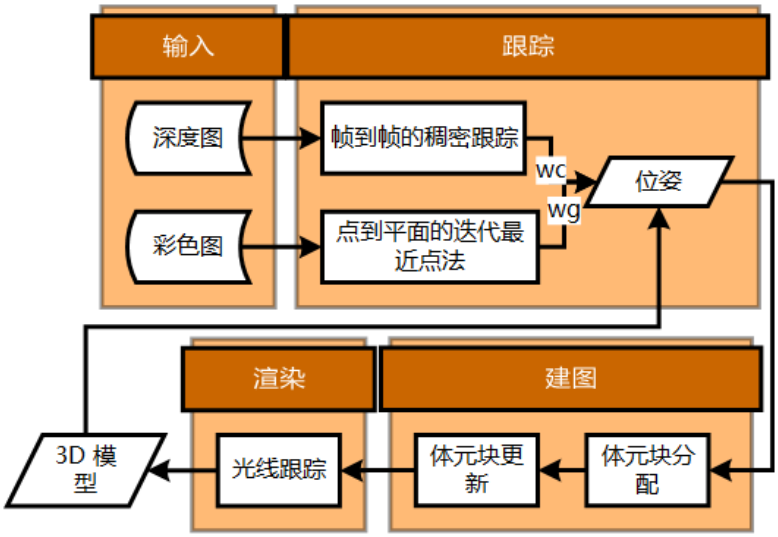

图 4.1 算法流程

整个系统的流程与 InfiniTAM 类似，如图4.1所示。系统分为跟踪，建图和

渲染三大部分。跟踪过程是为了实时估计相机的位姿，与 InfiniTAM 类似，跟踪过程同样是在深度图和彩色图上进行的。对于深度图像，采用点到平面的迭代最近点算法。对于彩色图，采用帧到帧的稠密配准。两个估计的结果通过权重 w_c 和 w_g 相结合。建图部分先检测存在于当前帧但未分配的体素块，并对其在内存中进行分配，其次对整个场景中的体素块进行更新。渲染过程采用光线跟踪算法，最后获得重建好的 3D 模型。

与 VoxelHashing 类似，接下来我们介绍新结构的基本操作。

4.1.1 索引操作

对指定体素的索引操作如下:

- 使用结构化索引列表查找对应的索引项。
- 用 1 中找到的索引项获得对应的跳表起始位置。
- 遍历 2 中得到的跳表，得到对应的位置体素块。
- 在 3 中得到的体素块中搜索，获取目标体素。

如果上述任何一个过程失败，则目标体素不存在内存当中。

4.1.2 分配操作

与 InfiniTAM 类似，当新的帧到来时，未分配的体素块将首先分配。因为对体素块是否已分配的检查是并行的，因而存在严重的数据冲撞，即在一次检查中，同一个未分配的体素块会被检测多次，因而会被分配多次，因而会造成存储效率的降低。为了解决这一问题，我们使用散列分配列表（Hash Allocation List，HAL）来处理这个问题。当待检查的体素块在体素块列表中不存在时，所有对于该体素块的检查会被散列映射函数映射到散列分配表中的相同位置。我们知道，在同时定位与建图系统当中，连续帧的内容是相似的。因此，当前帧中的体素块通常可以在接下来的帧中观察到。因此同一块体素块在接下来的几帧中分配也是可以接受的。为了提升系统的运行效率，被映射到一个散列分配表中被占用的条目体素块在这一帧将不会被分配，而是在接下来的帧中分配。不同帧的体素分配的负担通常是不同的。而我们的设计可以使这些负担均衡的分布在连续的帧中，这对于整个系统的运行效率是有益的。

插入位置 (x,y,z) 被映射到散列分配表的散列函数为:

$$h = ((x \cdot P_1) \oplus (y \cdot P_2)) \mod ALLOCATE\_NUM \tag{4.1}$$

其中 P_1 和 P_2 经实验分别设置为 653 和 654。在我们的设计下，散列分配表在有限的长度下是高效的。整个分配过程如图4.2所示。

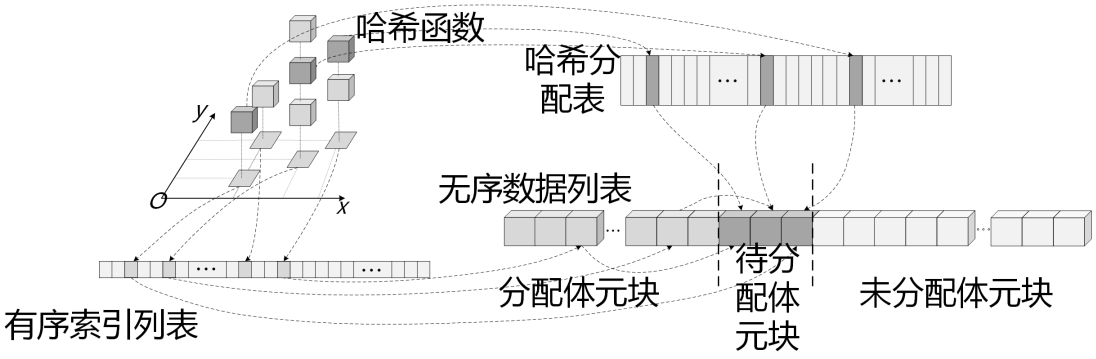

图 4.2 结构化跳表示意图

第二节 IMU 信息的融合

因为三维重建算法运行在 GPU 上，高度并行，而并行算法的时间消耗是由所有并行线程的中运行时间最长的线程决定的。因而当跳表长度不均衡时，对整个算法的运行效率会有极大的影响。因为跳表的索引操作的时间与跳表长度成正比，跳表长度不均衡会造成对每个跳表的操作时间的不同，进而影响运行效率。我们知道，数据通常沿主方向分布。为了保证跳表长度的均衡，IMU 信息被融合用于确定主方向。我们对世界坐标系的设置如下地面在 xy 平面上，z 轴与重力方向相反方向。虽然这种设置在大多数情况下都工作的很好，在某些情况下，这个问题仍然没有得到解决。例如，这种设置对于垂直的结构，如墙，表现效果并不是十分理想。在图4.3中，当 z 轴旋转一个小角度时，我们可以看到跳表的平均长度是减少了的。因此，融合 IMU 信息可以用来减少不平衡跳表长度的影响。

第三节 存储效率的分析

在本节中，我们将对本节中提到的三个结构的存储效率进行详细的分析：顺序结构，散列结构和半顺序结构。在这些方法中，数据被分为索引数据和体素数据两部分。我们定义以下分析中的符号如下:

- N_I：索引数据中的项数。

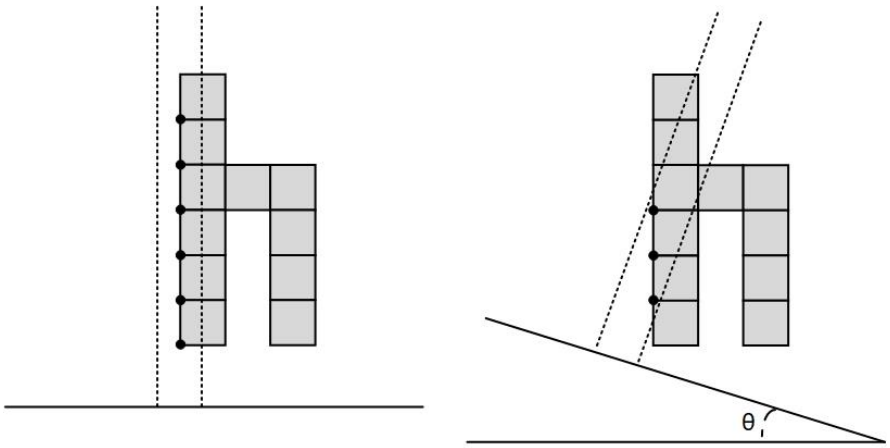

图 4.3 IMU 信息融合

- S_I：存储一个索引项所需要的空间。
- N_V：总体素块的数目。
- N_{NV}：非空体素块的数目。
- S_V：存储体一个元块所需要的空间。
- 我们定义存储效率 E 为：

$$E = \frac{N_{NV}S_V}{N_I S_I + N_V S_V} \tag{4.2}$$

我们期待 E 应该足够高。在以下分析中，我们假设每个方法中的体素数 N_{NV} 是固定的。

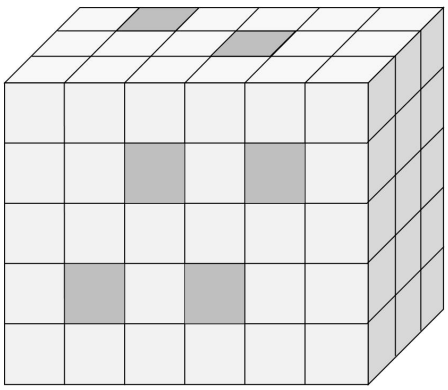

图 4.4 顺序结构

4.3.1 顺序结构

在顺序结构当中，不存在索引数据，因此 $N_I S_I = 0$。以此为代价，所有的体素块在存储空间中都必须保持有序，不管任意体素块有没有数据。只有这样，我们才可以访问任意体素块。如前几章所述，数据在现实世界中的分布是极其稀疏的，因而体素块的存储对于内存空间的利用率极低。换句话说，N_{NV} 远小于 N_V，即 $N_{NV} \ll N_V$。因此, 存储效率极低。我们可以将其视为以稠密矩阵的方式存储一个巨大的稀疏矩阵。如图4.4所示。所有最终存储效率为

$$E_{ordered} = \frac{N_{NV}}{N_V} \approx 0 \tag{4.3}$$

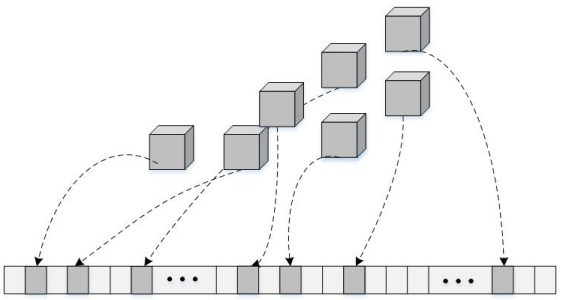

图 4.5 散列结构

4.3.2 散列结构

不同于顺序结构，散列结构只储存在重建表面附近的体素块。因而几乎所有储存的体素块都是有效的，即 $N_{NV} \approx N_V$。若要管理无序存储的体素块数据，则需要引入索引数据（散列表）做为索引数据。在大多数情况下，由于一些管理操作例如交换操作，索引项的数量远大于体素块项。因此 N_I 大于 N_V，即 $N_I \gg N_V$。更糟糕的是，一些体素块数据（体素块坐标）存储在索引数据中，占用空间很大。这是因为索引数据列表和体素块数据列表的长度不同而产生的。由于数据分布的稀疏性，散列结构相比结构化方法具有较好的存储效率。然而，仍然存在进一步压缩存储空间以提升存储效率的可能性。从上面的讨论中，我们可以看出这种无序方法类似于稀疏三维矩阵。体素块数据是这个矩阵中的非空元素，而散列表中的元素则用于数据访问 (如图4.5所示)。

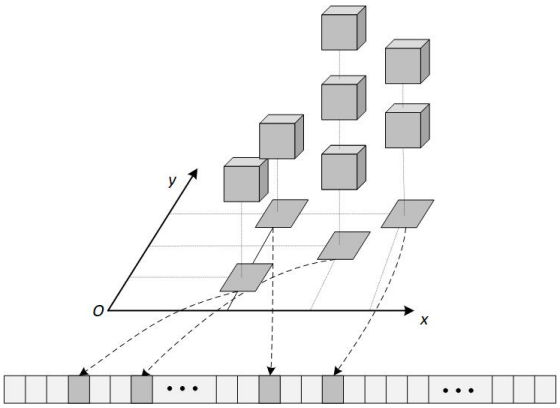

图 4.6 半顺序结构

4.3.3 半顺序结构

与散列结构类似，为了提高存储效率，只有重构表面附近的体素块被存储（$N_{NV} \approx N_V$）。每个索引项中的数据被最小化，也就是只存储了一个指针。所有与体素块相关的数据都被存储在体素块中（$S_I < S_V$）。因为数据沿主方向的分布通常是均匀的，因而几乎所有的索引项都保存有效数据。我们有 $N_I < N_V$，因为一个索引项会指向多个体素块（跳表）。总之，索引项按顺序组织，类似于二维矩阵。体素块数据在每个索引项中被分隔，类似于链表，是一种无序方法。因为半顺序结构结合了顺序结构和散列结构的优点，因而该结构被称为半顺序结构。半顺序结构的示意图如图4.6所示。

第四节 实验结果

在本节中，我们将在两个数据集上对提出的方法进行比较:TUM RGB-D 数据集[59] 和 ScanNet 数据集[19]。本节中的所有算法都在相同的硬件平台上进行评估，平台采用 Intel i7- 4790K (4GHz) CPU 和 NVIDIAGTX 1080 GPU。作为对比，我们选择最先进的开源三维重建系统 InfiniTAM。除了高效的实现，InfiniTAM 还包括用于三维重建的几种主流数据结构，如 VoxelHashing(hash)[33]、VoxelHashing 的一个变种——分层 VoxelHashing(hhash)[84]，以及传统的顺序结构 (ordered)。在我们的实验中, 顺序结构的重构尺寸设置为 $512 \times 512 \times 512$ 个体素。为了比较的公平性，我们的方法均是基于 InfiniTAM 实现的。因此，造成不同的实验结果的主要原因就是不同数据结构的性能差异。

4.4.1 TUM RGB-D 数据集

TUM RGB-D 数据集包含多个序列，均是深度传感器（Kinect）在不同场景中获得的。每个序列包含 RGB 图像、深度图像和加速器数据。

采用结构化跳表重建的一些模型如图4.7所示。数值结果如表4.1所示。在所有比较的方法当中，我们提出的结构化跳表在存储效率上表现得最好。而且，索引数据占所有数据的比例维持了极低的水平 (小于 0.02%)。至于其他方法，存储效率随不同的序列而变化，这意味着我们的结构化跳表在存储效率上比其他方法更为鲁棒和高效。为了保证实验效果的有效性。在实验中，我们将每个序列运行三次，取平均运行时间为最终的运行时间。不同方法的每帧的平均运行时间在表中也进行呈现。我们可以注意到 VoxelHashing 的性能是所有方法中性能最好的。而我们的结构化跳表与之相比也具有类似的性能。并且结构化跳表的运行效率可以满足算法的实时性要求。此外，我们可以看到顺序结构的运行效率比结构化跳表要慢得多。

表 4.1 在 TUM RGBD 数据集上的实验结果，存储效率上的实验结果：存储效率（%）和每帧的平均运行时间（ms）

序列	Ours	hash	hhash	ordered
fr1_desk	99.988	96.124	93.19	0.043
fr3_cabinet	99.982	99.516	99.47	0.036
fr3_str_notex_far	99.986	99.727	99.694	0.043
fr3_str_notex_near	99.982	99.031	99.553	0.011
fr3_str_tex_far	99.988	99.375	99.704	0.021
fr3_str_tex_near	99.983	92.963	98.99	0.173
平均运行时间（ms）	4.99	3.33	4.77	14.81

4.4.2 ScanNet 数据集

ScanNet 数据集与 TUM RGB-D 数据集类似，由从不同场景收集到的序列所组成。每个序列的组成数据均按 TUM RGB-D 数据集提供，即每个序列包含 RGB 图像、深度图像和加速器数据。

一些重建模型如图4.8所示。我们可以从表中得到与上述结果类似的结论。

表 4.2 在 Scannet 数据集上的实验结果：存储效率（%）和每帧的平均运行时间（ms）

序列	Ours	hash	hhash	ordered
0013_00	99.981	99.052	98.045	0.003
0032_00	99.986	98.848	93.857	-
0036_00	99.981	98.55	96.218	-
0047_00	99.988	97.514	93.484	0.005
0049_00	99.985	98.299	98.291	-
平均运行时间（ms）	3.92	3.4	2.83	11.75

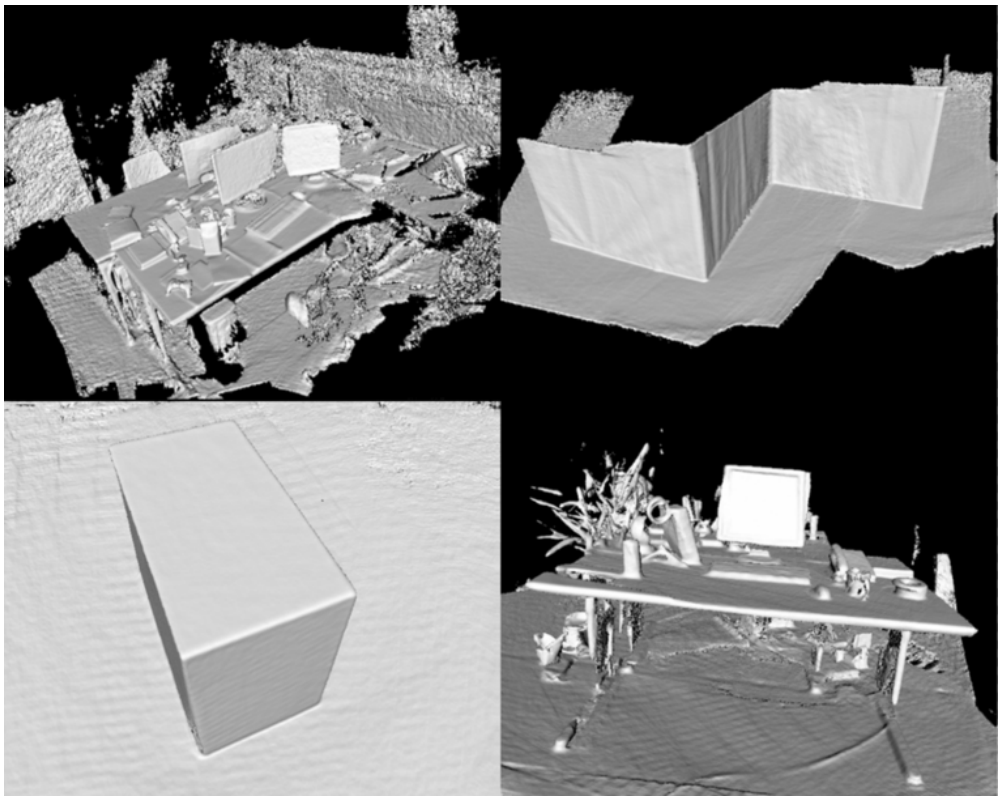

图 4.7 结构化跳表在 TUM RGB-D 数据集上的重建结构

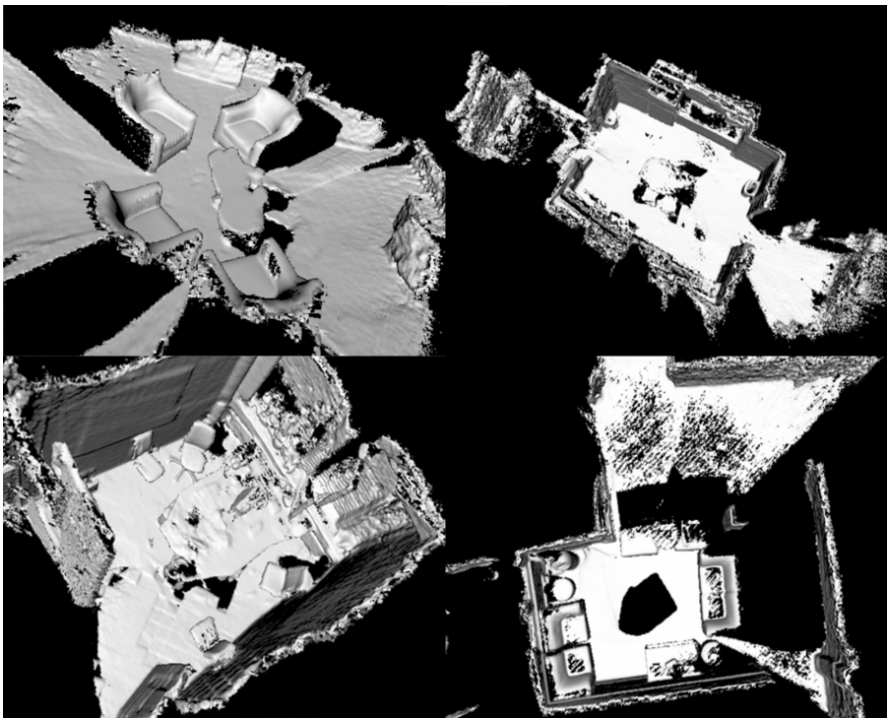

图 4.8 结构化跳表在 ScanNet 数据集上的重建结构

具体来说，结构化跳表在所有序列上都达到了最佳的存储效率，并且整个算法更加的鲁棒。唯一的代价就是一点额外的时间消耗。然而这点消耗远不足以影响系统的实时性。值得注意的是，由于重建尺度的限制（$512 \times 512 \times 512$），顺序结构在一些序列中的重建失败。整个实验结果如表4.2所示。

第五章 平面结构的利用

我们可以看到以上所有工作只使用了单一结构来表示整个模型。然而在现实世界中，通常存在许多几何结构，如线结构、平面结构等等。如果这些结构能够被利用，则三维重建算法的性能就会得到提升。但是主流的三维重建算法只采用同一种结构来管理整个模型，如空间点、体素。尽管这些结构的通用性很好，可以表示任意的结构，但是采用合适的存储方式存储对应的结构显然是一种更优的选择。在这一章中，我们提出了一个新的数据管理方式，这种管理方式使用一个统一的索引数据来管理不同的结构。这是因为在我们的设计当中，索引项和数据项被完全的分离。索引项仅仅存储数据项存储的位置，并不知道存储的结构是什么样的。所有的数据项都存储在与索引项分离的内存空间中。这样设计的一大优点是，我们的方法可以很容易的扩展到任意的结构。

在所有的常见结构中，平面结构不仅仅存在于大多数场景中，而且对于平面结构的采用体素进行存储往往会消耗很多空间。为了提升系统的存储效率，平面结构被引入到我们的系统。在本章中，我们设计了一个实时的三维重建系统。与过去的方法不同，这个系统采用多种结构表示重建的模型。与 VoxelHashing 相同，为了获得更高的存储效率，只有在重建表面附近的体素被存储。在模型中被检测到的平面结构将采用更高效的结构进行存储。所有的数据都由单个散列表进行管理。因为模型中存在的平面会在多帧中被观测到，因而会被重复提取多次。这一问题采用平面散列表（Plane Hash Array，PHA）进行解决。这种解决方案除了散列表地址的计算外没有额外的计算量，因而计算非常的高效。更为重要的是，所有提取了的平面均存储在平面散列表中，所以也没有额外的空间损失。为了更高的运行效率，针对三维重建模型，我们提出了一种全新的高效的并行平面提取方法。这个算法是完全并行的，可以充分利用 GPU 的所有计算资源。我们将在场景中提取平面的操作分布在连续帧中，而不是集中在在一帧中进行提取。因此，我们的算法可以以很高的帧率运行。

整个系统的流程与 InfiniTAM 类似，如图5.1所示。系统分为跟踪、建图、渲染和平面简化四大部分。跟踪过程是为了实时估计相机的位姿，与 InfiniTAM 类似，跟踪过程同样是在深度图和彩色图上进行的。对于深度图像，采用点到平面

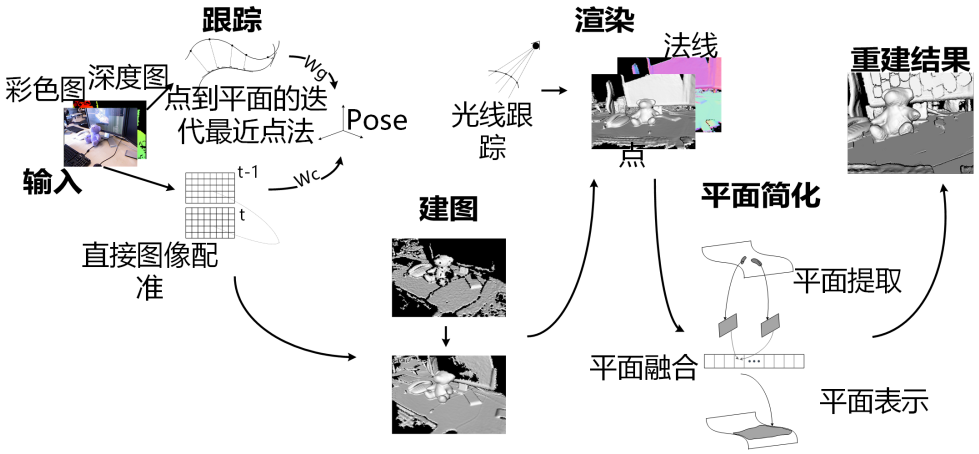

图 5.1 算法流程

的迭代最近点算法。对于彩色图，采用帧到帧的稠密配准。两个估计的结果通过权重 w_c 和 w_g 相结合。建图部分先检测存在于当前帧但未分配的体素块，并对其在内存中进行分配，其次对整个场景中的体素块进行更新。渲染过程采用光线跟踪算法。在平面检测当中，我们从重建好的 3D 模型中提取场景中存在的平面，之后插入新的平面、融合新的测量值，最后获得重建好的 3D 模型。

第一节 数据管理

在我们的系统中，我们对于重建模型采用了多结构表示方式（体素块和平面）。存储内容分为索引和数据。索引和数据在我们的系统中完全分离。索引项存储对应数据项的存储位置，而不清楚数据项内容的具体形式。在我们的系统中，采用散列表作为索引。与前几章的方法类似，体素被聚合成体素块进行存储，并且只有重建表面附近的体素块被存储。除了体素块，我们的系统中还存储了平面结构。对于平面结构，对应在相同平面上的散列项将指向相同平面。体素块被存储在体素块列表中（Voxel Block Array，VBA）。与之相对，所有平面存储在平面散列表当中（Plane Hash Arrray，PHA）。由于以上设计，我们的存储效率处于较高水平。数据管理方法如图5.2所示。

第二节 高效的平面提取

在这一节中，我们介绍我们设计的高效的平面提取算法。如果直接从整个模型中提取平面，计算负担将非常高高，很难满足算法实时性的需求。因而我

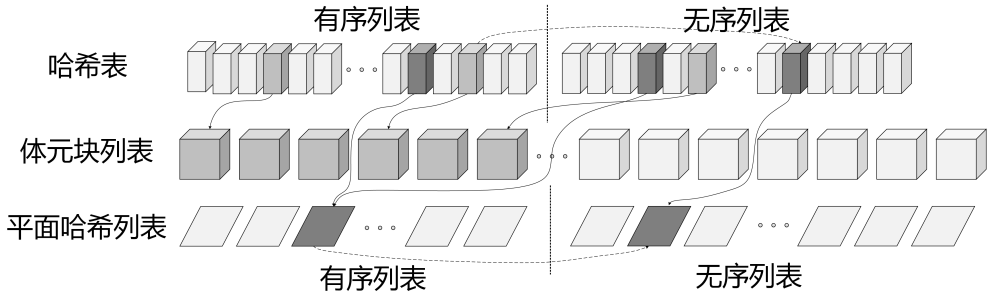

图 5.2 数据管理示意图

们从由三维模型渲染生成的点云图中提取平面。与来自原始深度图像的点云相比，渲染生成的点云图由于融合了来自不同帧的观测量，因而具有较少的噪声。更为重要的是，从渲染的点云图上提取平面很容易并行化，并在运行效率方面取得良好表现。这是因为由渲染生成的点图被组织为结构化的 2D 图像，很容易进行划分，进行并行化计算。

我们的平面提取算法可以被视为一种随机样本一致性（Random sample consistency，RANSAC）方法[71] 的变种。随机样本一致性算法使用随机抽样观察到的数据，采用投票方案进行估计模型的参数。因为实时三维重建算法通常运行在强大的并行设备上，为了提高运行效率，我们的算法也设计为完全平行。众所周知，一个平面可以表示为：

$$n_x(x-x_0)+n_z(z-z_0)+n_z(z-z_0)=0 \tag{5.1}$$

其中 (x,y,z) 表示平面上的任意点，(x_0,y_0,z_0) 是平面上的给定点，(n_x,n_y,n_z) 表示平面的法线方向。

在传统的随机样本一致性方法中，从所有点中随机选择三个点用于拟合模型，剩余的点用于对此模型投票。因为点云图被组织为结构化的 2D 图像，整个图像被均匀的分成小块，每小块都包含 16×16 个点。在大多数情况下，如果一个点在一个平面上，附近的点也极有可能可能在同一平面上，而这一规律不适用于距离较远的点。因此在每次迭代中，采用一个图像小块中的点进行模型的拟合和投票，而不是整点云。通过这种设计，每个线程的计算负担都会大大减少。更重要的是，各个图像小块上的提取是完全平行的。因而我们的设计方式有利于提高算法的运行效率效率。

由于相邻帧之间的相机移动非常微小，因而连续帧之间的内容通常只存在很小的差异。换句话说，每个图像小块中的内容会随着时间的推移逐渐变化，相

似的内容会多次出现在其他图像小块中。根据上面的讨论，不需要在一个图像小块上迭代多次。在我们的算法中，当一个的新帧到来时，每个图像小块中只会拟合一个模型。当相机移动时，相同的空间平面将会在不同的图像小块中进行拟合。因而在每一帧进行平面提取的计算负担进一步减少。每个图像小块使用固定初始模式而不是三个随机点来拟合最初的模型。因为同一平面通常可以在几个连续的帧中被观测到，因而同一平面将被提取多次。这个问题将在接下来的章节中进行处理。从实验结果来看，我们可以看到我们的平面提取算法对整个系统的运行效率影响很小。该算法也适用于其他三维重建算法。整个平面提取算法的流程如图5.3所示。

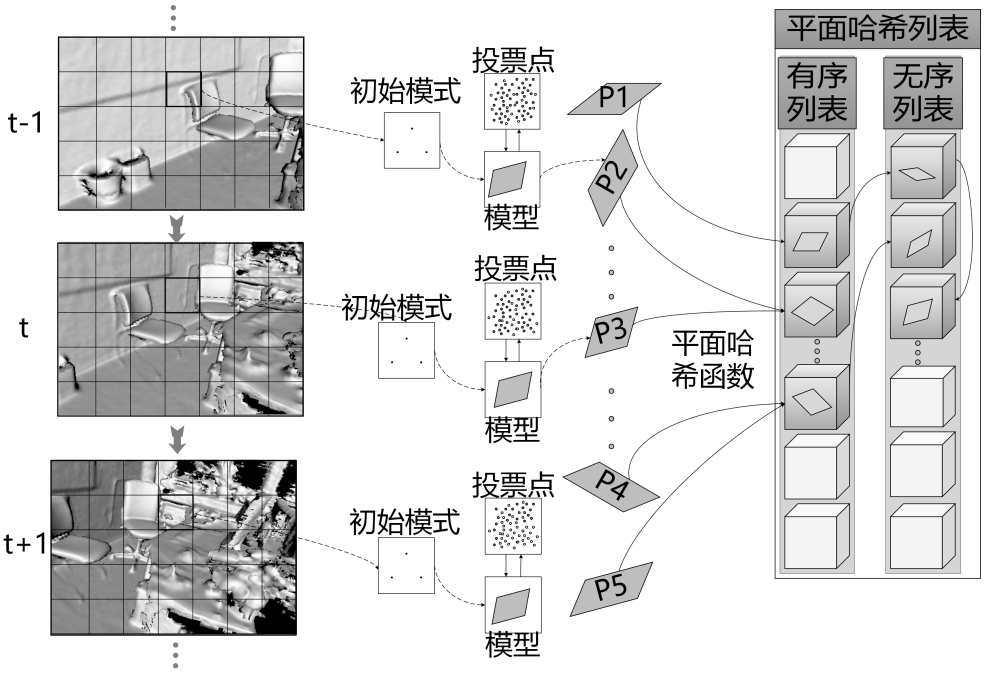

图 5.3　平面提取示意图

第三节　平面融合

因为同一平面会被观察多次，因而会被反复提取。为了解决这个问题，我们采用了平面散列表。在我们的系统中，所有被提取的平面存储在平面散列表中。平面通过散列函数计算得到存储位置。同样的平面将落在平面散列表中的相同位置。通过这种设计，除了散列表位置操作之外没有额外的成本。

5.3.1 规范化平面表示

众所周知，一个平面可以表示为两个三维向量：一个落在平面上的点和该平面的法线。但是这种表示方法很难评估两个平面之间的差异程度。由于在平面上的空间点的选取是任意的，因而两个平面之间的差异并不固定。为此，我们基于该表示方法，设计了一种新的规范化平面表示方法。在我们的设计中，标准化的平面表示仅需更少的参数，与此同时，还易于评估的两个平面之间的差异。在规范化平面表示中，平面仅仅表示为一个三维向量，这个向量的方向与平面法向量方向保持一致，长度等于该平面到世界坐标系下原点的距离。换言之，该平面由平面上距离世界原点最近的点表示。标准化平面表示如图5.4所示。

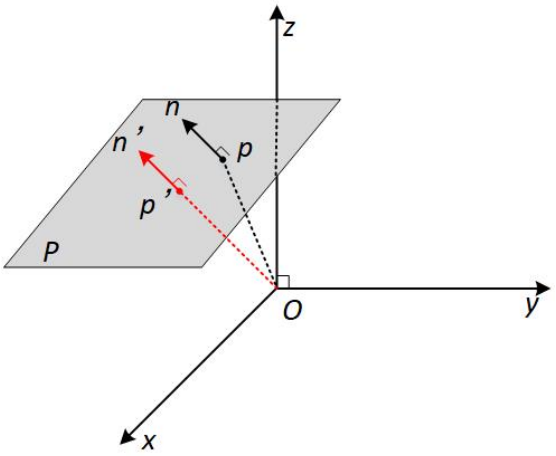

图 5.4 规范化平面表示

5.3.2 平面插入

当一个新的平面被提取出来时，它将被插入到平面散列表中去。首先，平面散列表分为有序列表和无序列表。插入的平面将首先使用散列表函数找到它在平面散列表中的有序表的存储位置。当此位置被其他平面占据时，该平面将被存储到平面散列表中无序表中的第一个为空的位置，并使用偏移项来将两个位置连接起来。无序列表当中的偏置项可以链接到更多后面在无序列表中的插入。如果检测到插入平面存在在平面散列表中，就执行平面更新过程。详细信息将在下一节中介绍。由于归一化平面由一个三维向量表示，因而我们采用二范数衡量平面间的差异。由于存在噪声，从模型中中提取出来的相同的平面结构将在很小的范围内变化。为了减少这种影响，我们将归一化平面所落入的体

素位置，用于散列函数的计算。散列函数数可以写成:

$$h = ((p_x \cdot H_x) \oplus (p_y \cdot H_y) \oplus (p_z \cdot H_z)) \mod L \tag{5.2}$$

其中 (p_x, p_y, p_z) 表示标准化平面落入的体素的位置。H_1, H_2, H_3 是散列表函数的参数。L 为平面散列表中有序列表的长度。在我们的实现中，平面散列表的长度设置为 1024。有序列表和无序列表具有相同的长度。索引散列表也采用了相同的设计方式，和平面散列表唯一的不同之处在于参数选择上的不同。

5.3.3 平面更新

按照上面的设计，对于同一个平面的不同次测量将落入平面散列表中相同的位置。在平面插入过程中，当检测插入平面插入到已有项时，就进行平面更新。平面更新过程可视为融合同一平面的不同测量值的过程。我们采用加权融合的方式，融合公式如下：

$$\mathbf{P}_{PHA} = \frac{S_m \cdot \mathbf{P}_m + S_{PHA} \cdot \mathbf{P}_{PHA}}{S_m + S_{PHA}} \tag{5.3}$$

$$S_{PHA} = S_m + S_{PHA} \tag{5.4}$$

其中 P_{PHA} 是存储在平面散列表中的平面的标准化平面表示，$\mathbf{P}_m$ 表示新的平面测量值。S_{PHA} 和 S_m 为权值，是在平面检测中支持他们的点的个数。

5.3.4 平面的光线追踪

平面的光线追踪过程可视为三维空间中平面与直线求交点的过程。平面方程式与式 (5.1) 相同，而直线线方程定义为：

$$l : \begin{cases} x = x_c + dir_x \cdot t \\ y = y_c + dir_y \cdot t \\ z = z_c + dir_z \cdot t \end{cases} \tag{5.5}$$

其中 $\mathbf{p} = (x, y, z)$ 是直线上的任意点，$\mathbf{p}_c = (x_c, y_c, z_c)$ 是线上的给定点（摄像机位置），$\mathbf{dir} = (dir_x, dir_y, dir_z)$ 是直线的方向（射线方向），t 是直线的长度。所以直线和平面的交点为：

$$\mathbf{p} = \mathbf{p}_c + t \cdot \mathbf{dir} \tag{5.6}$$

$$t = \frac{n_x(x_0 - x_c) + n_y(y_0 - y_c) + n_z(z_0 - z_c)}{n_x \cdot d_x + n_y \cdot d_y + n_z \cdot d_z} \tag{5.7}$$

第四节 平面替换

在真实中通常存在大量的平面结构。如果每个平面都用体素块进行存储，通常会耗费很多的空间。如前所述，平面的解析形式是非常简单的，仅需三个参数。如果是模型中的平面结构都用对应的解析形式进行表示，储存效率将取得极大的提升。

在插入新的提取平面后，对渲染出来的点云图进行共面性检查。对于每个点，执行共面检查。对于每一个点，我们检测其是否落在提取出来的平面上。为了精确性，共面性检查不仅考虑了点到平面的距离，同时还检验了该点的法向量与平面的法向量的一致性。因而法向量图也参与了平面检测的过程。如果一个点有较高的可能性落在平面上，则标记其落在该平面上。因为体素被组织成了体素块，所以只有体素块中包含足够多的体素落在该平面上，该体素才被标记为在此平面上。通过这种设计，我们保证了在边界处的体素块不被移除。当一个体素块被标记在一个平面上时，它将从体素块列表中释放，并且索引散列表中对应的索引项将指向对应的平面。

第五节 实验结果

本节中的实验设定与数据集与上一章中的实验部分一致。所有算法都在相同的硬件平台上进行评估，平台采用 Intel i7- 4790K (4GHz) CPU 和 NVIDIAGTX 1080 GPU。除了上一章的方法以外，我们还加入了面元结构的重建（Surfel）。

5.5.1 TUM RGBD 数据集

TUM RGB-D 数据集包含多个序列，均是深度传感器（Kinect）在不同场景中获得的。每个序列包含 RGB 图像、深度图像和加速器数据。在这个实验中，一个体素块的存储空间是一个平面存储的空间的 100 倍。该结果如表5.1所示。我们可以看到被替换的体素块的数量非常大 (如图5.5)。如果重建尺度变大或在场景中存在更多的平面结构，被替换的体素块的数量也将是增加。

表 5.1 在 TUM RGBD 数据集上节省的体素块数量

序列	fr2_desk	fr2_fb	fr3_cabinet	fr3_str_notex_ne	fr3_str_tex_ne
节省数量	532	698	1078	1245	715

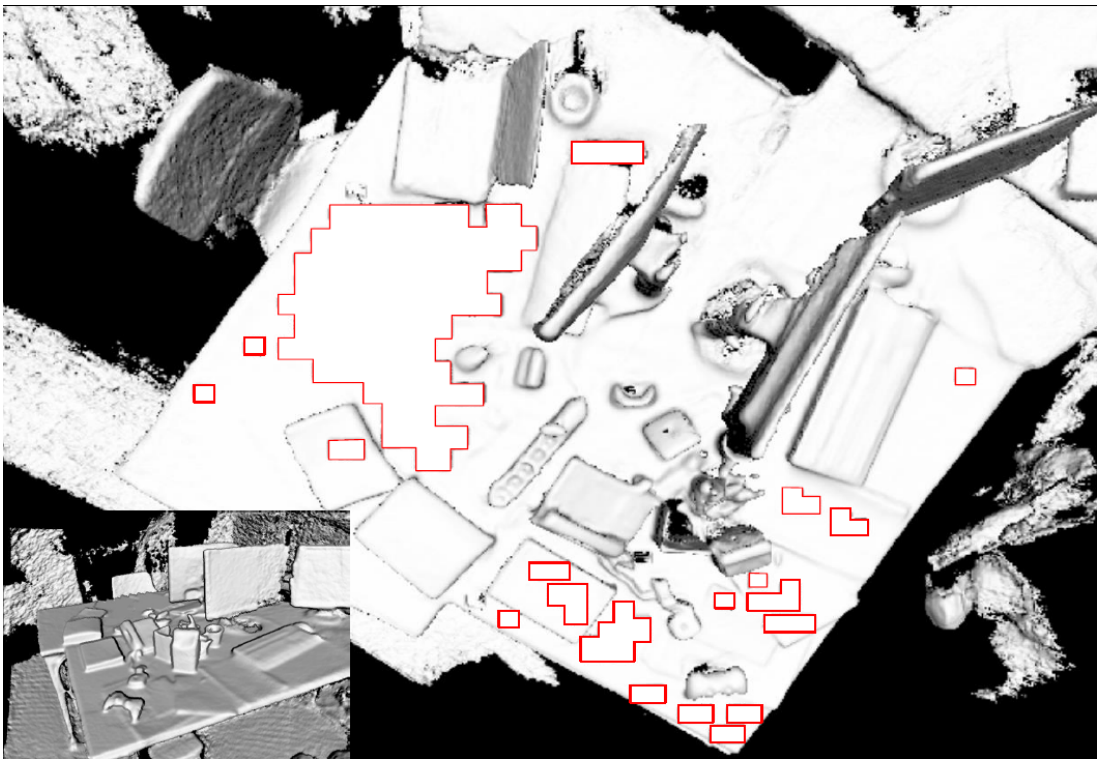

图 5.5 被替换的体素块

5.5.2 ScanNet 数据集

表 5.2 在 ScanNet 数据集上的每帧平均运行时间 (毫秒)

Scene	07_00	13_00	17_00	39_00	42_00	47_00	49_00	平均
VH	2.88ms	2.96ms	3.25ms	2.94ms	2.87ms	3.17ms	3.26ms	3.00ms
Ours	3.01ms	3.1ms	3.53ms	2.95ms	3.01ms	3.37ms	3.41ms	3.17ms
Dense	13.82ms	11.81ms	12.45ms	-	11.89ms	-	11.84ms	12.36ms
Surfel	21.3ms	17.62ms	20.01ms	19.4ms	18.17ms	23.89ms	26.5ms	20.9ms

ScanNet 数据集与 TUM RGB-D 数据集类似，由从不同场景收集到的序列所组成。每个序列的组成数据均按 TUM RGB-D 数据集提供，即每个序列包含 RGB 图像、深度图像和加速器数据。为了公平的进行比较，每个序列运行五次，取平均运行时间为最终的每帧的平均运行时间。实验结果显示在表5.2中。从表中我们可以看到我们的方法比 VoxelHashing 略慢（小于 0.2ms）。注意额外的时间不仅包含平面提取，还包含平面融合和平面更换。因此我们的方法取得了非

常好效率。更重要的是，我们的方法要比其他方法更快。另外两种方法，受重建规模的限制，在一些序列上重建失败并且每帧仅仅获得 12.36ms 的每帧的平均运行时间。采用面元的方法在此指标上实现最差的性能。

第六章 总结展望

在本文中，针对三维重建系统中存在的问题：位姿估计不够准确与鲁棒、存储效率过低和对于几何结构利用的不完善分别提出了对应的解决方案。构建了一个结构指导的实时鲁棒高效三维重建系统。

对于位姿估计模块，现有的基于点特征的方法在少纹理场景当中（如墙壁）经常失败。因为直线结构广泛存在于自然场景当中。我们将直线结构融入位姿估计模块。近年来，融合直线结构的方法在位姿估计中得到了广泛的应用。虽然位姿估计的精度和鲁棒性有了一定的提高，但是这些方法大大增加了计算量。这是因为这些方法将直线结构看作独立于点特征的另一种特征，进行了复杂的计算。我们与这些方法不同，将直线结构作为指导信息融入位姿估计模块，而不是额外的特征。因而我们的方法在不损失运行效率的前提下提升了位姿估计的精度与鲁棒性。

对于存储效率。现有的三维重建算法的数据管理形式通常可以分为顺序结构和散列结构。顺序结构，例如 KinectFusion 按空间位置存储数据，因此不需要存储额外的索引数据。然而，这种方法需要存储整个空间中的所有体素数据。散列结构仅仅存储重建表面附近的体素，因而避免了存储大量的空体素。然而由于体素数据在内存中的排列是与其空间位置无关的，所以如何对数据进行管理就成为了一个难题，如索引、插入、删除。为此，一个设计良好的散列表被引入用来管理存储的数据。我们在这些工作的基础之上设计了一种半顺序结构来提升系统的存储效率。这种结构既能够只存储非空的体素，又能够减少所需的索引数据。

我们可以看到大部分工作只使用了单一结构来表示整个模型。然而在现实世界中，通常存在许多几何结构，如线结构、平面结构等等。如果这些结构能够被利用，则三维重建算法的性能就会得到提升。由于在所有的常见结构中，平面结构不仅仅存在于大多数场景中，而且对于平面结构的采用体素进行存储往往会消耗很多空间。为了提升系统的存储效率，我们将平面结构引入到我们的模型表达中来。

我们在许多数据集上测试了我们的方法，发现我们的方法均优于传统方法。

这证明了我们设计的有效性。

对于位姿估计，我们接下来工作的重点是引入额外的传感器信息使其更加鲁棒、准确。

实验证明我们提出的两个提升存储效率的方法取得了很好的效果。然而我们所提出的两个方案相对独立。因而我们考虑在是否有将两个方案相整合，以进一步提升存储效率的可能性。

对于结构化跳表，我们通过引用结构信息提升了系统的存储效率，然而这些信息并未进一步的得到利用。如利用结构化信息对数据在 CPU 与 GPU 之间的交换进行简化是接下来工作的重点。

对于平面结构的利用我们仅停留在最初始的状态，并未更进一步。在接下来的工作当中，我们考虑将平面结构引入整个系统中更多的部分，如位姿跟踪过程。除此之外，对于平面边缘部分的对齐问题，我们没有很好的解决。这些问题将是接下来工作的重中之重。

参考文献

[1] DURRANT-WHYTE H, BAILEY T. Simultaneous localization and mapping: part I. [J]. IEEE robotics & automation magazine, 2006, 13 (2): 99–110.

[2] BAILEY T, DURRANT-WHYTE H. Simultaneous localization and mapping (SLAM): Part II. [J]. IEEE Robotics & Automation Magazine, 2006, 13 (3): 108–117.

[3] CADENA C, CARLONE L, CARRILLO H, et al. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. [J]. IEEE Transactions on robotics, 2016, 32 (6): 1309–1332.

[4] FRAUNDORFER F, SCARAMUZZA D. Visual odometry: Part ii: Matching, robustness, optimization, and applications. [J]. IEEE Robotics & Automation Magazine, 2012, 19 (2): 78–90.

[5] SCARAMUZZA D, FRAUNDORFER F. Visual odometry [tutorial]. [J]. IEEE robotics & automation magazine, 2011, 18 (4): 80–92.

[6] FORSTER C, PIZZOLI M, SCARAMUZZA D. SVO: Fast semi-direct monocular visual odometry. [C] // 2014 IEEE international conference on robotics and automation (ICRA). IEEE. 2014: 15–22.

[7] 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.

[8] NISTÉR D, NARODITSKY O, BERGEN J. Visual odometry for ground vehicle applications. [J]. Journal of Field Robotics, 2006, 23 (1): 3–20.

[9] HANNA M G, AHMED I, NINE J, et al. Augmented reality technology using Microsoft HoloLens in anatomic pathology. [J]. Archives of pathology & laboratory medicine, 2018, 142 (5): 638–644.

[10] DIOSI A, KLEEMAN L. Advanced sonar and laser range finder fusion for simultaneous localization and mapping. [C] // 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566). Vol. 2. IEEE. 2004: 1854–1859.

[11] ZHANG J, SINGH S. LOAM: Lidar Odometry and Mapping in Real-time. [C] // Robotics: Science and Systems. Vol. 2. 2014: 9.

[12] ZHANG J, SINGH S. Visual-lidar odometry and mapping: Low-drift, robust, and fast. [C] // 2015 IEEE International Conference on Robotics and Automation (ICRA). IEEE. 2015: 2174–2181.

[13] ENGEL J, KOLTUN V, CREMERS D. Direct sparse odometry. [J]. IEEE transactions on pattern analysis and machine intelligence, 2018, 40 (3): 611–625.

[14] MUR-ARTAL R, MONTIEL J M M, TARDOS J D. ORB-SLAM: a versatile and accurate monocular SLAM system. [J]. IEEE transactions on robotics, 2015, 31 (5): 1147–1163.

[15] WANG R, SCHWORER M, CREMERS D. Stereo dso: Large-scale direct sparse visual odometry with stereo cameras. [C] // Proceedings of the IEEE International Conference on Computer Vision. 2017: 3903–3911.

[16] AQEL M O, MARHABAN M H, SARIPAN M I, et al. Review of visual odometry: types, approaches, challenges, and applications. [J]. SpringerPlus, 2016, 5 (1): 1897.

[17] NEWCOMBE R A, IZADI S, HILLIGES O, et al. KinectFusion: Real-time dense surface mapping and tracking. [C] // 2011 IEEE International Symposium on Mixed and Augmented Reality. IEEE. 2011: 127–136.

[18] IZADI S, KIM D, HILLIGES O, et al. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. [C] // Proceedings of the 24th annual ACM symposium on User interface software and technology. ACM. 2011: 559–568.

[19] DAI A, CHANG A X, SAVVA M, et al. Scannet: Richly-annotated 3d reconstructions of indoor scenes. [C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017: 5828–5839.

[20] BERGER M, TAGLIASACCHI A, SEVERSKY L M, et al. A survey of surface reconstruction from point clouds. [C] // Computer Graphics Forum. Vol. 36. 1. Wiley Online Library. 2017: 301–329.

[21] CHEN K, LAI Y.-K, HU S.-M. 3D indoor scene modeling from RGB-D data: a survey. [J]. Computational Visual Media, 2015, 1 (4): 267–278.

[22] HITOMI E E, SILVA J V, RUPPERT G C. 3D scanning using RGBD imaging devices: A survey. [G] // Developments in Medical Image Processing and Computational Vision. Springer, 2015: 379–395.

[23] CURLESS B, LEVOY M. A volumetric method for building complex models from range images. [J]. 1996.

[24] HILTON A, STODDART A J, ILLINGWORTH J, et al. Reliable surface reconstruction from multiple range images. [C] // European conference on computer vision. Springer. 1996: 117–126.

[25] NEWCOMBE R A, LOVEGROVE S J, DAVISON A J. DTAM: Dense tracking and mapping in real-time. [C] // 2011 international conference on computer vision. IEEE. 2011: 2320–2327.

[26] PRADEEP V, RHEMANN C, IZADI S, et al. MonoFusion: Real-time 3D reconstruction of small scenes with a single web camera. [C] // 2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR). IEEE. 2013: 83–88.

[27] WHELAN T, KAESS M, FALLON M, et al. Kintinuous: Spatially extended kinectfusion. [J]. 2012.

[28] WHELAN T, JOHANNSSON H, KAESS M, et al. Robust real-time visual odometry for dense RGB-D mapping. [J]. 2013.

[29] WHELAN T, KAESS M, JOHANNSSON H, et al. Real-time large-scale dense RGB-D SLAM with volumetric fusion. [J]. The International Journal of Robotics Research, 2015, 34 (4-5): 598–626.

[30] WHELAN T, KAESS M, LEONARD J J, et al. Deformation-based loop closure for large scale dense RGB-D SLAM. [C] // 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE. 2013: 548–555.

[31] STEINBRÜCKER F, STURM J, CREMERS D. Volumetric 3D mapping in real-time on a CPU. [C] // 2014 IEEE International Conference on Robotics and Automation (ICRA). IEEE. 2014: 2021–2028.

[32] ZENG M, ZHAO F, ZHENG J, et al. Octree-based fusion for realtime 3D reconstruction. [J]. Graphical Models, 2013, 75 (3): 126–136.

[33] NIEßNER M, ZOLLHÖFER M, IZADI S, et al. Real-time 3D reconstruction at scale using voxel hashing. [J]. ACM Transactions on Graphics (ToG), 2013, 32 (6): 169.

[34] DAI A, NIEßNER M, ZOLLHÖFER M, et al. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. [J]. ACM Transactions on Graphics (ToG), 2017, 36 (4): 76a.

[35] GOMEZ-OJEDA R, ZUÑIGA-NOËL D, MORENO F.-A, et al. Pl-slam: a stereo slam system through the combination of points and line segments. [J]. ArXiv preprint arXiv:1705.09479, 2017.

[36] PUMAROLA A, VAKHITOV A, AGUDO A, et al. PL-SLAM: Real-time monocular visual SLAM with points and lines. [C] // 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE. 2017: 4503–4508.

[37] ZUO X, XIE X, LIU Y, et al. Robust visual SLAM with point and line features. [C] // 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE. 2017: 1775–1782.

[38] YANG S, SCHERER S. Direct monocular odometry using points and lines. [C] // 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE. 2017: 3871–3877.

[39] LI S.-J, REN B, LIU Y, et al. Direct Line Guidance Odometry. [C] // 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE. 2018: 1–7.

[40] CHOI C, TREVOR A J, CHRISTENSEN H I. RGB-D edge detection and edge-based registration. [C] // 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE. 2013: 1568–1575.

[41] LU Y, SONG D. Robust rgb-d odometry using point and line features. [C] // Proceedings of the IEEE International Conference on Computer Vision. 2015: 3934–3942.

[42] WHELAN T, MA L, BONDAREV E, et al. Incremental and batch planar simplification of dense point cloud maps. [J]. Robotics and Autonomous Systems, 2015, 69: 3–14.

[43] MA L, WHELAN T, BONDAREV E, et al. Planar simplification and texturing of dense point cloud maps. [C] // 2013 European Conference on Mobile Robots. IEEE. 2013: 164–171.

[44] DZITSIUK M, STURM J, MAIER R, et al. De-noising, stabilizing and completing 3D reconstructions on-the-go using plane priors. [C] // 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE. 2017: 3976–3983.

[45] MA L, KERL C, STÜCKLER J, et al. CPA-SLAM: Consistent plane-model alignment for direct RGB-D SLAM. [C] // 2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE. 2016: 1285–1291.

[46] KAESS M. Simultaneous localization and mapping with infinite planes. [C] // 2015 IEEE International Conference on Robotics and Automation (ICRA). IEEE. 2015: 4605–4611.

[47] SALAS-MORENO R F, GLOCKEN B, KELLY P H, et al. Dense planar SLAM. [C] // 2014 IEEE international symposium on mixed and augmented reality (ISMAR). IEEE. 2014: 157–164.

[48] HSIAO M, WESTMAN E, ZHANG G, et al. Keyframe-based dense planar SLAM. [C] // 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE. 2017: 5110–5117.

[49] MA L, FAVIER R, DO L, et al. Plane segmentation and decimation of point clouds for 3d environment reconstruction. [C] // 2013 IEEE 10th Consumer Communications and Networking Conference (CCNC). IEEE. 2013: 43–49.

[50] YANG S, SONG Y, KAESS M, et al. Pop-up slam: Semantic monocular plane slam for low-texture environments. [C] // 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE. 2016: 1222–1229.

[51] NEWCOMBE R A, FOX D, SEITZ S M. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. [C] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 343–352.

[52] ZOLLHÖFER M, NIEßNER M, IZADI S, et al. Real-time non-rigid reconstruction using an RGB-D camera. [J]. ACM Transactions on Graphics (ToG), 2014, 33 (4): 156.

[53] INNMANN M, ZOLLHÖFER M, NIEßNER M, et al. VolumeDeform: Real-time volumetric non-rigid reconstruction. [C] // European Conference on Computer Vision. Springer. 2016: 362–379.

[54] PURI P, JIA D, KAESS M. GravityFusion: Real-time dense mapping without pose graph using deformation and orientation. [C] // 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE. 2017: 6506–6513.

[55] LAIDLOW T, BLOESCH M, LI W, et al. Dense RGB-D-inertial SLAM with map deformations. [C] // 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE. 2017: 6741–6748.

[56] WHELAN T, LEUTENEGGER S, SALAS-MORENO R, et al. ElasticFusion: Dense SLAM without a pose graph. [C] //. Robotics: Science, Systems. 2015.

[57] WHELAN T, SALAS-MORENO R F, GLOCKER B, et al. ElasticFusion: Real-time dense SLAM and light source estimation. [J]. The International Journal of Robotics Research, 2016, 35 (14): 1697–1716.

[58] CHANG A, DAI A, FUNKHOUSER T, et al. Matterport3d: Learning from rgb-d data in indoor environments. [J]. ArXiv preprint arXiv:1709.06158, 2017.

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

[60] MCCORMAC J, HANDA A, DAVISON A, et al. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. [C] // 2017 IEEE International Conference on Robotics and automation (ICRA). IEEE. 2017: 4628–4635.

[61] KOSTAVELIS I, GASTERATOS A. Semantic mapping for mobile robotics tasks: A survey. [J]. Robotics and Autonomous Systems, 2015, 66: 86–103.

[62] SÜNDERHAUF N, PHAM T T, LATIF Y, et al. Meaningful maps with object-oriented semantic mapping. [C] // 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE. 2017: 5079–5085.

[63] XIANG Y, FOX D. DA-RNN: Semantic mapping with data associated recurrent neural networks. [J]. ArXiv preprint arXiv:1703.03098, 2017.

[64] MCCORMAC J, CLARK R, BLOESCH M, et al. Fusion++: Volumetric object-level slam. [C] // 2018 International Conference on 3D Vision (3DV). IEEE. 2018: 32–41.

[65] YANG S, SCHERER S. CubeSLAM: Monocular 3d object detection and slam without prior models. [J]. ArXiv preprint arXiv:1806.00557, 2018.

[66] SALAS-MORENO R F, NEWCOMBE R A, STRASDAT H, et al. Slam++: Simultaneous localisation and mapping at the level of objects. [C] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2013: 1352–1359.

[67] XU B, LI W, TZOUMANIKAS D, et al. MID-Fusion: Octree-based Object-Level Multi-Instance Dynamic SLAM. [J]. ArXiv preprint arXiv:1812.07976, 2018.

[68] PRISACARIU V A, KÄHLER O, GOLODETZ S, et al. InfiniTAM v3: a framework for large-scale 3D reconstruction with loop closure. [J]. ArXiv preprint arXiv:1708.00783, 2017.

[69] 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.

[70] KÄHLER O, PRISACARIU V A, MURRAY D W. Real-time large-scale dense 3D reconstruction with loop closure. [C] // European Conference on Computer Vision. Springer. 2016: 500–516.

[71] HARTLEY R, ZISSERMAN A. Multiple view geometry in computer vision. [M]. Cambridge university press, 2003.

[72] BESL P J, MCKAY N D. Method for registration of 3-D shapes. [C] // Sensor Fusion IV: Control Paradigms and Data Structures. Vol. 1611. International Society for Optics, Photonics. 1992: 586–607.

[73] RUSINKIEWICZ S, LEVOY M. Efficient variants of the icp algorithm. [C] // 3dim. Vol. 1. 2001: 145–152.

[74] TOMASI C, MANDUCHI R. Bilateral filtering for gray and color images. [C] // Iccv. Vol. 98. 1. 1998: 2.

[75] ENGEL J, SCHÖPS T, CREMERS D. LSD-SLAM: Large-scale direct monocular SLAM. [C] // European conference on computer vision. Springer. 2014: 834–849.

[76] SIBLEY G, MATTHIES L, SUKHATME G. A sliding window filter for incremental SLAM. [G] // Unifying perspectives in computational and robot vision. Springer, 2008: 103–112.

[77] CIVERA J, DAVISON A J, MONTIEL J M. Inverse depth parametrization for monocular SLAM. [J]. IEEE transactions on robotics, 2008, 24 (5): 932–945.

[78] STRASDAT H, MONTIEL J, DAVISON A J. Real-time monocular SLAM: Why filter? [C] // 2010 IEEE International Conference on Robotics and Automation. IEEE. 2010: 2657–2664.

[79] VON GIOI R G, JAKUBOWICZ J, MOREL J.-M, et al. LSD: A fast line segment detector with a false detection control. [J]. IEEE transactions on pattern analysis and machine intelligence, 2010, 32 (4): 722–732.

[80] VON GIOI R G, JAKUBOWICZ J, MOREL J.-M, et al. LSD: a line segment detector. [J]. Image Processing On Line, 2012, 2: 35–55.

[81] PIRE T, FISCHER T, CIVERA J, et al. Stereo parallel tracking and mapping for robot localization. [C] // 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE. 2015: 1373–1378.

[82] GEIGER A, ZIEGLER J, STILLER C. Stereoscan: Dense 3d reconstruction in real-time. [C] // 2011 IEEE Intelligent Vehicles Symposium (IV). IEEE. 2011: 963–968.

[83] 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.

[84] KÄHLER O, PRISACARIU V, VALENTIN J, et al. Hierarchical voxel block hashing for efficient integration of depth images. [J]. IEEE Robotics and Automation Letters, 2016, 1 (1): 192–197.

致谢

时光如梭，三年的研究生生涯如白驹过隙，一晃眼已是尽头。在这三年多来的学习生涯当中，我成长了许多。不仅有学业上的，也有生活当中的。在即将毕业之际，回首在南开这三年来的日日夜夜，唯有感谢。

在这三年来的研究生生涯中，我最为感谢的就是我的导师程明明教授。因为本科所学并不是计算机专业，初入南开时对于计算机专业的相关知识只能说是一知半解。幸而老师不厌其烦，为我指点迷津，悉心教导我，使我快速的入门。当毕业之际，我已经对于相关的领域知识了然于胸，并且发表了一些相关的论文。这其中的一大部分功劳都要归功于我的导师。除了学术上的指点，为了我的更好发展，程老师还会资助我们出国开会交流，了解相关领域的最新发展，使得我们拥有了更为广阔的学术视野。当我想要去慕尼黑工业大学访学交流之际，也是程老师第一个表示支持，并主动向对方教授推荐了我。所以在这里我发自内心的要对程老师表示我最真挚的谢意。

其次，我要对实验室的各位同学表示感谢。在这三年来的实验室生活当中，大家互帮互助，共同解决了我很多学术上的问题。尤其是我同届的博士生刘云、侯淇滨和硕士生刘笑畅、胡晓伟和王亚慧。正是有了你们的热情帮助，我的研究生生涯才能如此顺利的完成。

再者，我要感谢我的家人。虽然不在同一个地方，很少团聚。但是你们仍旧给了我充分的理解与支持，在我气馁时给予我鼓励，心情不好时听我倾诉。有了你们的陪伴与支持才使我在遇到任何困难时都有迎难而上的勇气。

接下来，真诚的感谢各位评审组老师能够在百忙之中抽出宝贵的时间对我的学位论文进行审阅。同时也感谢这三年以来所有任课老师对于我的信息教导。我从课堂上受益良多。

最后的最后，感谢南开大学对于我的培养。

个人简历

李仕杰，出生于 1993 年 08 月 21 日。在 2016 年毕业于电子科技大学自动化专业并获得工学学士学位。于 2016 年至今在南开大学就读计算机科学与技术专业研究生。

研究生期间发表论文

1. **Li, Shi-Jie**, Bo Ren, Yun Liu, Ming-Ming Cheng, Duncan Frost, and Victor Adrian Prisacariu. ”Direct Line Guidance Odometry.” In 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 1-7. IEEE, 2018.(自动化学会 **A** 类)
2. **Li, Shi-Jie**, Ming-Ming Cheng, Yun Liu, Shao-Ping Lu, YaHui Wang, and Victor Adrian Prisacariu. ”Structured Skip List: A Compact Data Structure for 3D Reconstruction.” In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1-7. IEEE, 2018.(自动化学会 **A** 类)
3. Liu, Yun, Peng-Tao Jiang, Vahan Petrosyan, **Shi-Jie Li**, Jiawang Bian, Le Zhang, and Ming-Ming Cheng. ”DEL: Deep Embedding Learning for Efficient Image Segmentation.” In IJCAI, vol. 864, p. 870. 2018.(**CCF A** 类)
4. Wang, Yahui, Shaojun Cai, **Shi-Jie Li**, Yun Liu, Yangyan Guo, Tao Li, and Ming-Ming Cheng. ”CubemapSLAM: A Piecewise-Pinhole Monocular Fisheye SLAM System.” ACCV (2018). (**CCF C** 类)
5. Jiang, Huaizu, Ming-Ming Cheng, **Shi-Jie Li**, Ali Borji, and Jingdong Wang. ”Joint salient object detection and existence prediction.” Frontiers of Computer Science (2017): 1-11.(**CCF C** 类)

所获荣誉

- 研究生一等奖学金, 2018.