专 业 学 位 硕 士 学 位 论 文

基于 RGB-D 相机的 SLAM 技术研究

Study on Technology of SLAM Based on RGB-D Camera

作 者 姓 名：李 彤

工 程 领 域：电子与通信工程

学　　号：31709068

指 导 教 师：殷福亮 教授

完 成 日 期：2019 年 5 月 18 日

大连理工大学

Dalian University of Technology

摘　要

随着科技的发展，服务机器人慢慢进入了人们的生活，变成人们生活中很重要的一部分，目前各个领域对机器人的要求都包括实现机器人的自主定位和路径规划，要实现这些功能，首先要完成机器人的同时定位与地图构建，即 SLAM (Simultaneous Localization and Mapping)。SLAM 系统能够在机器人所处环境未知的情况下，通过传感器获得的环境信息，完成机器人的自主定位与环境地图构建。

本文研究了基于 RGB-D 相机的视觉 SLAM 系统，主要研究工作如下。

(1)介绍了经典的视觉 SLAM 框架，然后详细介绍了基于 KINECT 相机的 SLAM 系统的整体流程，研究了 KINECT 相机的结构、原理和成像模型，并对它进行标定，得到了相机的内参和外参。

(2)利用特征点法实现了 SLAM 系统的前端。图像特征的选择为 ORB(OrientedFAST andRotatedBRIEF)特征，在为 ORB 特征添加了尺度不变性和旋转不变性后，使用汉明距离作为度量进行特征匹配，针对在匹配结果中出现的误匹配，通过设置距离阈值和利用次优匹配来剔除，实验结果表明利用上述方法剔除误匹配之后可以得到正确的特征匹配结果。之后利用 EPnP(Efficient Perspectiven-Point)的方法估计相机的位姿，然后用图优化的方法对估计的位姿进行优化。

(3)实现了系统的回环检测、后端优化以及环境地图构建的功能。首先定义关键帧的概念，通过词袋模型的方法选取关键帧来进行回环检测，以消除误差累积产生的定位漂移问题，在回环之后进行全局位姿优化，然后构建地图。

(4)为了验证本文 SLAM 系统的有效性和实用性，利用数据集和真实环境进行了实验。首先利用 TUM 数据集中的序列进行实验，对全局优化前后系统估计的相机运动轨迹和真实的轨迹分别进行对比，并计算全局优化前后轨迹的误差，实验结果表明，在经过系统后端的全局优化之后，系统的定位精度增强，位姿跟踪效果较好。之后利用数据集以及真实的室内环境进行系统地图构建的实验。结果表明，无论是数据集还是真实环境，系统都能实时构建地图，并且构建的地图可应用于定位和导航，验证了本文 SLAM 系统的有效性和实用性。

关键词：视觉 SLAM; RGB-D; ORB 特征; 回环检测; 图优化

Study on SLAM Based on RGB-D Camera of Mobile Robot

Abstract

With the development of science and technology, mobile robots have gradually entered people's lives and become a very important part. At present, the requirements for robots in various fields include the autonomous positioning and path planning of robots. To achieve these functions, we must first realize the simultaneous localization and mapping of the robot, namely SLAM. The system of SLAM can complete the robot's autonomous positioning and environment map construction through the environmental information obtained by the sensor when the environment of the robot is unknown.

This paper studies the visual SLAM system based on RGB-D camera. The main research work is as follows.

(1) A classic framework of visual SLAM is introduced, then is the complete circuit of the SLAM system based on KINECT. And the structure, principle and imaging model of KINECT is also introduced. The KINECT is calibrated so that the internal and external parameters of the camera are obtained.

(2) The realization of the visual odometer at the front of the SLAM system is based on the method of feature The image features are selected ORB (Oriented FAST and Rotated BRIEF) features. Scale invariance and rotation invariance are added to the ORB feature by image pyramid and gray centroid method. Then, the Hamming distance is used as the metric for feature matching. To eliminate the mismatching in the matching result, the distance threshold is set and the suboptimal matching is used. The experimental results show that the correct feature matching result can be obtained after using the method of eliminating the mismatch above. Then, the method of EPnP (Efficient Perspective-n-Point) is used to estimate the motion of the camera between two adjacent frames, and then the estimated pose is optimized by the method of graph optimization.

(3) The function of loop closing, global optimization and environment map construction of the system is implemented. The concept of key frame is defined first. The key frame is selected based on the bag of words is to perform loop closing detection so that problem of position drift caused by error accumulation can be solved. The global pose optimization is performed using graph optimization after the loop closing, and then the globally consistent trajectories and maps are constructed.

(4) In order to verify the validity and practicability of the system in this paper, experiments were carried out using datasets and real environment. First, the experiments in the TUM dataset

are used to conduct experiments. The trajectories estimated before and after the global optimization are compared with the real trajectories, and the errors of the trajectories are calculated. The experimental results show that the accuracy of positioning is enhanced and the effect of pose tracking is better after the global optimization of the system. The results show that the system can construct maps in real time, whether it is used in dataset or real environment, and the constructed map can be applied to positioning and navigation, which verifies the validity and practicability of the system.

Key Words：Visual SLAM; RDB-D; Feature of ORB; Loop Closing; Graph Optimization

目　　录

1 绪论

1.1 研究背景与意义

随着科技的发展，服务机器人慢慢进入了日常生活中，为人们带来了很多便捷。目前，移动机器人在许多领域都得到了大范围应用。在民用方面，许多公司在无人驾驶汽车方面都取得了不错的成就；医疗方面，有病人移动机器人，看护机器人和手术机器人，手术机器人能够代替医生在手术中进行一些精确度较高的操作；生活方面，在市面上已经有很多扫地机器人和清洁机器人。机器人在人们的生活和工作中扮演着越来越重要的角色，在各个领域发挥着它们不可替代的作用。

机器人在实际移动过程中要思考三个问题：(1)我现在在哪里？(2)我要往哪里走？(3)我该如何到达？解决这些问题涉及到的关键技术分别是导航系统中的定位及其跟踪问题和导航技术中的路径规划问题。要解决这些问题，就需要机器人能够自主定位和对行走过的环境构建地图，这样机器人才能完成导航任务。机器人的自主定位需要依靠传感器获得的环境信息来确定自身的位置坐标。目前很多定位系统都采用 GPS(Global Positioning System)定位技术，但是在一些室内，水下，甚至其他星球上不能使用 GPS，所以需要使用其他的方法进行定位。经过大量的测试实验，研究者们提出了一种机器人同步定位与地图创建的系统框架，也叫作 SLAM [1-2]，该系统框架可以实现将机器人的定位和构图融合在一起，能够使处在未知环境中的机器人对自身所处位置进行定位。定位的实现是利用机器人自身的传感器，机器人通过传感器获得外界的环境信息，对信息进行分析处理可以实现同时定位和创建环境地图。

在 SLAM 系统中，传感器的作用是比较重要的，目前 SLAM 系统中所使用的传感器有激光雷达传感器[3]，视觉传感器[4-6]，还有多传感器融合[7]。激光雷达传感器可以直接得到场景的深度信息，并且能够达到较高的精度，便于构建实时地图，但目前激光传感器的市场价格相对较高，不适用于低成本机器人。视觉传感器就是使用相机作为机器人的传感器，视觉传感器包含单目相机，双目相机和 RGB-D 深度相机。在 SLAM 系统中单目相机指的是使用一个相机作为传感器，其结构简单，成本较低，但它的缺点是不能直接或者间接得到场景的深度信息，也就是物体与机器人之间的距离，所以使用单目相机的 SLAM 系统得到的轨迹在尺度上存在误差，这使单目 SLAM 系统不能广泛应用。与单目相机不同，双目相机为两个并行的相机，作为机器人的“左右眼”，在同一时刻双目相机能够获得场景的两幅不同角度的照片，根据两幅图像的视差可以计算出图像的深度信息，然而双目相机的 SLAM 系统计算量较大，要想获得实时的地图则对处理器的

能力要求较高。RGB-D 相机主要是通过红外结构光或者时间飞行原理，通过发射和接收红外光来测量图像的深度信息[8-11]。它的优势在于不仅能够获得场景的彩色图像，还能够直接得到图像的深度信息。不需要花费时间计算物体的深度，为实时 SLAM 提供了方便。但由于 RGB-D 相机需要发射和接收红外结构光，所以容易受光照影响，因此，使用 RGB-D 相机的 SLAM 系统通常只能在室内使用而不能在室外使用。因为 RGB-D 相机不需要系统计算物体的深度就可以直接得到场景的深度信息，且价格低廉，并且主要应用场景为室内，故本文中 SLAM 系统的传感器的选择为 RGB-D 相机。

1.2 视觉 SLAM 国内外研究现状

1.2.1 国外研究现状

1986年，Smith等人提出了 SLAM 思想，他们的思路是用联合概率分布来表示机器人的状态和所有的路标点，之后利用扩展卡尔曼滤波(Extended Kalman Filter，EKF)算法求解[12]，EKF 的原理是用一阶泰勒展开式近似非线性函数，故可用于非线性状态方程的求解。

文献[13]中 Lu 等人提出了基于图优化的 SLAM 方法。该方法介绍了图优化的基本模型。但是由于计算方式和计算速度，该方法在当时不能满足实时性的要求。近些年来，随着计算速度的提升，基于图优化的 SLAM 已成为当前研究的主流方向[14]。

2007年，Davison教授提出了单目视觉 SLAM 系统 MonoSLAM[15]，这是第一个实时的单目 SLAM 系统，MonoSLAM 的后端为扩展卡尔曼滤波器。这个方法在当时是里程碑式的工作。同年 Klein 等人提出了 PTAM(Parallel Tracking and Mapping)[16]方法，PTAM 使机器人的跟踪与地图构建两个过程并行化，由此视觉 SLAM 第一次提出前后端的概念，同时 PTAM 使用了非线性优化作为后端，并且引入了关键帧机制，即把几个关键的图像帧串联起来，优化机器人的位姿和地图，不需要对每一帧都进行处理，这为 SLAM 的实时性提供了很大的方便。

微软公司在 2010年发布了一款名叫 KINECT 的深度相机，在相机推出之后，华盛顿大学的 Henry等人第一次提出了基于 RGB-D 相机的 SLAM 算法[17]，利用 KINECT 相机得到图像的深度信息，实现同步定位与地图构建的 SLAM 系统，该算法利用图像的 SIFT 特征，并采用随机采样一致性算法(Random Sample Consensus，RANSAC)剔除图像间的错误匹配，在回环检测和全局 BA(Bundle Adjustment)优化之后得到机器人的位姿。之后 Endres等人在此基础上进行研究，提出了 RGB-D SLAM 方法，该方法使用提取图

像的 SURF 特征，用 ICP(Iterative Closest Point)算法估计机器人的位姿，然后采用 G2O(General Graph Optimization)方法进行优化[18]。

在文献[19]中，Newcombe 等人提出了一种基于 RGB-D 相机的视觉 SLAM 方案 KinectFusion，该方法利用相机获得的深度图像生成点云，通过 ICP 算法实时估计机器人的位姿，并实现三维重建。该方法是首个基于 RGB-D 相机的实时三维重建系统。之后，Whelan 等人对 KinectFusion 进行了改进，直接使用 GPU 实现 ICP 算法和直接法的相机位姿估计，并加入了闭环检测，使用 deformation graph 对三维刚体重建做非刚体变换，使闭环后两次重建的结果能够重合[20]。

文献[21]中 Whelan 等人提出了 ElasticFusion 的 SLAM 方法，该方法利用 RGB 的颜色一致性估计相机位姿，并利用深度图像生成点云进行 ICP 来估计相机位姿，该方法充分利用了颜色与深度信息，但是由于代码没有进行优化，所以只适合对房间大小的场景进行重建。

2014 年，Forster 等人提出了一种单目的视觉里程计 SVO(Semi-Direct Sparse Odometry)，该方法基于稀疏直接法，通过最小化稀疏像素点的光度误差来获得相机的运动。既与特征点法不同，该方法不需要计算描述子进行匹配，所以该方法计算速度极快，即使在配置不高的计算平台上也能达到实时的效果[22-23]。虽然 SVO 能构建稀疏点云地图，但它没有回环检测和后端优化的部分，不能解决随着机器人运动时间增加而出现的误差累积问题。同年，Engle 等人提出了 LSD-SLAM(Large Scale Direct SLAM)，LSD-SLAM 将直接法应用到了半稠密的单目 SLAM 中，它不需要计算特征点，但可以在室外大规模环境中构建半稠密的地图，在后端中使用位姿图优化的方法来优化全局位姿，在 CPU 上该方法可以实现实时的效果[24]。

在文献[25]中，Labbé 等人提出了 RTAB SLAM 方法，该方法通过 STM/WM/LTM 的内存管理机制，减少了图优化和闭环检测中需要用到的结点数，保证了系统的实时性和回环检测的准确性，能够在大的场景中运行。

在文献[26]中，Endres 等人提出了 RGBD-SLAM 算法，该算法是一个非常全面的 SLAM 系统，将 SLAM 领域的图像特征、优化、闭环检测、点云、等技术融为一体，但是它的缺点是算法的实时性不好，相机必须缓慢运动。

2015 年，Murartal 等人在文献[27]中提出了基于 ORB 特征的单目 SLAM 系统：ORB-SLAM，该系统为同时运行的三个线程构成，分别为跟踪线程，局部建图线程和回环检测线程，该系统能够在 CPU 上实现实时运行。2017 年，该算法扩展成 ORB-SLAM2[28]，在原有单目相机的基础上还可以支持双目相机和 RGB-D 相机的实现。

在文献[29]中，Ullah 等人提出了一种可以解决尺度漂移问题的多传感器融合的单目 SLAM 算法，该算法使用 IMU 惯性导航传感器解决单目视觉中的尺度漂移问题，然后使用无迹卡尔曼滤波将单目相机和 IMU 传感器估计的机器人位姿融合在一起，该算法的性能较之前的视觉惯导 SLAM 算法更好。

1.2.2 国内研究现状

虽然国内的视觉 SLAM 的研究比国外晚，但是目前国内的许多高校和研究机构都对视觉 SLAM 进行了深入的研究，取得了一些成果。

2012年，北京大学的 Huang等人提出了一种结合 RGB-BA 和 RGBD-BA 的 SLAM 方法[30]，该算法能够减小 SLAM 系统对场景的深度信息的依赖，使 SLAM 系统能够在大的场景中发挥更好的作用。清华大学的杨东方等人提出了一种基于 KINECT 系统的场景建模与机器人自主导航方案[31]，该方法采用了增量式参数化模型来实现降维，能够控制运动尺度不确定性的问题，并且能够保证系统观测状态一致性，实现机器人的位姿估计。

2013年，苏州大学 Li 等人通过 KINECT 相机对天花板和水平空间进行特征跟踪，利用自适应重采样粒子滤波完成了精确的栅格地图重构[32]。

2015 年，北京理工大学的付梦印等人提出了一种实时 SLAM 算法，这是一种对 RGB-D SLAM 改进的算法，通过对 ICP(IterativeClosestPoint)算法中加入粒子采样，提高了算法的定位精度[33]。武汉科技大学的 Min 等人提出了一种 IMU 惯导与 KINECT 融合的 SLAM 方法，同时采用 IMU(Inertial Measurement Unit)数据和图像数据，并通过 EKF 进行运动估计[34]。同年，武汉科技大学的 Shen等人提出了一种基于仿生视觉深度模型的 RGB-D SLAM 算法，该方法将图像的深度值作为参数，对深度信息进行梯度滤波，提高了算法的实时性与准确性[35]。2015 年，高翔等人也提出了一种基于平面二维点特征的 RGB-D SLAM 算法，该算法通过平面二维点减小 ICP 算法的匹配误差，从而提高了精度[36]。

2016年，哈尔滨工业大学的 Xin 等人提出利用 ORB 特征结合 PROSAC的 RGB-D SLAM 算法，该算法基于 G2O进行后端优化，提高了室内机器人 SLAM 的一致性[37]。

2018年，上海交通大学的 Cheng等人在 ORB-SLAM 的基础上，提出了一种基于点和线特征的实时 SLAM 算法[38]，该算法改进了点特征的提取方法，使系统能够减少由于光照变化带来的影响，同时该算法还提取了线特征用于定位，与原始的 ORB-SLAM 相比，该算法的准确性和可用性更好。

2019 年，清华大学的 Han 等人一种可以在不牺牲精度的情况下显著降低系统复杂度的 SLAM 方法[39]，该方法将优化过程中的非线性问题分解为线性分量和非线性分量，其中线性分量可以通过其紧凑的二阶统计量有效地表示，而非线性分量由每个相机位姿的六个自由度表示。该算法通过 CPU 实现的位姿估计与使用 GPU 计算的精度相当。

1.3 研究内容与章节安排

本文研究了基于 RGB-D 相机的 SLAM 系统，系统能够通过 KINECT 相机获得的环境信息实现机器人的同时定位和地图构建。SLAM 系统包括前端的视觉里程计，回环检测，后端优化和地图构建这四个部分，前端的作用是根据相邻两帧之间的特征匹配关系对机器人的运动进行估计，由于运动的估计存在误差，所以通过回环检测消除系统的误差。回环检测的作用是判断机器人是否在历史运动过程中到达过当前的位置，如果检测到发生了回环，就通过回环校正和全局优化来减小系统的误差，最后构建全局一致的地图。同时通过数据集和真实环境的实验对系统的性能进行验证。本文的结构安排如下：

第一章介绍了视觉 SLAM 的研究背景以及国内外的研究现状，并介绍了本文的研究内容和章节安排。

第二章首先介绍了经典的视觉 SLAM 框架，然后详细介绍了基于 KINECT 相机的视觉 SLAM 系统以及 KINECT 相机的结构和原理，并对 KINECT 相机进行标定。

第三章介绍了基于特征点法的视觉里程计，包括图像特征的选择，特征检测和匹配，然后介绍了机器人位姿的估计方法和位姿优化的方法。

第四章介绍了系统的回环检测，后端优化和地图构建的方法，定义了关键帧的概念，通过词袋模型的方法对关键帧进行相似性比较，以此检测是否有回环，通过构建位姿图进行全局优化，然后介绍了构建可用于导航的八叉树地图的方法。

第五章是全文的实验部分，通过数据集和真实环境的实验对系统的性能进行验证。

最后对本文的工作进行总结。

2 基于 RGB-D 相机的 SLAM 系统概述

2.1 经典视觉 SLAM 框架

在视觉 SLAM 中，机器人能顺利地行走依靠的是 SLAM 系统的定位和建图功能。如图 2.1所示为经典的视觉 SLAM 框架。

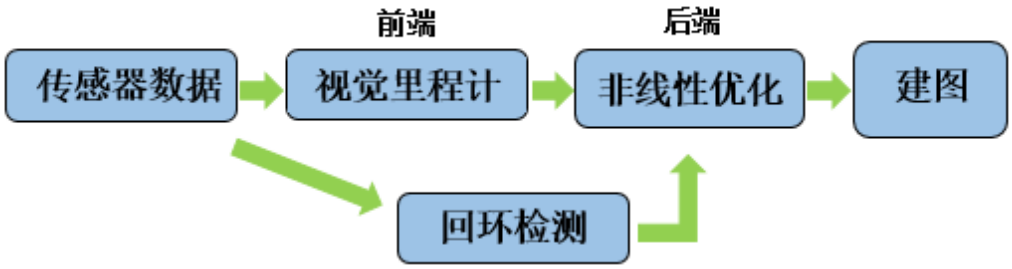


图 2.1 经典视觉 SLAM 框架

Fig. 2.1 Classic framework of visual SLAM

一个完整的视觉 SLAM 系统包括前端的视觉里程计，能够消除累积误差的回环检测(Loop Closing)，对系统进行优化的后端和构建地图(Mapping)这四个部分。整个系统的工作情况如下。

首先通过机器人身上的相机采集环境数据，得到数据后，对采集到的图像进行预处理。

视觉里程计的作用是通过传感器读取的图像信息来估计机器人的位姿和局部的地图。然后将得到的机器人的位姿信息送到后端进行优化。

由于每次的定位都有误差的存在，所以机器人在行走了一定的时间后，误差的累积会使其产生定位漂移，回环检测是消除误差的一种手段。回环检测的作用是判断机器人有没有到达过当前的位置，如果检测到在历史过程中机器人到达过当前的位置，就把信息传给后端。通过回环，能够调整机器人的运动轨迹，矫正由于误差累积造成的定位漂移，降低全局的系统误差。

系统的后端主要是处理机器人行走过程中的噪声问题。它接收前端和回环检测的信息，机器人的位姿进行全局优化，使系统得到较为精确的机器人运动轨迹和环境地图。

建图部分的作用是根据后端的优化结果，建立对应的地图。

2.2 基于 RGB-D 相机 SLAM 系统

2.2.1 SLAM 问题建模

要实现和解决 SLAM 系统中的问题，首先需要建立一个数学模型，用数学模型描述机器人在一段时间内的运动和轨迹。设在某一时刻机器人自身的位置为 $\boldsymbol{x}$, 在一段时间内机器人各个时刻的位置分别为 $\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_K$，它们就构成了这段时间内机器人的运动轨迹。在 SLAM 系统中，地图由许多的路标点组成，地图中的路标点分别用 $\boldsymbol{y}_1,\boldsymbol{y}_2,\cdots,\boldsymbol{y}_N$ 表示。要获得 k-1 时刻和 k 时刻也就是相邻两个时刻之间机器人的运动，就要考虑相机的位置变化关系。两个时刻之间机器人的运动方程可以用式(2.1)描述。

$$\boldsymbol{x}_k = f\left(\boldsymbol{x}_{k-1},\boldsymbol{u}_k,\boldsymbol{w}_k\right) \tag{2.1}$$

式中，函数 f 用来描述机器人的运动过程，$\boldsymbol{u}_k$ 为相机的输入，$\boldsymbol{w}_k$ 代表噪声。观测方程描述的是机器人在 $\boldsymbol{x}_k$ 时刻观测到了路标点 $\boldsymbol{y}_j$ 之后产生了观测数据 $\boldsymbol{z}_{k,j}$，可以用式(2.2)来描述这个关系。

$$\boldsymbol{z}_{k,j} = h(\boldsymbol{y}_j,\boldsymbol{x}_k,\boldsymbol{v}_{k,j}) \tag{2.2}$$

其中，$\boldsymbol{v}_{k,j}$ 是观测的噪声。式(2.1)和式(2.2)这两个方程就是 SLAM 系统要解决的基本问题，在得到观测到的数据 $\boldsymbol{z}$ 和相机的输入 $\boldsymbol{u}$ 之后，求解机器人的定位和建图问题变成了如何估计 $\boldsymbol{x}$ 和 $\boldsymbol{y}$，这是一个状态估计问题。两个方程的形式是线性方程还是非线性方程关系到问题的求解。在本文的 SLAM 系统中方程的形式为非线性，需要利用非线性优化的方法进行求解。

2.2.2 基于 KINECT 相机的 SLAM 系统

基于 KINECT 相机的 SLAM 系统的整体流程如图 2.2 所示。首先通过 KINECT 相机进行图像采集，得到场景的彩色图像和深度图像，然后对采集到的图像提取特征，并对相邻的两帧图像进行特征匹配，根据匹配的结果对相邻两帧之间相机的运动关系进行估计，得到相机的位姿，之后对估计的结果进行优化。由于估计的结果存在误差，所以通过回环检测和全局优化来减小误差。系统的后端接收不同时刻前端和回环检测的信息，对系统中机器人的位姿进行优化，以此来降低系统的误差，在优化后系统可以进行地图构建，得到全局一致的轨迹和地图。

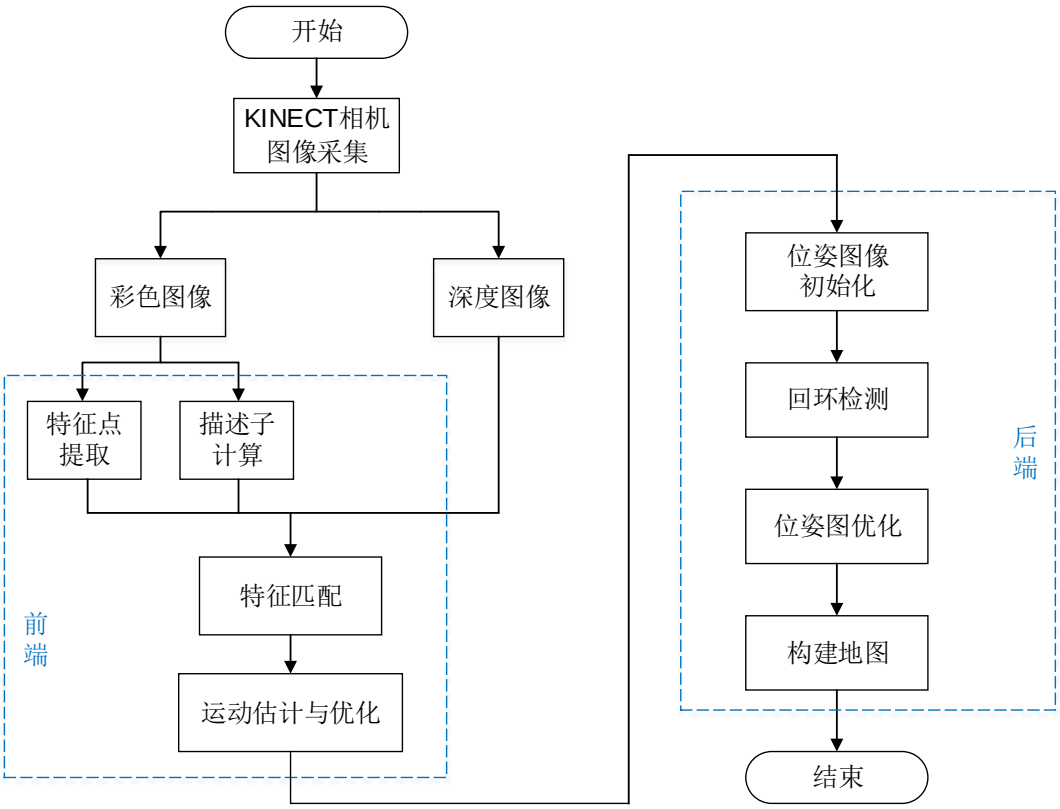


图 2.2　基于 KINECT 相机的 SLAM 系统流程图

Fig. 2.2　Flow chart of SLAM system based on KINECT camera

2.3　KINECT 简介

图 2.3　KINECT 相机

Fig. 2.3　KINECT camera

本文使用的相机为 KINECT 相机，它是微软公司发布的一款深度相机。如图 2.3 所示，它一共有三个摄像头，图 2.3 中最左边的摄像头为红外结构光的发射器，最右边的摄像头为红外结构光的接收器，中间的摄像头为 RGB 摄像头，能够采集场景的彩色图像。通过对红外光发射器和接收器进行分析之后就能够得到与彩色图像对应的深度图像

[40]。此外，在 KINECT 相机还带有麦克风阵列，使相机在获得图像信息的同时也能够获得声音信息。相机的底部还有一个底座，并配有一个马达来调整相机的观测角度。表 2.1 为 KINECT1.0 的相关参数。

表 2.1　KINECT 1.0相关参数

Tab. 2.1　Related parameters of KINECT1.0

相机特性		技术参数
RGB 图像	分辨率	640×480
	fps	30fps
深度图像	分辨率	320×240
	fps	30fps
角度范围	水平方向	57°
	垂直方向	43°
检测范围		0.8m-4.0m
检测人物数量		6 人
音频格式		16KHz，16bit 单声道
接口类型		USB2.0+外部电源

2. 4　KINECT 相机模型

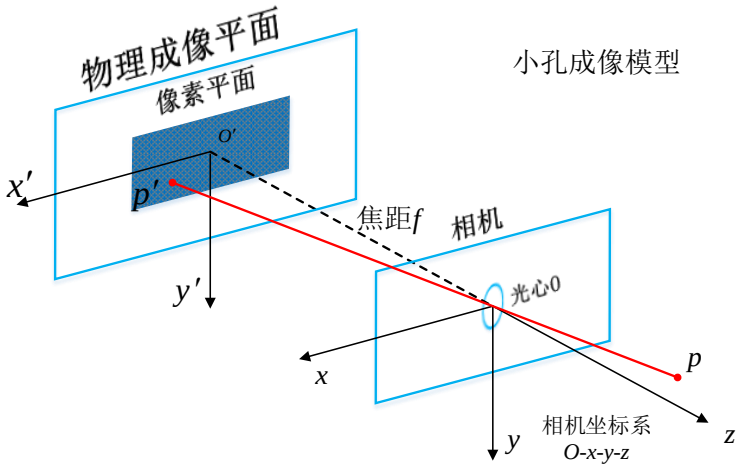


图 2.4　针孔相机模型

Fig. 2.4　Pinhole camera model

KINECT 相机的成像模型为针孔模型。在一束光线通过相机的光心之后，会在相机的光心另一侧的成像平面上投影成像，如图 2.4 所示。

对针孔成像建立数学模型，$O-x-y-z$ 为相机坐标系，其中 O 为相机的光心，z 轴指向相机前方，任意的一个空间点 P 在经过相机的成像模型后成为成像平面 $O'-x'-y'$ 上的点 P'，设 P 点的坐标为 $[X,Y,Z]^{\mathrm{T}}$，P' 点的坐标为 $[X',Y',Z']^{\mathrm{T}}$，相机的焦距为 f，根据三角形相似的原理，可得

$$\frac{Z}{f}=-\frac{X}{X'}=-\frac{Y}{Y'} \tag{2.3}$$

其中，负号代表在成像平面的像与原本的物体相比是倒立的。将式(2.3)中的负号去掉，有

$$\frac{Z}{f}=\frac{X}{X'}=\frac{Y}{Y'} \tag{2.4}$$

整理后可得

$$X'=f\frac{X}{Z} \tag{2.5}$$

$$Y'=f\frac{Y}{Z} \tag{2.6}$$

式(2.5)和式(2.6)描述了空间点 P 与它通过针孔模型成的像之间的关系。之后定义像素坐标系 $o-u-v$，原点在图像的左上角，u 轴与 x 轴平行，v 轴与 y 轴平行。设 P' 点的像素坐标为 $[u,v]^{\mathrm{T}}$，由于成像平面与像素坐标系相差尺度与平移，设 u 轴上像素坐标缩放的倍数为 α，v 轴缩放倍数为 β，原点的平移量为 $[c_x,c_y]^{\mathrm{T}}$，则有

$$\begin{cases}u=\alpha X'+c_x\\ v=\beta Y'+c_y\end{cases} \tag{2.7}$$

将式(2.5)与式(2.6)代入，可得

$$\begin{cases}u=f_x\dfrac{X}{Z}+c_x\\ v=f_y\dfrac{Y}{Z}+c_y\end{cases} \tag{2.8}$$

其中

$$\begin{cases} f_x = \alpha f \\ f_y = \beta f \end{cases} \tag{2.9}$$

这里，f_x，f_y的单位都是像素。

将式(2.9)改写成矩阵形式，为

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \frac{1}{Z}\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \frac{1}{Z}\boldsymbol{KP} \tag{2.10}$$

整理，得

$$Z\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \mathbf{KP} \tag{2.11}$$

式(2.11)中的矩阵 $\boldsymbol{K}$ 是相机的内参，通常，它是固定不变的，确定相机的内参的过程称为相机的标定。除了内参，相机还有外参。在式(2.10)中点 P 的位置描述为相机坐标系中的坐标，而要完成机器人的定位则需要计算它的世界坐标，所以需要完成两个坐标系之间的转换，点 P 在两个坐标系之间的坐标转换关系为

$$Z\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \boldsymbol{KP} = \boldsymbol{K}(\boldsymbol{RP}_w + \boldsymbol{t}) \tag{2.12}$$

其中，矩阵 $\boldsymbol{R}$ 和向量 $\boldsymbol{t}$ 代表相机的位姿，会跟随相机的运动而发生变化。

2.5 KINECT 相机标定

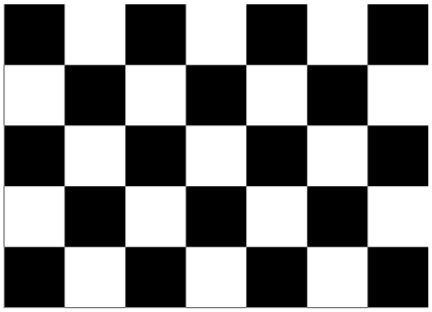


图 2.5 5×7 棋盘格

Fig. 2.5 Checkerboard of 5×7

相机的标定包括对内参和外参的标定，内参指的是摄像头自身的参数，外参指的是两个摄像头之间的相对位姿。在 KINCET 相机中有三个摄像头，但只需要对 RGB 摄像头和获得深度图像的红外摄像头标定即可，目前标定算法已经成熟[41]，本文采用张正友的棋盘格标定法[42]进行标定，采用 5×7 的黑白棋盘格对 KINECT 相机标定，如图 2.5 所示。

为了标定的结果更加准确，需要从各种角度和距离拍摄不少于 20 张的 RGB 图像和深度图像。得到用于标定的图像后，由于在 MATLAB 中有相机标定工具包 Camera Calibration Toolbox，所以使用工具包对采集的图像进行标定计算，就能分别计算出相机内部两个摄像头的内参和相机的外参。表 2.2 中的数据为 KINECT 相机内参标定的结果，其中 p_1 和 p_2 是镜头的切向畸变，k_1，k_2 和 k_3 是镜头的径向畸变。

表 2.2　KINECT 相机内参标定结果

Tab. 22　Result of internal reference calibration in KINECT camera

参数		RGB 摄像机	深度摄像机
焦距	f_x	517.306408	520.908620
	f_y	516.469215	521.007327
中心点	c_x	318.643040	325.14442
	c_y	255.313989	249.701764
畸变	k_1	0.262383	0.231222
	k_2	-0.953104	-0.784899
	k_3	1.163314	-0.003257
	p_1	-0.005358	-0.000105
	p_2	0.002628	0.917205

两摄像头之间的相对位姿即 KINECT 的外参，矩阵 R 和矩阵 t 的标定结果如下。

$$\boldsymbol{R}=\begin{bmatrix} 0.999854 & -0.006362 & 0.02382 \\ -0.003561 & 0.999641 & 0.03287 \\ -0.024053 & -0.038459 & 0.99970 \end{bmatrix}$$

$$\boldsymbol{t} = \begin{bmatrix} -0.267015 \\ -0.490854 \\ -0.961431 \end{bmatrix}$$

2.6 本章小结

本章主要介绍了基于 RGB-D 相机的视觉 SLAM 系统的相关基础知识及概念。首先系统介绍了经典的视觉 SLAM 框架。然后对基于 KINECT 相机的 SLAM 系统整体流程进行说明，并介绍了 KINECT 相机的结构、原理以及相机模型，最后对 KINECT 相机进行标定，得到了相机的内参和外参。

3　基于特征点法的视觉里程计

3.1　引言

在第 2 章的经典视觉 SLAM 框架中提到，视觉里程计的主要功能是根据相邻两帧之间的运动关系来估计相机的运动，并对相机的位姿进行优化，能够给后端的优化提供比较好的初始值。

利用相邻两帧之间的对应关系估计相机位姿的方法有特征点法和直接法两种方法。特征点法的视觉里程计需要提取图像的特征点进行匹配，以此来估计相机的运动；另一种方法不提取图像的特征点，而是直接根据图像中的像素计算两幅图像之间的关系，以此来估计相机的运动。由于直接法受光照等因素影响较大，且不稳定，所以本文选择特征点法实现视觉里程计，特征提取与匹配的流程如图 3.1 所示。

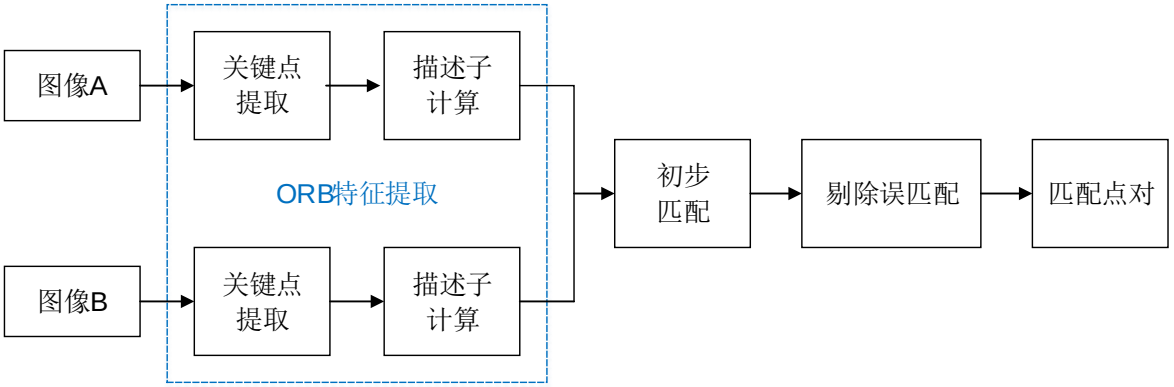


图 3.1　图像特征提取与匹配流程

Fig. 3.1　Process of image feature extraction and matching

3.2　图像特征检测

图像的特征点是从图像中提取的某些可以代表整幅图像的点。通常情况下，图像发生某些变化，特征点并不会改变，正是根据这种特性，才可以用特征点进行图像之间的关联。在视觉 SLAM 框架中，图像的特征点就是观测模型中的路标点。图像的特征点有很多种，例如图像中的边缘点，区块点和角点等都可以作为图像的特征，如图 3.2 所示。但是边缘点和区块点在图像中却比较难提取和匹配，而角点作为特征点的提取和匹配相对来说比较容易和准确，所以角点的辨识度更高。

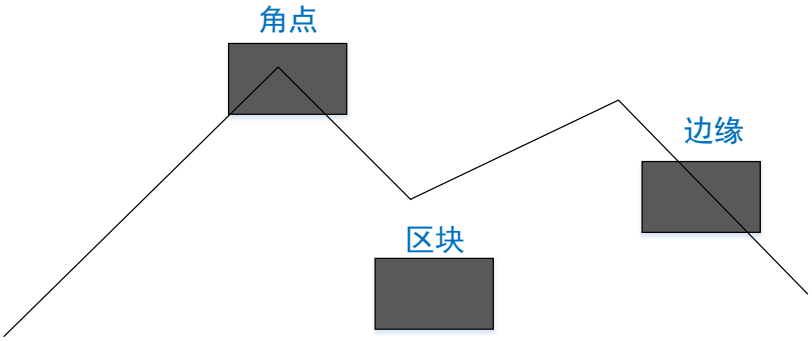


图 3.2　图像特征示例

Fig. 3.2　Exampleof image feature

但是，单纯的角点特征也存在着一些问题，比如相机的旋转和远近会使角点发生变化，无法进行角点辨别。针对这个问题，研究者们设计出了辨识度更高，更加稳定的人工图像特征，例如 SIFT 特征，SURF 特征，ORB 特征[43-45]等。这些特征点由两部分构成，分别是关键点和描述子。当提取某一特征时，就代表既提取了该特征的关键点，也得到了相应的描述子。其中关键点包含了特征点的位置、方向等信息，描述子是对提取的关键点的周围信息的一种描述，一般为向量形式。理论上当两幅图像具有相似的特征点时它们的描述子也应该是相似的，所以，通过比较两个特征点对应的描述子就可以判断特征点是否相同，根据这个原理就可以对图像间的特征进行匹配以获得两幅图像之间的关系。

在视觉 SLAM 中图像特征的选择尤为重要，如果特征点数量少，系统的精确度会下降，对后续的估计和跟踪有所不利，而特征点数目多就会带来计算量大的问题，使系统不能够实时运行。根据文献[43-45]中的作者在论文中提供的信息，在同一图像中提取同样数量的特征点时，ORB 特征提取的时间远小于 SIFT 特征和 SURF 特征，由于在 SLAM 系统中定位和建图需要实时，所以在图像的特征数量满足的条件下，选择 ORB 特征完成 SLAM 系统前端的位姿估计。

ORB 特征的关键点(Oriented FAST)是一种改进的 FAST 角点，FAST 关键点主要分布在图像中某些像素灰度变化比较显著的地方，例如一块比较暗的地方中的一个亮点，或者一块比较亮的区域中的一个暗的地方。提取 FAST 关键点的思想是如果一个像素点与它的周围区域内足够多的像素点灰度值差别较大，则该点为 FAST 关键点，如图 3.3 所示[46]。FAST 角点提取的原理为：确定图像中的某一像素点 p，在像素点 p 的半径为 3 的圆周上一共有 16 个像素点，每一个像素点 x 相对于像素点 p 的状态有三种，如式(3.1)所示。

$$S_{p\to x}=\begin{cases} d, I_{p\to x}\le I_p - t & (\text{较暗}) \\ s, I_p - t \le I_{p\to x} < I_p + t & (\text{相似}) \\ b, I_p + t \le I_{p\to x} & (\text{较亮}) \end{cases} \tag{3.1}$$

其中，I_p 为像素点 p 的灰度值，$I_{p\to x}$ 为圆周上的像素点 x 的灰度值，t 为预先设置的阈值，d 表示点 x 的亮度比点 p 底，b 表示点 x 的亮度比点 p 高，s 表示两点灰度值相似。如果存在连续 N 个点与 p 点的比较结果为 d 或者 b，就可以认为点 p 是 FAST 关键点。对一幅图像中的所有像素都利用上述方法比较，就能得到该图像的 FAST 关键点。

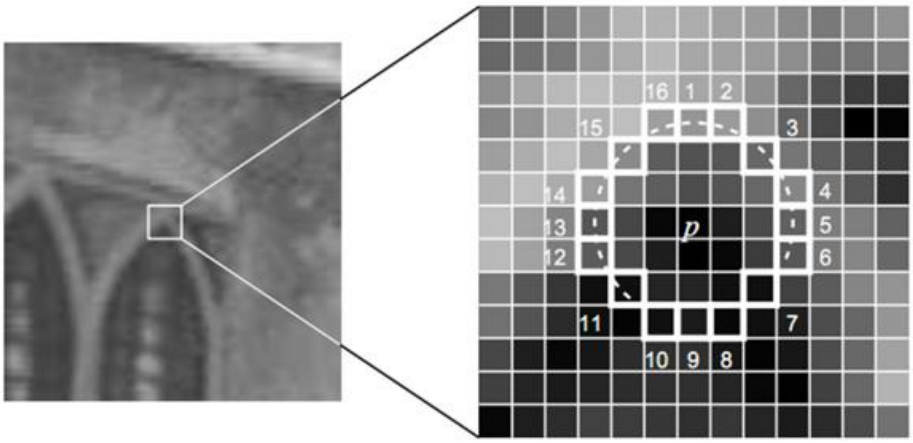


图 3.3　FAST 关键点

Fig. 3.3　key-pointsof FAST

FAST 关键点不包含特征点的方向信息和尺度信息，在图像发生旋转和尺度缩放的情况下会对系统产生一定的影响。为了解决这个问题，通过构建图像金字塔为 ORB 特征增加了尺度不变性，通过灰度质心法为特征增加了旋转不变性，使 ORB 特征更加稳定和精确。

图像的尺度可以理解为，对于一幅图像，通过不同的距离观察看到的图像效果通常不同。近距离观察时看到的画面能够看到图像的一些细节信息，远距离观察时图像就会比较模糊，图像的尺度是自然存在的。图像金字塔是将同一幅图像的不同尺寸集合在一起，通过对图像向下采样可以得到一层一层以金字塔形状排列的分辨率逐步下降的图像集合，金字塔的每一层的图像都来源于同一张原始图像，如图 3.4 所示。虽然下采样的过程中不会引入其它噪声，但是在图像缩放的过程中信息会丢失，所以图像分辨率会降低。金字塔底层的图像分辨率最高，金字塔层数越高，图像分辨率越低，金字塔顶层的

图像分辨率是最低的。得到不同尺度的图像之后，在每一层不同分辨率的图像上都提取FAST关键点，就得到了具有尺度不变性的ORB特征点。

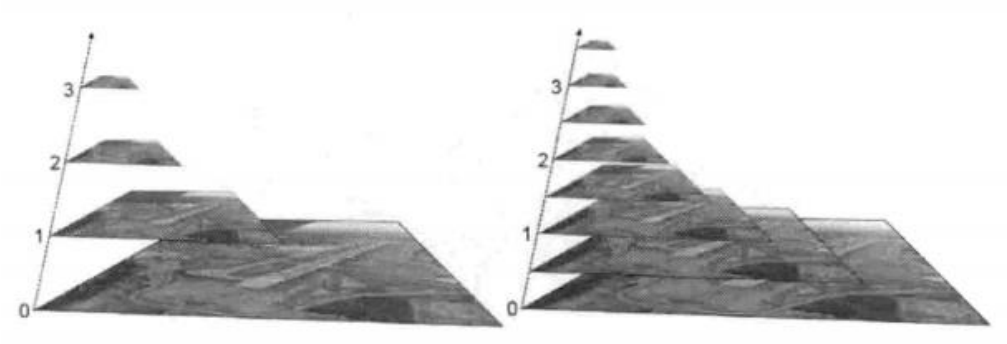


图3.4　图像金字塔

Fig.3.4　Image pyramid

为了给ORB特征添加旋转不变性，利用灰度质心法定义特征点的方向。图像的质心为一个图像块内所有像素的灰度值权重的中心，特征点的方向定义为该特征点到这个特征点所在的图像块内灰度质心的向量方向，其具体计算方法如下[47]。

在一个小的图像块 A 中，每个像素位置 (x,y) 对应的灰度值为 $I(x,y)$，则图像块 A 的矩为

$$m_{pq}=\sum_{x,y\in A}x^{p}y^{q}I(x,y),p,q=\{0,1\} \tag{3.2}$$

则图像块 A 的0阶矩和1阶矩为

$$m_{00}=\sum_{x,y}I(x,y) \tag{3.3}$$

$$m_{10}=x\sum_{x,y}I(x,y) \tag{3.4}$$

$$m_{01}=y\sum_{x,y}I(x,y) \tag{3.5}$$

那么可以得到图像块 A 的质心为

$$C=(\frac{m_{10}}{m_{00}},\frac{m_{01}}{m_{00}}) \tag{3.6}$$

由此就可以得到图像块的几何中心 O 到它的质心 C 的向量 $\overrightarrow{OC}$，则特征点的主方向为

$$\theta = \arctan(m_{01}, m_{10}) \tag{3.7}$$

其中，反正切函数 arctan返回的是 FAST 角点到(m_{01}, m_{10})的方位角。在计算过程中，不需要考虑 FAST 角点是较亮的还是较暗的点，因为角度的测量是一致的，不依赖角点的类型。

ORB 特征的描述子为 BRIEF(Binary Robust Independent Elementary Features)描述子，用于描述特征点的属性。BRIEF 描述子是由 0 和 1 组成的二进制的描述向量，存储方便，适合实时的图像匹配。算法的主要思想是在提取的关键点 p 的附近选取 N 对像素点，然后将像素点的灰度的比较结果用 0 和 1 表示，组成描述向量，本文中 N 的值为 256。算法的具体步骤如下。

Step1 在以关键点 p 为圆心，半径为 d 的圆内选取 N 个像素点对，为方便说明，取 $N=4$。选取的 4 对像素点分别为 $P_1(A,B)$，$P_2(A,B)$，$P_3(A,B)$，$P_4(A,B)$ 。

Step2 定义 T 的取值为：

$$T(P(A,B)) = \begin{cases} 1 & I_A > I_B \\ 0 & I_A < I_B \end{cases} \tag{3.8}$$

其中，I_A 和 I_B 分别为像素点 A 和像素点 B 的灰度值。

Step3 对选取的点对分别进行比较，每一个点对都得到一个 T 的值，组成最终的二进制描述子。

在得到改进后的 FAST 关键点和改进后的 BRIEF 描述子之后，ORB 特征就具有了尺度不变特性和旋转不变特性，采用改进后的 ORB 特征完成图像特征检测可以为之后的环节提供更高的准确度。

3.3 图像特征匹配

在 SLAM 系统中图像特征匹配是很重要的，通过特征匹配能够使机器人当前观测到的路标与之前观测到的路标联系起来，从而实现数据的关联，高准确度的特征匹配结果能够为机器人的位姿估计和后端的计算优化提供很大的优势。要实现特征匹配首先需要建立一个衡量标准，由于本文中 ORB 特征点的描述子是二进制的向量，所以特征之间的相似性可以利用特征点之间的距离来表示，本文选择汉明距离作为特征之间的相似性度量。

在两幅图像进行特征匹配时，将两幅图像中汉明距离最小的点作为匹配的特征点对，如图 3.5 所示为初步匹配效果图。

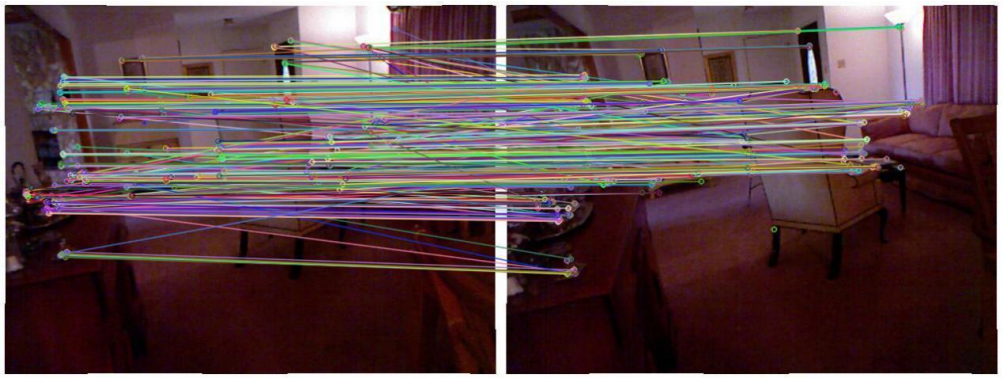


图 3.5　初步匹配结果图

Fig. 3.5　Result ofpreliminary matching

在初步匹配的结果中可以看出来，用最小汉明距离作为匹配条件的匹配结果存在着许多错误的匹配。错误的匹配对后续的计算和估计会产生很严重的影响，所以需要某些条件来去除错误的匹配。传统的方法是通过设置阈值来去除误匹配，常用的方法为：首先在计算两幅图像的特征点之间的距离时保留所有距离中的最小值 $d_{\min}$，然后将所有大于 2 倍最小距离 $d_{\min}$ 的匹配点对剔除。基于传统的去除误匹配的方法，本文增加了筛选错误匹配的条件，利用次优匹配来进一步剔除错误的匹配。具体方法为：首先在计算两幅图像的特征点的距离时不仅保留待匹配的特征点与候选匹配点的距离最小值，还需要保留一个距离次小值，在将所有大于 2 倍最小距离 $d_{\min}$ 的匹配点对剔除并得到初次筛选后的匹配点对后，利用次优匹配进一步筛选，进一步筛选的条件是，如果在匹配时待匹配的特征点与候选的特征点之间的距离次小值大于距离最小值的 10 倍，即最小值远小于次小值，则认为匹配点对有效，进行保留。否则即最小值与次小值相差不多，就剔除该特征点对。剔除错误匹配之后的效果如图 3.6 所示。

图 3.6　筛选后匹配结果图

Fig. 3.6　Result ofmatching afterfiltering

3.4 运动估计与优化

3.4.1 位姿估计方法

得到相邻两帧之间匹配的特征点对之后，就可以根据匹配关系计算相机的位姿和运动关系。相邻两帧之间机器人的运动关系可以用旋转矩阵 $\boldsymbol{R}$ 和平移向量 $\boldsymbol{t}$ 来描述，在三维空间中，旋转矩阵 $\boldsymbol{R}$ 为3×3的矩阵，用于描述相机的旋转，描述向量 $\boldsymbol{t}$ 的维度为三维，描述相机的平移。在某时刻 k 机器人的坐标为 $\boldsymbol{P}_k$，则在 $k+1$ 时刻机器人的位置为

$$\boldsymbol{P}_{k+1} = \boldsymbol{R}\boldsymbol{P}_k + \boldsymbol{t} \tag{3.9}$$

根据特征匹配估计相机运动的方法有三种，分别是 2D-2D，3D-2D，3D-3D[48]的方法。其中 2D-2D 的方法[49]常用于单目相机的 SLAM 系统，这种方法也称为对极几何的方法。这种方法是在两张图像已经完成特征匹配之后，从它们之间的匹配关系中求出相机的运动和位姿。设两帧图像分别为 I_1 和 I_2，I_1 中有一个特征点 p_1，与 I_2 中的特征点 p_2 是相互匹配，它们对应的空间中的点为 P。在 I_1 的坐标系下，P 的坐标为 $\boldsymbol{P} = [X,Y,Z]^{\mathrm{T}}$，特征点 p_1 和 p_2 的像素位置为式(3.10)所示。

$$\boldsymbol{p}_1 = \boldsymbol{K}\boldsymbol{P}\ ,\ \boldsymbol{p}_2 = \boldsymbol{K}(\boldsymbol{R}\boldsymbol{P} + \boldsymbol{t}) \tag{3.10}$$

取

$$\boldsymbol{x}_1 = \boldsymbol{K}^{-1}\boldsymbol{p}_1,\quad \boldsymbol{x}_2 = \boldsymbol{K}^{-1}\boldsymbol{p}_2 \tag{3.11}$$

其中，$\boldsymbol{x}_1$ 和 $\boldsymbol{x}_2$ 是两个特征点的归一化平面上的坐标，将其带入式(3.10)，可得

$$\boldsymbol{x}_2 = \boldsymbol{R}\boldsymbol{x}_1 + \boldsymbol{t} \tag{3.12}$$

两边同时左乘 $\boldsymbol{t}^{\wedge}$，相当于两边同时和 $\boldsymbol{t}$ 做外积，得到式(3.13)。

$$\boldsymbol{t}^{\wedge}\boldsymbol{x}_2 = \boldsymbol{t}^{\wedge}\boldsymbol{R}\boldsymbol{x}_1 \tag{3.13}$$

其中，$\wedge$ 是一个将 向量变成反对称矩阵的符号。对式(3.13)两边同时左乘 $\boldsymbol{x}_2^{\mathrm{T}}$，得到

$$\boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{t}^{\wedge}\boldsymbol{x}_2 = \boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{t}^{\wedge}\boldsymbol{R}\boldsymbol{x}_1 \tag{3.14}$$

由于 $\boldsymbol{t}^{\wedge}\boldsymbol{x}_2$ 与 $\boldsymbol{t}$ 和 $\boldsymbol{x}_2$ 都是垂直的，所以它和 $\boldsymbol{x}_2$ 做内积的结果为 0，于是可以得到

$$\boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{t}^{\wedge}\boldsymbol{R}\boldsymbol{x}_1 = 0 \tag{3.15}$$

将 $\boldsymbol{p}_1$ 和 $\boldsymbol{p}_2$ 代入，得

$$\boldsymbol{p}_2^{\mathrm{T}}\boldsymbol{K}^{-\mathrm{T}}\boldsymbol{t}^{\wedge}\boldsymbol{R}\boldsymbol{K}^{-1}\boldsymbol{p}_1=0 \tag{3.16}$$

式(3.16)称为对极约束，对极约束中同时包含旋转矩阵和平移向量，进一步简化对极约束，记

$$\boldsymbol{E}=\boldsymbol{t}^{\wedge}\boldsymbol{R},\boldsymbol{F}=\boldsymbol{K}^{-\mathrm{T}}\boldsymbol{E}\boldsymbol{K}^{-1},\boldsymbol{x}_2^{\mathrm{T}}\boldsymbol{E}\boldsymbol{x}_1=\boldsymbol{p}_2^{\mathrm{T}}\boldsymbol{F}\boldsymbol{p}_1=0 \tag{3.17}$$

其中，$\boldsymbol{E}$ 是本质矩阵，$\boldsymbol{F}$ 是基础矩阵。因此，估计相邻两帧之间相机的位姿就变为以下两步：(1)根据匹配的特征点的位置求出本质矩阵 $\boldsymbol{E}$ 或者基础矩阵 $\boldsymbol{F}$；(2)由 $\boldsymbol{E}$ 或 $\boldsymbol{F}$ 求出 $\boldsymbol{R}$ 和 $\boldsymbol{t}$。在实际的工程中，通常使用 $\boldsymbol{E}$ 来求解。$\boldsymbol{E}$ 实际有 5 个自由度，常用五点法或者经典的八点法[50-51]求解，通过奇异值分解(Singular Value Decomposition，SVD)就可以得到 $\boldsymbol{R}$ 和 $\boldsymbol{t}$。

3D-3D 的位姿估计是利用一组匹配好的 3D 点来估计相机的运动，用于双目相机和 RGB-D 相机的 SLAM 系统。设有一组匹配好的 3D 点，分别是：$P=\{p_1,p_2,\cdots p_n\}$ 和 $P'=\{p_1',p_2',\cdots p_n'\}$，目标就是找到一个变换 $\boldsymbol{R}$ 和 $\boldsymbol{t}$，使得

$$\forall i,\boldsymbol{p}_i=\boldsymbol{R}\boldsymbol{p}_i'+\boldsymbol{t} \tag{3.18}$$

这个问题可以使用迭代最近点(Iterative Closest Point，ICP)方法求解[52]。ICP 的求解主要分为以下四个步骤。

Step1：定义第 i 对点的误差项

$$e_i=\boldsymbol{p}_i-(\boldsymbol{R}\boldsymbol{p}_i'+\boldsymbol{t}) \tag{3.19}$$

令误差平方和最小，构建最小二乘问题，求解 $\boldsymbol{R}$ 和 $\boldsymbol{t}$，如式(3.20)所示。

$$\min_{\boldsymbol{R},\boldsymbol{t}}J=\frac{1}{2}\sum_{i=1}^{n}\left\|(\boldsymbol{p}_i-(\boldsymbol{R}\boldsymbol{p}_i'+\boldsymbol{t}))\right\|_2^2 \tag{3.20}$$

Step2：计算两组点的质心位置

$$\boldsymbol{p}=\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{p}_i,\quad \boldsymbol{p}'=\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{p}_i' \tag{3.21}$$

然后计算每个点的去质心坐标

$$\boldsymbol{q}_i=\boldsymbol{p}_i-\boldsymbol{p},\quad \boldsymbol{q}_i'=\boldsymbol{p}_i'-\boldsymbol{p}' \tag{3.22}$$

Step3 计算旋转矩阵

$$\boldsymbol{R}^{*}=\arg\min_{\boldsymbol{R}}\frac{1}{2}\sum_{i=1}^{n}\left\|\boldsymbol{q}_i-\boldsymbol{R}\boldsymbol{q}_i'\right\|^2 \tag{3.23}$$

Step4 根据第 3 步的 $\boldsymbol{R}$ 计算 $\boldsymbol{t}$

$$\boldsymbol{t}^{*}=\boldsymbol{p}-\boldsymbol{R}\boldsymbol{p}' \tag{3.24}$$

为了求解问题中最优的旋转矩阵 $\boldsymbol{R}$，先定义矩阵

$$\boldsymbol{W}=\sum_{i=1}^{n}\boldsymbol{q}_i\boldsymbol{q}_i'^{\mathrm{T}} \tag{3.25}$$

式中，$\boldsymbol{W}$ 是一个 3×3 的矩阵，对 $\boldsymbol{W}$ 进行 SVD 分解，可得

$$\boldsymbol{W}=\boldsymbol{U}\sum\boldsymbol{V}^{\mathrm{T}} \tag{3.26}$$

其中，$\sum$、$\boldsymbol{U}$ 和 $\boldsymbol{V}$ 都为对角矩阵，矩阵 $\sum$ 的对角线元素为从大到小排列的奇异值组成。当 $\boldsymbol{W}$ 满秩时，$\boldsymbol{R}$ 为

$$\boldsymbol{R}=\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}} \tag{3.27}$$

解得 $\boldsymbol{R}$ 后，根据式(3.24)就可以求出 $\boldsymbol{t}$。

3D-2D 的方法描述的是在获得 3D 空间点和它们对应的投影位置时估计相机运动的方法，它的求解方法为 PnP(Perspective-n-Point)方法[53]，在本文中，通过 KINECT 相机采集的深度图可以得到特征点的 3D 位置，因此本文中估计相机位姿和相机运动的方法为 PnP 方法，PnP 问题的解法有 P3P 方法[54]，UPnP 方法[55]和 EPnP(Efficient PnP)方法[56]等。本文采用 EPnP方法作为 PnP 问题的解法。

EPnP 方法的中心思想是在相机坐标系和世界坐标系中有相对应的非共面的四个控制点，相机坐标系或世界坐标系中的任意一点都可以用相应坐标系中的四个控制点的线性组合表示出来，如果相机坐标系和世界坐标系中的四个控制点都能够确定，就可以根据它们求出相机的位姿 $\boldsymbol{R}$ 和 $\boldsymbol{t}$。所以问题就转化为求解控制点的坐标。设世界坐标系下四个控制点为 c_j^w，其中 $j=1,\cdots4$，某个 3D 点为 p_i^w。在两个坐标系下，四个控制点的线性组合可以表示任意的 3D 点。

$$p_i^w=\sum_{j=1}^{4}\alpha_{ij}c_j^w\ ,\quad st.\sum_{j=1}^{4}\alpha_{ij}=1 \tag{3.28}$$

其中，α_{ij}是控制点的系数，对每一个 3D 点，四个系数的和都为 1。同样，相机坐标系中每一个点 p_i^c 也可以用四个控制点的线性组合表示。

$$p_i^c = \sum_{j=1}^{4} \alpha_{ij} c_j^c \ , \ \ s.t. \sum_{j=1}^{4} \alpha_{ij} = 1 \tag{3.29}$$

在式(3.28)和式(3.29)中，p_i^w 和 c_j^w 的位置坐标为世界坐标系中的坐标，p_i^c 和 c_j^c 的位置为相机坐标系中的坐标。根据相机的投影模型

$$\lambda_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = p \cdot \sum_{j=1}^{4} \alpha_{ij} c_j^c = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^{4} \alpha_{ij} \begin{bmatrix} x_j^c \\ y_j^c \\ z_j^c \end{bmatrix} \tag{3.30}$$

展开可得

$$\sum_{j=1}^{4} \alpha_{ij} f_u x_j^c + \alpha_{ij} (u_c - u_i) z_j^c = 0 \tag{3.31}$$

$$\sum_{j=1}^{4} \alpha_{ij} f_v y_j^c + \alpha_{ij} (v_c - v_i) z_j^c = 0 \tag{3.32}$$

将式(3.31)和式(3.32)写成矩阵的形式，可以得到一个线性方程

$$\boldsymbol{Mx = 0} \tag{3.33}$$

其中，待求量 $\boldsymbol{x}$ 为四个控制点在相机坐标系下的坐标：$\boldsymbol{x} = [c_1^{cT}, c_2^{cT}, c_3^{cT}, c_4^{cT}]$，方程(3.33)的解位于矩阵 $\boldsymbol{M}$ 的核空间

$$\boldsymbol{x} = \sum_{i=1}^{N} \beta_i v_i \tag{3.34}$$

其中，v_i 是矩阵 $\boldsymbol{M}$ 的 N 个零特征值对应的特征向量，β_i 可以通过在世界坐标系和相机坐标系中控制点两两之间的距离相同求得，这样就可以得到四个世界坐标系中的控制点在相机坐标系中的坐标，此时可以用 ICP 求解方法中的步骤 2 到步骤 4 来求解相机位姿。

3.4.2 位姿优化

在求得相机位姿初始解后，需要对相机的位姿进行优化。带有相机位姿和空间点的优化模型称为 Bundle Adjustment 简称 BA[57]，BA 是一个图优化模型[58]。在 BA 中，把机器人的位姿和地图点都作为优化变量进行优化，在机器人的位姿和地图点的位置做

出最优的调整之后，使从每一个地图点反射出来的光线最后都通过相机的光心这个过程就是 BA。在 PnP中，BA 问题是一个最小化重投影误差的问题，如图 3.7 所示。重投影指的是空间点的第二次投影，第一次投影指的是相机的某一帧中所对应的三维空间点投影到这一帧上，投影点为 p_1，然后利用深度信息和几何信息可以计算出一个空间点的位置 P，之后利用这个空间点的坐标和计算出来的相机位姿进行第二次投影，而重投影误差就是图像中真实的像素点的位置和重投影得到的空间点的位置之间的误差，通过将这个误差最小化来得到最优的相机位姿和三维空间点的坐标。

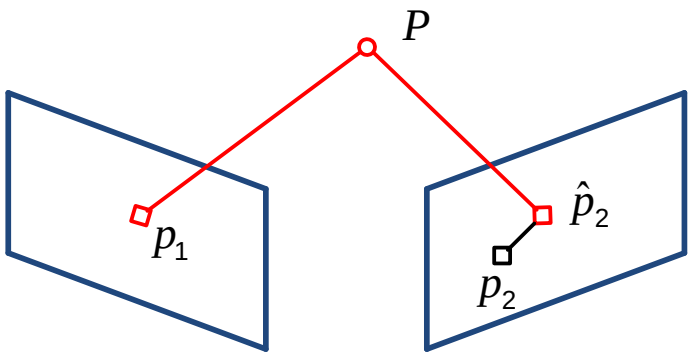


图 3.7　重投影误差示意图

Fig. 3.7 Diagramof reprojection error

如图 3.7 所示，空间点 p 的坐标为 $\boldsymbol{P}_i=[X_i,Y_i,Z_i]^{\mathrm{T}}$，在前一帧中的像素坐标为 $\boldsymbol{u}_i=[u_i,v_i]^{\mathrm{T}}$，则有

$$s_i\begin{bmatrix}u_i\\v_i\\1\end{bmatrix}=\boldsymbol{K}\exp(\boldsymbol{\xi}^{\wedge})\begin{bmatrix}X_i\\Y_i\\Z_i\\1\end{bmatrix} \tag{3.35}$$

其中，s_i 代表像素点的深度信息，$\boldsymbol{\xi}$ 为相机位姿的李代数[48]表示，式(3.35)的矩阵形式为

$$s_i\boldsymbol{u}_i=\boldsymbol{K}\exp(\boldsymbol{\xi}^{\wedge})\boldsymbol{P}_i \tag{3.36}$$

由于噪声的存在，式(3.36)存在一个误差，对误差求和并使它最小化，得到优化后的相机位姿。

$$\xi^* = \arg\min_{\xi} \frac{1}{2} \sum_{i=1}^{n} \left\| \boldsymbol{u}_i - \frac{1}{s_i} \boldsymbol{K} \exp(\hat{\xi}) \boldsymbol{P}_i \right\|_2^2 \tag{3.37}$$

前面提到，BA 实际上是一个图优化模型，所以式(3.37)中的 BA 问题可以使用图优化的方法来求解。图优化是将非线性优化和图论结合起来，通过图模型，使优化问题更加直观地表达出来[58]。在图模型中，图是由顶点和边组成的。在图优化问题中，顶点为优化变量，连接顶点的边为优化问题的约束项。图 3.8 是一个图优化的例子，图中的圆形和三角形为待优化的变量，实线和虚线为约束，连接图中的顶点。

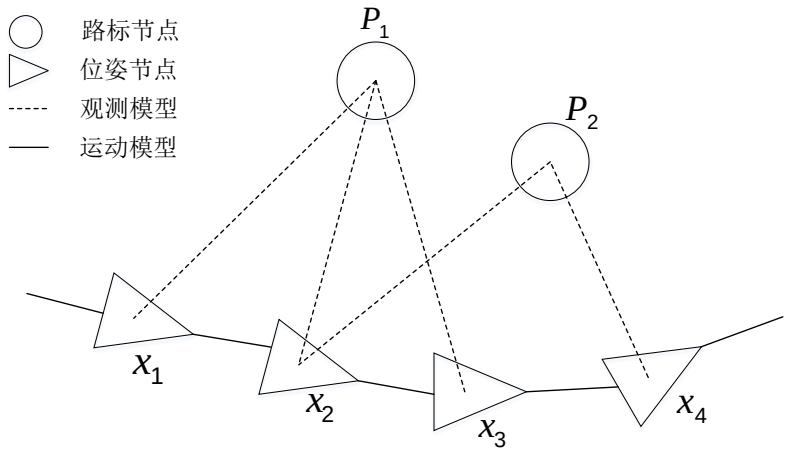


图 3.8　图优化示例

Fig. 3.8　Exampleof graph optimization

式(3.36)中的最小二乘问题是由许多误差项之和组成的，然而从式中只能看出有一组优化变量和多组误差项，并不能看出优化变量和误差项之间的关联，将这个优化问题用图模型表示，可以直观地看出优化变量和误差项之间的关联，利用图模型的某些性质，可以对这个问题做更好的优化。本文中图优化的求解使用 G2O(General Graphic Optimization)方法，G2O 是一个基于图优化的算法开源库，用于求解非线性最小二乘问题，它提供了大量的顶点和边的类型，非常便于相机位姿优化。G2O 中还提供多种迭代算法，包括高斯牛顿法和列文伯格-马夸尔特方法等，在优化之前，用户需要选择一种想要使用的迭代算法进行求解。在 G2O 框架中对图优化问题求解的步骤为：

(1) 定义顶点和边的类型；

(2) 选择优化算法；

(3) 向图中添加顶点和边；

(4) 调用 g2o 进行优化，返回结果。

在 SLAM 系统的位姿优化中，BA 是一种常用的方法。在前端，通常只需要考虑局部相机位姿和特征点的小型 BA 问题，希望对它进行实时求解和优化。但 BA 不仅限于小规模的优化问题，也可以对多幅图像匹配到的位姿和空间点进行迭代优化，这种做法规模较大，主要在后端使用。

3.5 本章小结

本章详细介绍了基于 KINECT 相机的 SLAM 系统前端的实现，系统的前端采用特征点法，通过对不同特征对比后选择 ORB 特征，通过图像金字塔和灰度质心法为 ORB 特征添加尺度不变性和旋转不变性。使用汉明距离作为度量进行特征匹配，针对在匹配结果中出现的误匹配情况，通过设置距离阈值和利用次优匹配来剔除错误的匹配。得到正确的特征匹配结果之后，利用 EPnP 的方法估计机器人的运动和位姿，然后用图优化的方法对估计出的位姿进行优化，实现了一个简单的视觉里程计。

4 回环检测与后端优化

4.1 引言

在完成图像的特征检测与匹配以及机器人的位姿估计之后，由于随着机器人运动时间的增加会导致系统误差累积，所以为了降低系统误差，提高系统精度，还需要在后期进行回环检测和全局位姿优化。本章主要介绍回环检测和全局优化的方法以及地图的构建。

SLAM 系统中KINECT相机采集图像的频率是每秒30帧，在机器人行走的过程中，系统采集的图像数量是很大的，如果对采集的每一帧图像都进行后端的优化和回环检测，那么整个系统的计算量会非常庞大，使 SLAM 系统难以达到实时的要求。针对这个问题，需要在后期的回环检测和优化过程中定义关键帧，通过选取关键帧进行回环检测和全局优化，能够节约计算机的存储空间，减少计算量，有利于系统实时性的实现。

4.2 关键帧的选择

在 SLAM 系统中，关键帧是机器人在运动的过程中一些比较有代表性的图像帧，它的选择是很重要的。如果系统保存的关键帧过多，就会造成冗余的计算量，失去了关键帧的意义，而如果系统保存的关键帧很少，相邻两个关键帧之间差异很大，就会导致优化和回环检测结果不准确，所以应该制定适当的关键帧选取方案，保证关键帧的合理性。关键帧的选取应该遵循几个原则，首先如果系统较长时间没有生成关键帧，就应该生成关键帧，如果检测到机器人此时的位姿状态相比上一个关键帧时刻的运动变化比较明显也要生成关键帧。机器人的运动变化可以用当前帧与上一关键帧之间的旋转和平移量来衡量，如式(4.1)所示。

$$\varepsilon=\lambda_1\|\Delta\boldsymbol{t}\|_2+\lambda_2\|\Delta\boldsymbol{\theta}\|_2 \tag{4.1}$$

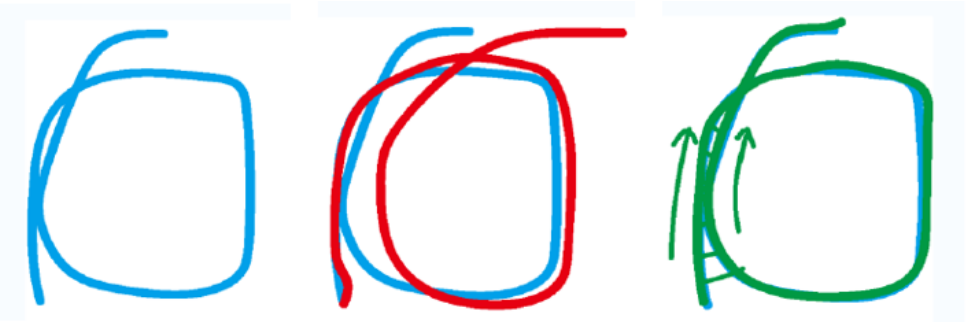
其中，ε用来表示两帧之间运动变化的大小，根据设置的阈值来判断是否将当前帧保存为关键帧。$\Delta\boldsymbol{t}$和$\Delta\boldsymbol{\theta}$分别是当前帧与上一关键帧之间的平移和旋转向量，$\lambda_1$和$\lambda_2$为对应的权重，因为旋转对于相机的运动变化比平移大，所以λ_2取值应大于λ_1，本文中$\lambda_1=0.6$，$\lambda_2=1$。

所有生成的关键帧保存在一个关键帧的库里面，用于后面的回环检测和全局优化。每个保存的关键帧都包含该帧的所有信息，在不同的关键帧之间也存在着连接关系，如果两个关键帧能够观测到共同的地图点，那么这两个关键帧就称为共视关键帧。为了避

免冗余，在优化的同时也要有条件地剔除多余的关键帧，剔除关键帧的条件是如果某个关键帧生成的地图点有 90%以上都能够被其他至少 3 个关键帧观测到，那么就舍弃该关键帧，这样能够降低系统的复杂度。

4.3 回环检测

机器人在运动的过程中，计算出来的每一帧位姿都是存在误差的，而机器人在每一帧的位姿都与前一帧相关，机器人运动时间的增加会导致系统估计位姿的误差逐渐累积，使位姿的误差越来越大，误差变大就会产生定位漂移问题，使估计的运动轨迹不准确，如图 4.1 中的 b 所示。针对这个问题，可以在系统中添加回环检测，回环是消除误差的一种方法。回环检测就是判断机器人是否在历史运动过程中到达过当前的位置，如果检测到机器人在历史运动中到达过当前的位置，就可以建立历史帧与当前帧的约束关系，虽然在视觉里程计中的位姿优化也可以减小误差，但这只是对相邻帧之间添加了约束，回环检测能够添加一些时间间隔更长的约束，使当前帧与历史帧联系起来。检测到回环之后，通过回环校正，可以将系统的误差分散在回环的各个节点处，从而调整机器人的运动轨迹和全局位姿，降低全局系统误差。



(a) 真实轨迹　　(b) 定位漂移　　(c) 添加回环检测后

图 4.1　定位漂移示意图

Fig. 4.1　Diagrampof positioning drift

回环检测中最重要的问题是怎样判断是否存在回环，这就涉及到如何对比图像的相似度。计算图像相似度最直观的方法就是特征匹配的方法，对当前帧和过去的所有关键帧都进行特征匹配，通过比较匹配的数量是否大于一定值能够判断是否发生回环，但是这种方法比较耗时，因为需要当前帧与过去所有关键帧匹配，它的运算量是 SLAM 系统承受不了的。因此本文舍弃了这种回环检测的方法，而使用词袋模型(Bag-of-Words, BoW)[59]的方法来进行回环检测。

4.3.1 词袋模型

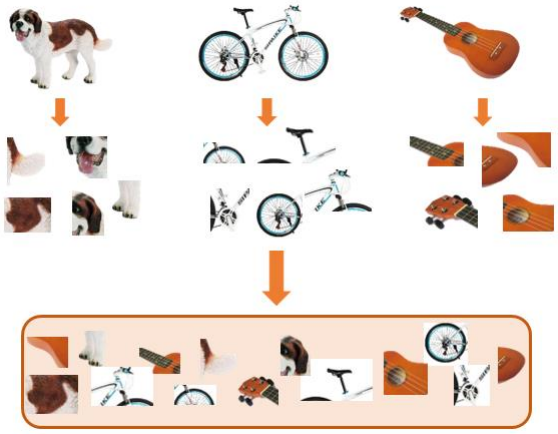


图 4.2 词袋模型

Fig.4.2 Bag-of-Words

最初的词袋模型[59]是在信息检索中应用于文本中，忽略文本中的语句和语法以及单词的顺序，用关键词的集合来表示文本，文本中每个词的出现都是独立的，不依赖于其他单词，通过对文本中单词出现的频率统计可以实现文档的匹配。词袋模型具有简单和高效的优点，现在也应用于计算机视觉领域。如图 4.2 所示，计算机视觉中的词袋模型，就是用“图像上有哪些特征”来描述一幅图像。举例来说，可以说某张图像 A 中有一个苹果，一个香蕉，另一张图像 B 中有两辆车，一个人，其中所说的“一个苹果，一个香蕉”和“两辆车，一个人”就是对两幅图像的描述，而“苹果”，“香蕉”，“车”和“人”等概念就是前面所说的词袋模型中的单词，许多单词的集合称为字典。在得到了字典之后，确定图像中有哪些单词，用单词出现的情况描述图像，将一幅图像用一个向量来描述，以前面举的例子来说，“苹果”，“香蕉”，“车”和“人”这四个单词构成了一个字典，分别用 ω_1，ω_2，ω_3，ω_4 表示，根据图像中含有的单词，图像 A 可以记为

$$A=1\cdot\omega_1+1\cdot\omega_2+0\cdot\omega_3+0\cdot\omega_4 \tag{4.1}$$

因此，只要用 $[1,1,0,0]^T$ 这个向量就可以表示图像 A，同样，可以用向量 $[0,0,2,1]^T$ 表示图像 B，如果只考虑单词是否出现而不考虑出现的数量，也可以用 $[0,0,1,1]^T$ 表示，得到两幅图像的向量表示之后，通过计算两个向量之间的距离就可以比较两幅图像的相似度。

为了通用性，字典中需要包含所有可能出现的单词，应该使用大量的数据来训练，字典的训练是一个聚类问题。图 4.3 中所示为视觉字典生成的过程，字典的生成使用经典的 K-means[60]算法。

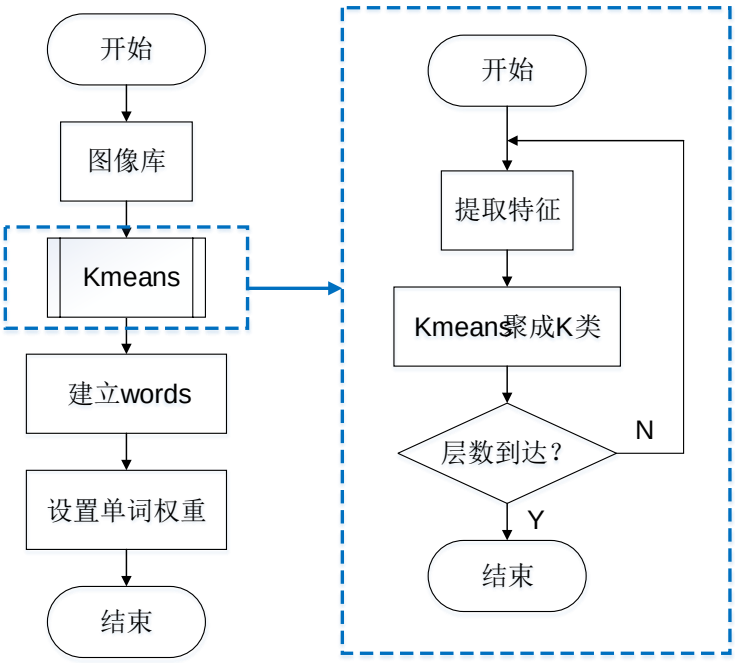


图 4.3　视觉字典生成过程

Fig. 4.3　Generation process of visual dictionary

K-means算法的原理是，将 N 个数据归为 k 个类，首先需要随机选取 k 个中心点，然后对所有样本都计算其与这 k 个中心点的距离，选择与样本距离最小的中心点作为该样本的归类。然后对每个类重新计算它的中心点，如果计算出来的中心点的变化很小，算法完成，否则重复上述步骤直到收敛。根据 K-means算法，可以把提取出来的特征点聚类成一个含有 k 个单词的字典。一般来说字典的规模都是庞大的，这就给查找字典中的单词带来了很大的难度，如果对每个单词逐一对比那么效率太低，为了提高效率，本文中使用 k 叉树表达字典，如图 4.4 所示。要构建一个深度为 d，每一层的分叉为 k 的 k 叉树，首先提取所有图像中的特征，把特征的描述子作为根节点，用 K-means算法把全部的描述子聚成 k 类，得到 k 叉树的第一层，然后再将第一层中的每个节点看成根节点，把属于该节点的样本再聚成 k 类，得到下一层，重复以上步骤，直到树的深度为 d，得到叶子层，即视觉单词。这样一个分支为 k，深度为 d 的树一共可以表示 k^d 个单词。得

到用 k 叉树表达的字典，在查找某个特征对应的单词时，只需要将特征与每一层的聚类中心比较就可以找到对应的单词，这种方法大大降低了查找的时间复杂度。

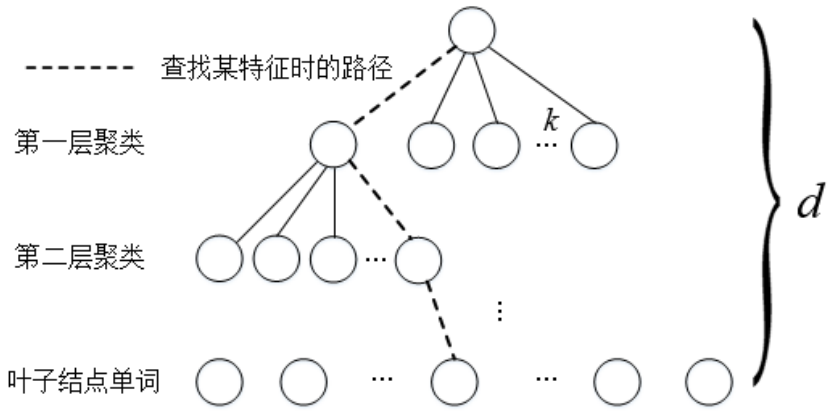


图 4.4　k 叉树字典

Fig. 4.4　k-tree dictionary

4.3.2　相似度计算

通过 K-means 算法得到了字典之后，对于任意一幅图像，都可以使用视觉向量来描述，但是应该注意的一个问题是，许多单词在区分性上的重要性是不一样的，所以需要对单词的区分性或者重要性加以衡量，为不同的单词赋予权重。在文本检索中常用的方法是 TF-IDF(Term Frequency-Inverse Document)，也叫做译频率-逆文档频率[61-62]。在视觉词袋模型中，TF 代表在一幅图像中，出现次数较多的单词对这幅图像来说很重要，所占的权重就比较高。而 IDF 表示如果包含某单词的图像越少，即单词中包含的特征数越少，该单词在图像分类时区分度越高。图像中单词的权重由 TF 和 IDF 两部分组成。例如一幅图像 P 中所有的单词总数为 m，某个单词 ω_i 在图像 P 中出现的次数为 m_i，则这个单词的 TF 值为

$$\mathrm{TF}_i = \frac{m_i}{m} \tag{4.2}$$

在建立词典时所有特征的总数量为 n，单词 ω_i 中包含的特征数量为 n_i，则对于单词 ω_i 来说，它的 IDF 值为

$$\mathrm{IDF}_i = \log\frac{n}{n_i} \tag{4.3}$$

由式(4.2)和式(4.3)可以得到单词 ω_i 的权重

$$\eta_i = \mathrm{TF}_i \times \mathrm{IDF}_i \tag{4.4}$$

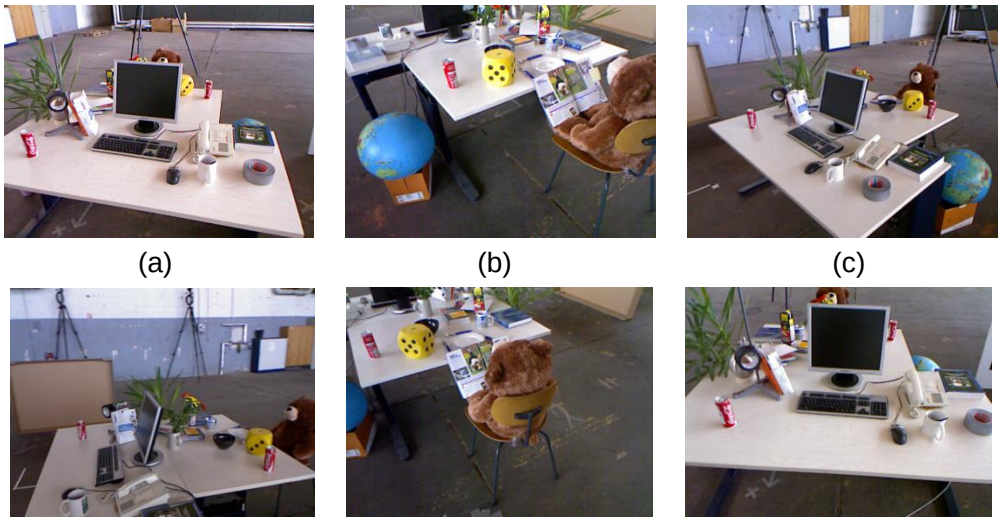
至此，对于图像 P，就可以利用词袋模型对它进行视觉向量的描述

$$P = \{(\omega_1,\eta_1),(\omega_2,\eta_2),\cdots(\omega_N,\eta_N)\} \triangleq \boldsymbol{v}_P \tag{4.5}$$

实际上，$\boldsymbol{v}_P$ 是一个稀疏向量，向量中非零的地方就是 TF-IDF 的值。至此，对于任意的两幅图像 P 和 Q，都可以计算它们之间的相似度，相似度的具体计算方法如下。

$$s(\boldsymbol{v}_P - \boldsymbol{v}_Q) = 1 - \frac{1}{N}\left\|\boldsymbol{v}_P - \boldsymbol{v}_Q\right\|_1 \tag{4.6}$$

当两个向量完全相同时，得到的相似度的值为 1，当两个向量完全相反时，相似度的值为 0。



(a) (b) (c)

(d) (e) (f)

图 4.5　实验所用 TUM 数据集图像

Fig. 4.5　Imageof TUM datasetn experiment

为了验证图像的相似度计算，本文选取了 TUM 视觉计算机组提供的数据集中的 fr2_desk数据集进行实验，该数据集中一共有 2900幅图像，利用所有图像构建一个词典，词典用 k 叉树进行存储，其中 k=10，d=5，生成的词典中单词的数量为 99566。选取数据集中的 6 张图像，如图 4.5 所示。经过观察可以看出，图 4.5 中的图像(a)和图像(f)、图像(b)和图像(e)、图像(c)和图像(d)之间比较相似，理论上来说它们之间的相似度应较

其他图像高。利用生成的词典对这 6 张图像两两之间进行相似度计算，得到的结果如表 4.1 中所示。

表 4.1　相似度计算结果

Tab. 41　Resultof similarity calculation

相似度得分	图像(a)	图像(b)	图像(c)	图像(d)	图像(e)	图像(f)
图像(a)	1	0.005345	0.022523	0.003455	0.007225	**0.092544**
图像(b)	0.005345	1	0.018846	0.008394	**0.051128**	0.006283
图像(c)	0.022523	0.018846	1	**0.043112**	0.008979	0.010700
图像(d)	0.003455	0.008394	**0.043112**	1	0.004796	0.009153
图像(e)	0.007225	**0.051128**	0.008979	0.004796	1	0.005386
图像(f)	**0.092544**	0.006283	0.010700	0.009153	0.005386	1

表格 4.1 中的结果显示，同样的图像之间的相似度得分为 1，较为相似的几组图像之间的相似度得分分别是 0.092544 0.051128 0.043112 比其他组图像之间的相似度得分高，但是就算是较为相似的两幅图像相似度得分也只有百分之几，与一般认知中两幅相似的图像相似度应该接近百分之百这种想法不同。但即使这样，相似度得分也足够用于区分图像之间的相似度，可以利用词袋模型进行回环检测。

4.3.3　回环检测与校正

虽然根据相似度得分可以判断两幅图像是否属于同一场景，但是只通过设定一个阈值就来判断是否发生回环这种做法并不可靠，因为机器人在行走过程中相邻或者距离比较近的关键帧都是比较相似的，如果相邻的关键帧被判断为回环，也会给系统增加很大的计算量，增加系统的复杂度。同样，如果在同一个回环处，那么相邻的几帧有可能都检测到同一个回环，同样会增加系统的复杂度，因此要对回环检测的关键帧进行筛选。

本文中回环检测的步骤如下。

(1) 判断距离上一次回环检测是否超过 10 帧，如果超过 10 帧就继续进行回环检测，否则不进行回环检测；

(2) 计算和当前帧有共视的地图点的关键帧与当前帧的相似度，保留最低的相似度得分 $s_{\min}$；

(3) 找出与当前帧具有公共单词但不与当前帧共视的关键帧，统计这些关键帧与当前帧具有的共同单词的最大值 $c_{\max}$；

(4) 在第 3 步的关键帧中筛选出与当前帧共有的单词数大于 $0.8\times c_{\max}$ 的关键帧，计算这些关键帧与当前帧的相似度得分，只保留得分大于 $s_{\min}$ 的关键帧；

(5) 将与第 4 步中得到的关键帧共视程度最高的前十个关键帧作为一组，计算每一组的累积得分，选出得分最高的组，将最高得分的 0.75 倍作为阈值，得分高于阈值的组，作为候选关键帧组，候选关键帧组中相似度得分最高的关键帧为候选关键帧；

(6) 为了将一些得分很高但与其他帧没有关联的候选帧去掉，检查候选关键帧的连续性，如果连续三帧都通过以上筛选，就得到了真正的回环帧，如果有多帧都通过筛选，就选择得分最高的候选关键帧组中的最高分关键帧。

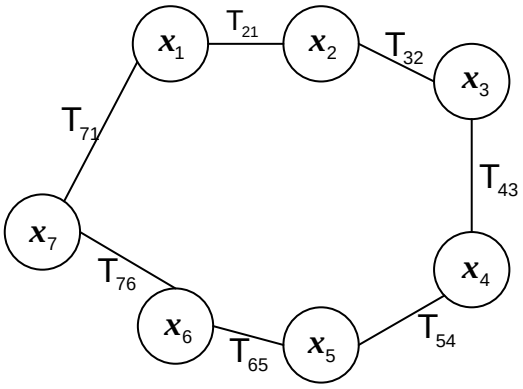


图 4.6　回环校正

Fig. 4.6　Loop closing

至此，如果没有检测到回环，就保存当前关键帧到关键帧的数据库中，如果检测到有回环，就进行回环校正消除误差。如图 4.6 所示为回环校正的示意图，$\boldsymbol{x}$ 表示机器人在某个关键帧处的位姿，矩阵 $\boldsymbol{T}$ 为位姿之间的变换矩阵，具体形式为式(4.7)所示，其中 $\boldsymbol{R}$ 和 $\boldsymbol{t}$ 分别为旋转矩阵和平移向量。

$$\boldsymbol{T}=\begin{bmatrix}\boldsymbol{R} & \boldsymbol{t}\\ 0^{\mathrm{T}} & 1\end{bmatrix} \tag{4.7}$$

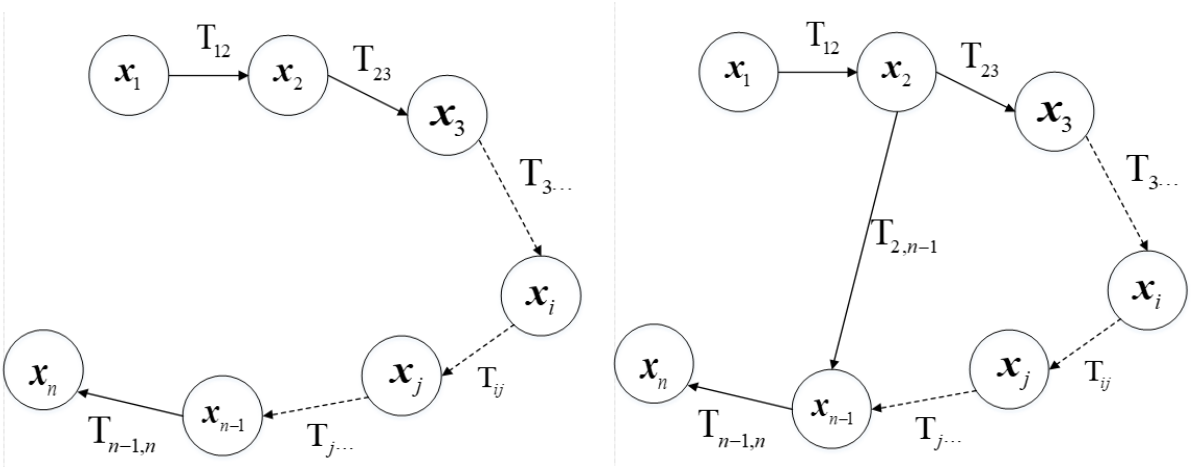
设在 i 时刻机器人的位姿为 $\boldsymbol{x}_i$，j 时刻机器人的位姿为 $\boldsymbol{x}_j$，那么两时刻之间机器人的位姿转换关系为

$$\boldsymbol{x}_i=\boldsymbol{T}_{ij}\boldsymbol{x}_j\ ,\ \boldsymbol{x}_j=\boldsymbol{T}_{ij}^{-1}\boldsymbol{x}_i \tag{4.8}$$

如图 4.6 所示，在关键帧 $\boldsymbol{x}_7$ 和 $\boldsymbol{x}_1$ 之间成功检测到了回环时，通过求解 $\boldsymbol{x}_7$ 与 $\boldsymbol{x}_1$ 之间位姿的相对变换 $\boldsymbol{T}_{71}$ ，可以得到校正后当前关键帧 $\boldsymbol{x}_7$ 的位姿 $\boldsymbol{x}_7 = \boldsymbol{T}_{71}\boldsymbol{x}_1$，此时可以利用校正后的 $\boldsymbol{x}_7$，通过位姿传播对整个回环中所有的关键帧位姿进行校正，式(4.9)为校正回环中位姿的计算方法。

$$\begin{cases} \boldsymbol{x}_6 = \boldsymbol{T}_{76}^{-1}\boldsymbol{x}_7 = \boldsymbol{T}_{76}^{-1}\boldsymbol{T}_{71}\boldsymbol{x}_1 \\ \boldsymbol{x}_5 = \boldsymbol{T}_{65}^{-1}\boldsymbol{x}_6 = \boldsymbol{T}_{65}^{-1}\boldsymbol{T}_{76}^{-1}\boldsymbol{T}_{71}\boldsymbol{x}_1 \\ \cdots\cdots \end{cases} \tag{4.9}$$

4.4 全局优化



(a) 无回环位姿图　　(b) 有回环位姿图

图 4.7　位姿图构建过程

Fig. 4.7　Construction process of pose graph

在回环之后，需要对系统进行一次完整的优化，即全局优化。在前端中，使用带有位姿和地图点的 BA 优化方法进行优化，但在后端中，随着时间的增加，机器人的轨迹和地图的规模越来越大，BA 这种优化方法的效率会越来越低。所以在后端的优化中考虑不再优化路标点的位置，而是构建一个只有轨迹的图优化，即位姿图(Pose Graph)[63-65]，如图 4.7 所示。在位姿图中，相机的位姿 $\boldsymbol{x}$ 是优化变量，为图中的节点，位姿之间的变换矩阵 $\boldsymbol{T}$ 是约束，为图中的边。如果检测到回环，就可以在不相邻的关键帧之间建立联系，在位姿图中添加更多的约束。

在位姿图中，相机在 i 时刻和 j 时刻之间的位姿变换关系为 $\boldsymbol{x}_j = \boldsymbol{T}_{ij}^{-1}\boldsymbol{x}_i$，$\boldsymbol{T}_{ij}$ 是通过前端的位姿估计或者回环检测得到的变换矩阵，实际上，从图优化的角度来看，这个位姿变换关系不是精确地成立的，所以构建误差项

$$\begin{aligned} e_{ij} &= \ln(\boldsymbol{T}_{ij}^{-1}\boldsymbol{x}_i^{-1}\boldsymbol{x}_j)^{\vee} \\ &= \ln(\exp((-\boldsymbol{\xi}_{ij})^{\wedge})\exp((-\boldsymbol{\xi}_i)^{\wedge})\exp(\boldsymbol{\xi}_j^{\wedge}))^{\vee} \end{aligned} \tag{4.10}$$

其中，符号 $\wedge$ 是一个将向量转化成反对称矩阵的符号，定义向量 $\boldsymbol{a}$

$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{4.10}$$

则有

$$\boldsymbol{a}^{\wedge} = \boldsymbol{A} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{4.11}$$

$\vee$ 为 $\wedge$ 的逆运算，即

$$\boldsymbol{A}^{\vee} = \boldsymbol{a} \tag{4.12}$$

有了误差项，就可以构建位姿图优化的目标函数，总体的目标函数为

$$\min_{\xi} \sum_{i,j\in\varepsilon} e_{ij}^{\mathrm{T}} e_{ij} \tag{4.13}$$

其中，ε 为位姿图中所有边的集合。这个最小二乘问题仍然可以用 G2O 优化库来解决。求解这个优化问题，得到优化后的位姿，就能构建准确的地图。

4.5 地图构建

SLAM 代表同时定位与建图，构建地图是 SLAM 系统的两大目标之一。SLAM 是一种底层的技术，通常用于为上层的应用提供信息。在 SLAM 系统中，根据需求的不同有不同的构建地图的方法。视觉 SLAM 中的地图大致分为稀疏地图，半稠密地图和稠密地图。稀疏地图就是机器人在行走过程中观测到的路标点的集合，根据相机采集的特征点构建稀疏的三维环境地图，这种地图在定位方面应用广泛，但是由于只用特征点构建地图，不能完整地反映环境中的三维结构，所以稀疏地图不能用于导航，避障和三维环

境重建等功能。半稠密地图能够较好地反映环境的结构，目前典型的半稠密地图的构建都是采用直接法，利用图片中的像素点进行建图。稠密地图一般为三维点云地图，利用关键帧位姿信息和彩色图与深度图等数据就能够在三维空间中构建稠密的点云环境地图，稠密地图能够完整地反映环境的结构和特点，重建环境和物体的三维结构，可以作为一个基本的可视化地图。但是在定位、导航和避障等功能方面，点云地图有明显的缺陷。第一，点云地图没有存储特征点的信息，所以不能用于本文中基于特征点的定位方法；第二，由于点云地图通常规模较大，所以需要很大的存储空间，在有限的内存中无法建模较大的环境；第三，点云不能够表示在哪一点“是否有障碍物”这一信息，而导航和避障正是需要这样的功能。因此，接下来介绍一种在导航中常用的地图形式，八叉树(Octo-map)地图[66]，它有较好的压缩性能，是一种能够随时更新的地图形式。

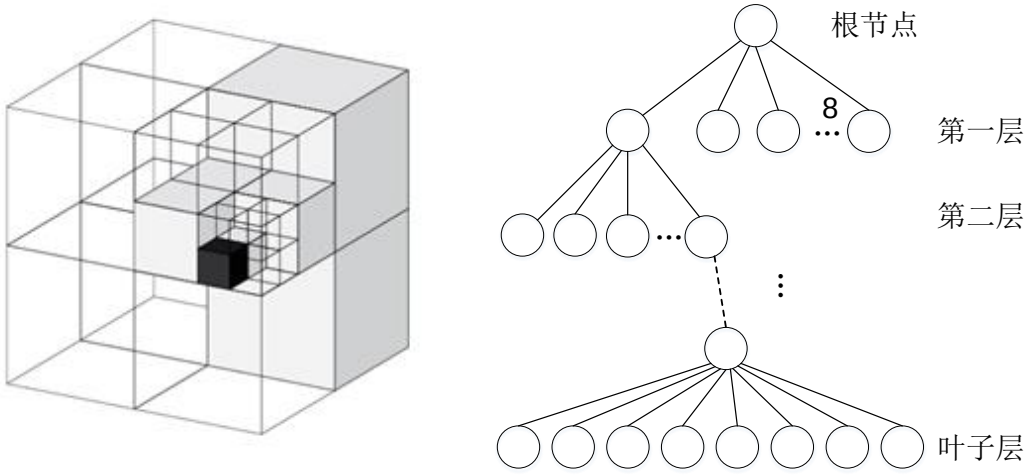


图 4.8　八叉树示意图

Fig. 4.8　Diagramof octree tree

八叉树是一种常用的数据结构，常用于描述三维空间场景。如图 4.8 所示，把左侧的大立方体看成一个空间，将这个空间作为根节点，均匀地分成八块，得到第一层的八个子空间，然后再将每个子空间均匀地分为八块，重复这个步骤直到最后得到的小方块达到最高的精度为止，将根节点和子节点连接起来就得到树状存储结构的八叉树。八叉树的叶子结点中存储了该节点是否被占据的信息，也就是对应的空间点是否有障碍物。某个叶子节点是否被占据可以用 0 和 1 来表示，但由于噪声等因素的影响，某个节点在不同时间的值可能会不同或者节点的值处于未知的状态，为了更加准确地描述，可以选择用概率的形式表示某个节点的值。x 是[0,1]之间的一个浮点数，代表概率，$y \in \square$ 为概率对数值，它们之间的变换关系可以描述为

$$y = \text{logit}(x) = \log\left(\frac{x}{1-x}\right) \tag{4.14}$$

$$x = \text{logit}^{-1}(y) = \frac{\exp(y)}{\exp(y)+1} \tag{4.15}$$

在一开始，节点的 x 取值为 0.5，y 取值为 0，如果观测到这个节点被占据，节点的 y 值增加，x 的值也相应地增加，反之则减小 y 的值。当 y 的值从 $-\infty$ 增加到 $+\infty$ 的过程中，对应的 x 的值也从 0 增加到了 1。在存储时，可以选择存储 y 的值，然后利用式(4.15)中的逆变换计算概率 x 的值，来表达某个节点是否被占据，这种方法既可以节省存储空间，也能够更新八叉树地图。

在本文中会对 SLAM 系统进行三种形式的地图构建，首先利用路标点构建稀疏的环境地图，之后对构建地图进行改进，利用 RGB-D 相机的彩色图和深度图以及关键帧的位姿信构建稠密的点云地图和八叉树地图。

4.6 本章小结

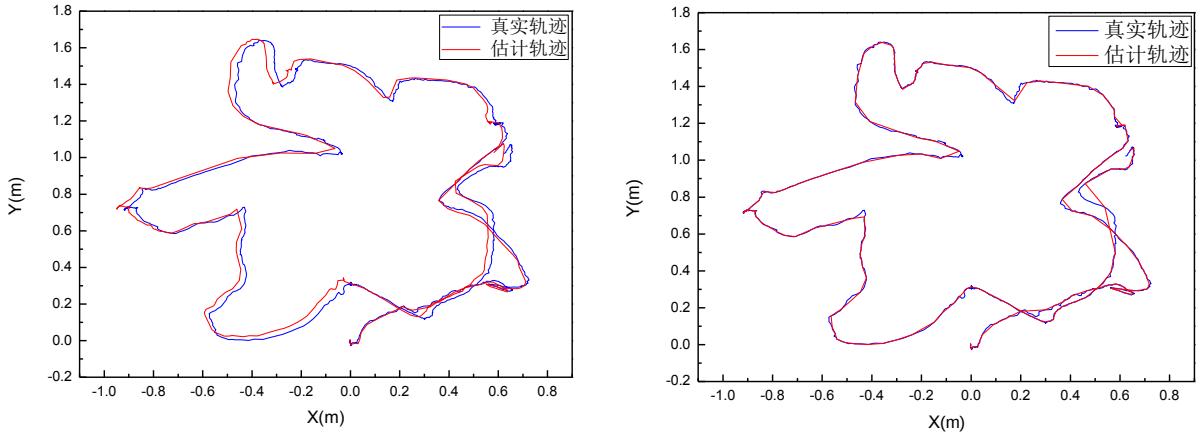
本章主要介绍了回环检测和后端优化的实现以及环境地图的构建。在机器人行进过程中，系统会采集大量图片，为了提高系统的实时性，定义了关键帧的概念，通过选取关键帧来进行回环检测和全局优化。机器人运动过程中由于位姿估计存在误差，随着时间增加，误差逐渐累积，会产生定位漂移的问题，回环是消除误差的一种方法，本文使用词袋模型对图像进行视觉向量的描述并计算图像间的相似度，利用相似度检测候选的回环帧，然后通过位姿传播进行回环校正。在回环之后，进行全局优化，之后可以利用位姿信息构建稀疏的和稠密的环境地图。

5 实验结果与分析

为了验证本文中 SLAM 系统定位与建图的效果，首先利用数据集对系统的定位效果进行评价，然后通过数据集和真实环境的实验验证系统构建地图的效果。实验中所用的数据集为 TUM 数据集，该数据集是计算机视觉组与慕尼黑工业大学提供的公开的 RGB-D 数据集。数据集中的场景是作者手持 KINECT 相机在不同的室内环境中录制的，数据集中包含场景的彩色图像和对应的深度图像以及相机真实的运动轨迹，通过系统估计的轨迹与真实轨迹的对比以及两轨迹之间的误差可以评价系统的位姿跟踪效果。

5.1 剔除误匹配实验

为了验证本文中剔除误匹配的方法与传统的剔除误匹配在 SLAM 系统中的效果，利用 TUM 数据集中的序列 fr1_plant和 fr2_desk进行实验。序列 fr1_plant的内容是手持相机从不同位置进行 360° 环绕拍摄的一盆植物，序列的时长为 41.53s 轨迹长度为 14.795m。序列 fr2_desk是一个典型的办公室的场景，包括办公桌，电脑，键盘，椅子和电话等，该序列时长为 99.36s 轨迹长度为 18.880m。将具有两种方法的 SLAM 系统分别在以上两个序列上进行实验，实验结果如图 5.1 和图 5.2 所示。



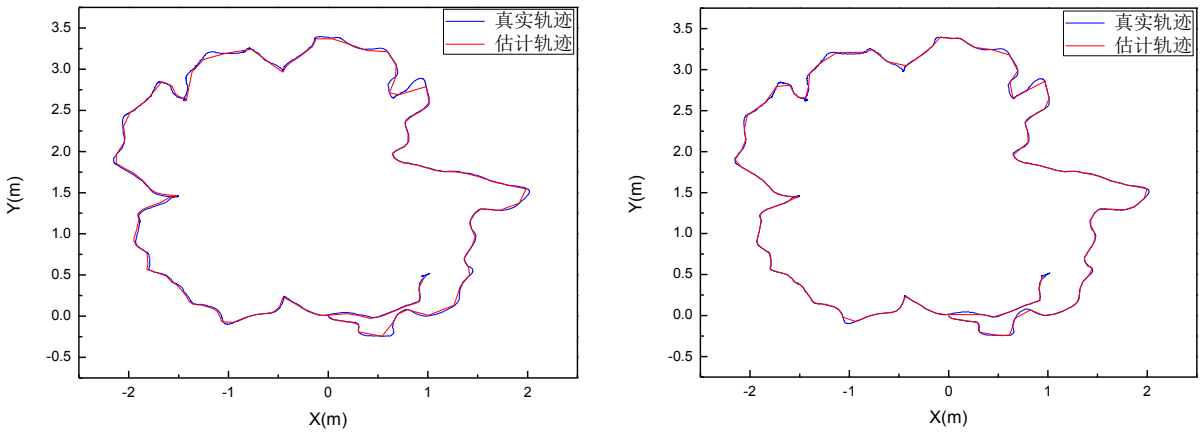
(a) 传统方法　　(b) 本文方法

图 5.1 fr1_plant数据集实验结果对比

Fig. 5.1 Comparisonof experimental resultof dataset fr1_plant

在图 5.1 和图 5.2 中，左侧的结果图为使用传统的剔除误匹配方法得到的相机估计轨迹与真实轨迹的对比，右侧的结果图为使用本文的剔除误匹配的方法得到的相机估计轨迹与真实轨迹的对比，图中所有蓝色的轨迹为相机的真实轨迹，红色曲线为系统估计

轨迹。从轨迹对比的结果图中能够看出，在对剔除误匹配的方法进行改进后，系统的定位精度明显增强，系统的位姿跟踪效果较改进之前更好。



(a) 传统方法　　(b) 本文方法

图 5.2　fr2_desk 数据集实验结果对比

Fig. 5.2　Comparison of experimental result of dataset fr2_desk

以上的实验结果能够直接通过轨迹判断出改进前后跟踪效果的不同，但还需要用更加客观的方法评价系统的位姿跟踪效果，本文中采用轨迹绝对误差(Absolute Trajectory Error，ATE)的均方根误差(Root Mean Square Error，RMSE)作为评价系统位姿跟踪效果的一个量化指标。相机的真实运动轨迹为 $X=\{X_1, X_2, \cdots X_n\}$，系统估计的相机运动轨迹为 $\hat{X}=\{\hat{X}_1, \hat{X}_2, \cdots \hat{X}_n\}$，则有：

$$ATE_{RMSE}(\hat{X}, X)=\sqrt{\frac{1}{n}\sum_{i=1}^{n}[tran(\hat{X}_i)-tran(X_i)]^2} \tag{5.1}$$

其中，tran 表示位姿的平移向量。表 5.1 中为改进前后的均方根误差，由表 5.1 的可见，本文中剔除误匹配的方法得到的轨迹误差比传统方法得到的轨迹误差小。

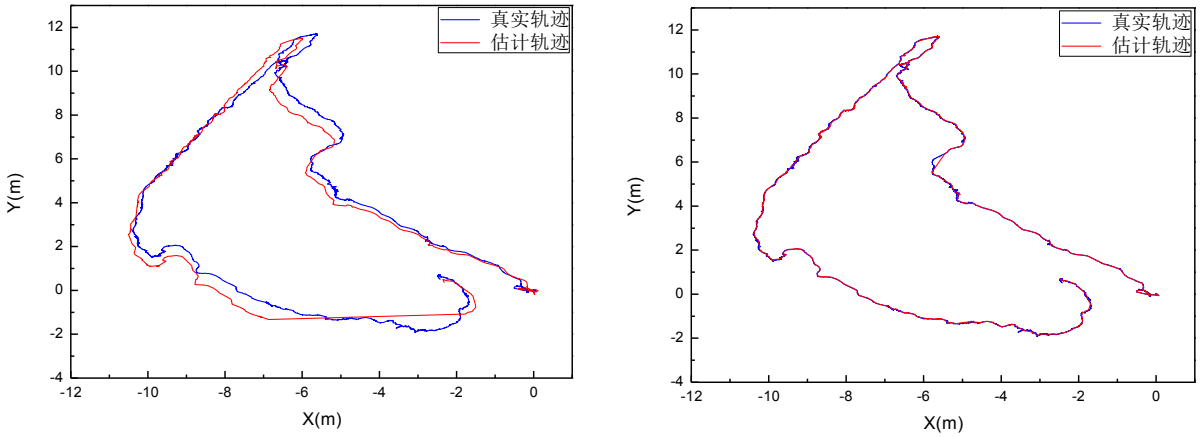
表 5.1　相机轨迹误差

Tab.5.1　RMSE of camera trajectory

数据集	传统方法 RMSE(m)	本文方法 RMSE(m)
fr1_	0.02	0.01

plant	1	4
fr2_	0.01	0.00
desk	1	9

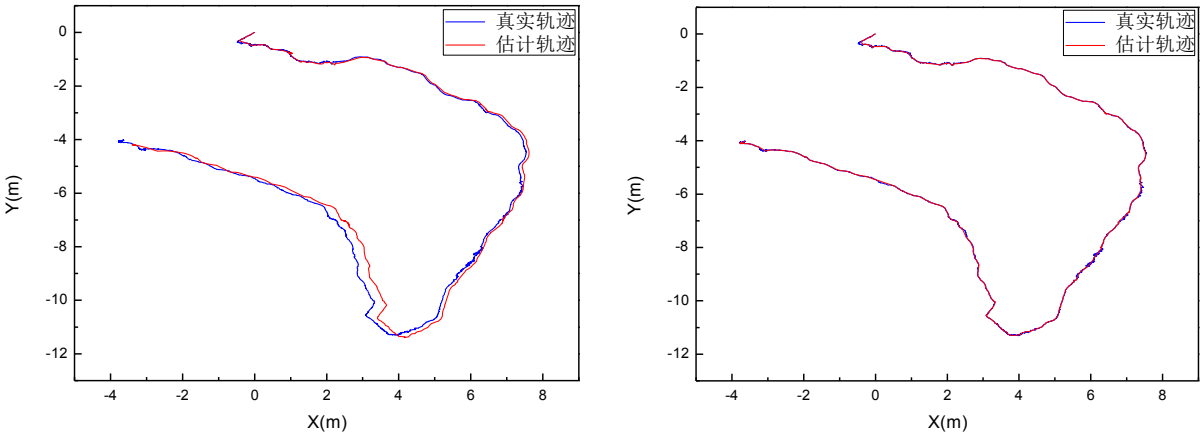
5.2 机器人位姿跟踪与全局轨迹



(a) 全局优化前　　(b) 全局优化后

图 5.3 fr2_large_with_loop数据集实验结果对比

Fig. 5.3 Comparison of experimental result dataset fr2_large_with_loop



(a) 全局优化前　　(b) 全局优化后

图 5.4 fr2_large_no_loop数据集实验结果对比

Fig. 5.4 Comparison of experimental result dataset fr2_large_no_loop

机器人的位姿估计实验中使用的数据集为 TUM 数据集中的 fr2_large_with_loop和 fr2_large_no_loop两个序列。序列 fr2__large_with_loop是一个场景为工厂车间的长序列，在轨迹的末端存在回环，序列总时长为 173.19s，轨迹长度为 39.111m。序列

fr2_large_no_loop的场景与 frr2__large_with_loop相同，相机的轨迹也与其类似，不同的是序列 fr2_large_no_loop中不存在回环，该序列的时长为 112.37s，轨迹长度为 26.086m。利用这两个数据集对全局优化前和优化后的 SLAM 系统进行测试，然后将系统估计的相机运动轨迹与真实运动轨迹进行对比，实验结果如图 5.3 和图 5.4 所示。在图 5.3 和图 5.4 中，左侧的结果图为全局优化前系统的估计轨迹与相机真实运动轨迹的对比，右侧的结果图为全局优化后系统的估计轨迹与真实轨迹的对比，图中所有蓝色的曲线为相机的真实轨迹，红色曲线为系统估计轨迹。从轨迹对比的结果图中能够看出，在经过系统后端的全局优化之后，系统估计轨迹与真实轨迹重合度变高，系统的定位精度增强。

在以上两个数据集上实验得到的全局优化前后轨迹均方根误差结果如表 5.2 所示。从表 5.2 中也可以看出，全局优化后相机轨迹的误差明显小于优化之前相机轨迹的误差。

表 5.2　全局优化前后相机轨迹误差

Tab.5.2　RMSE of camera trajectory before and after global optimization

数据集	全局优化前 RMSE(m)	全局优化后 RMSE(m)
fr2_large_with_loop	0.181	0.134
fr2_large_no_loop	0.185	0.163

5.3　系统性能分析

为了验证本文 SLAM 系统的有效性，本文选择三个基于 RGB-D 相机的 SLAM 系统进行对比实验，分别为文献[20]中的 Kintinous、文献[21]中的 ElasticFusion和文献[26]中的 RGBD-SLAM 方法，实验的数据为 TUM 数据集中的序列 fr1_plant、fr2_desk、fr2_large_with_loop、fr1_room和 fr3_office。其中序列 fr1_room是一个房间内的场景，序列的时长是 48.90s，轨迹长度为 15.989m。序列 fr3_office是一个办公室内的场景，场景中存在大的回环，该序列的时长为 87.09s，轨迹长度为 21.455m。实验结果如表 5.3 中所示。表格 5.3 中的实验结果为四个 SLAM 系统在所有实验的场景序列中得到的轨迹误差，实验结果表明，在序列 fr1_plant、fr2_desk、fr1_room和 fr3_office中本文方法的轨迹误差 RMSE 在四种 SLAM 方法中是最小的，而序列 fr2_large_with_loop中本文方法的轨迹误差比 ElasticFusion大 0.009m，比 Kintinous小 0.037m。

表 5.3　绝对轨迹误差 RMSE对比

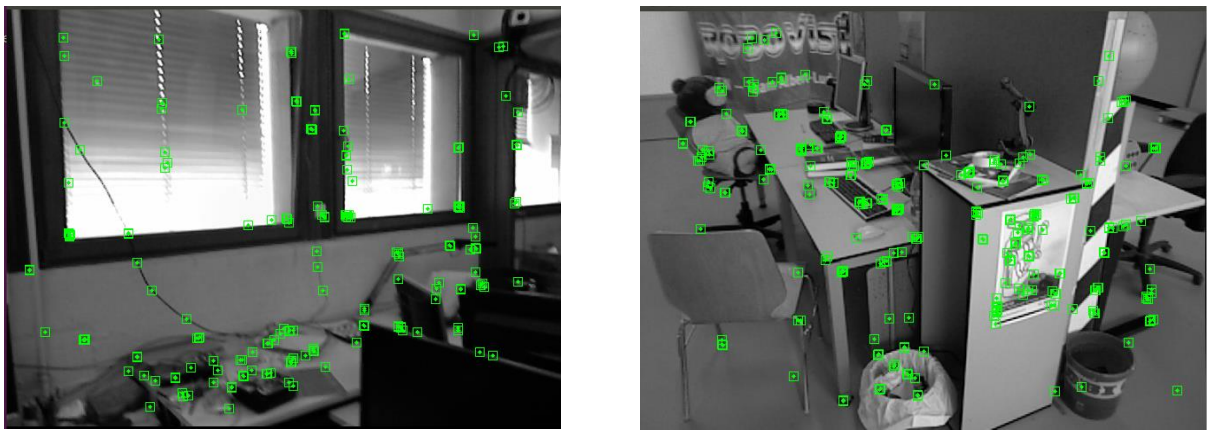
Tab.5.3 Comparison of RMSE

数据集	本文方法(m)	Kintinous(m)	ElasticFusion(m)	RGBD-SLAM(m)
fr1_plant	**0.014**	0.038	0.026	0.032
fr2_desk	**0.009**	0.034	0.071	0.057
fr2_large_with_loop	0.134	0.171	**0.125**	-
fr1_room	**0.047**	0.075	0.068	0.087
fr3_office	**0.010**	0.030	0.017	-

5.4 系统地图构建

5.4.1 基于数据集的实验

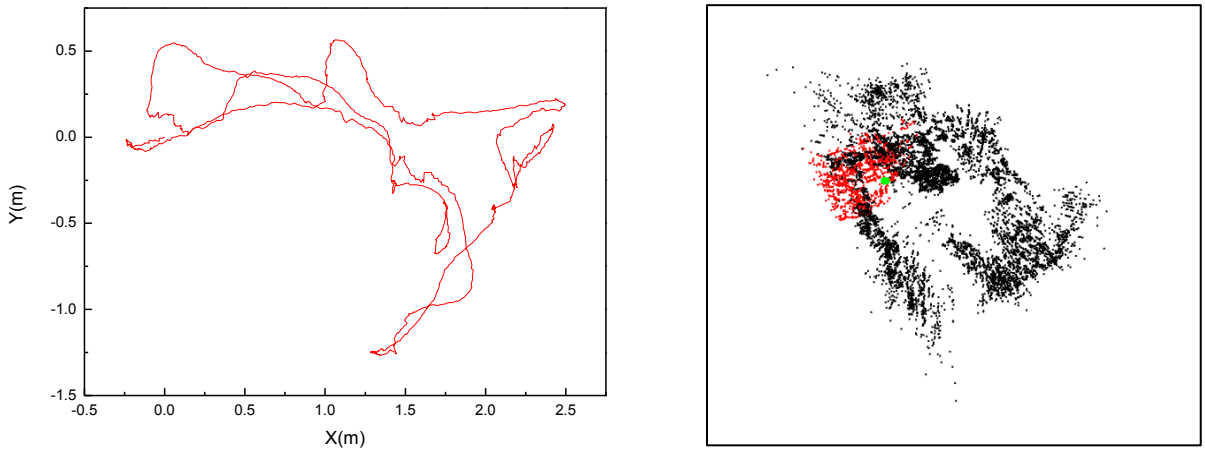
在本文的 SLAM 框架下，为了验证系统的有效性和实时性，首先利用数据集进行同时定位与地图构建实验。实验中所用数据为 TUM 数据集中的序列 fr1_room 和序列 fr3_office。这两个序列的实验环境如图 5.5 所示，图中的绿色方框为实验过程中提取的当前帧的特征。



(a) 序列 fr1_room场景　　(b) 序列 fr3_office 场景

图 5.5 数据集实验场景

Fig. 5.5 Experimental scene of dataset



(a) 相机运动轨迹　　(b) 稀疏点云地图

图 5.6　fr1_room数据集实验结果

Fig. 5.6　Experimental result of dataset fr1_room

图 5.6 为在数据集 fr1_room中得到的相机运动轨迹和实时的稀疏点云地图，稀疏的三维点云地图能够节省空间，有利于系统的实时性。构成地图的点为相机运动过程中提取到的特征点，相机的运动轨迹和地图都是实时且不断更新的，验证了本文中的 SLAM 系统满足实时性的要求。从图中可以看出，虽然地图是稀疏的，但房间的整体轮廓是比较清晰的，也能够看出物体的轮廓和位置。

图 5.7 为在数据集 fr1_room上进行实验构建的稠密的点云地图，如图所示，稠密的点云图可以反映出房间的环境结构，对环境和物体进行三维重建，从图中可以看出场景中的窗户，桌子和电脑等物体都是很清晰的，说明可以将其作为一个可视化地图。但是由于稠密的三维点云地图信息量很大，所以在经过一段时间之后可能会出现卡顿的情况。

图 5.7　数据集 fr1_room稠密点云地图

Fig. 5.7　Dense point cloud map of datasefr1_room

为了提高系统的实用性，还需要构建能够用于导航和避障的八叉树地图，图 5.8 为利用数据集 fr1_room构建的八叉树地图，地图的分辨率为 0.05m。如图所示，在有障碍物的地方节点都被占据，因此该地图可以用于导航和避障。

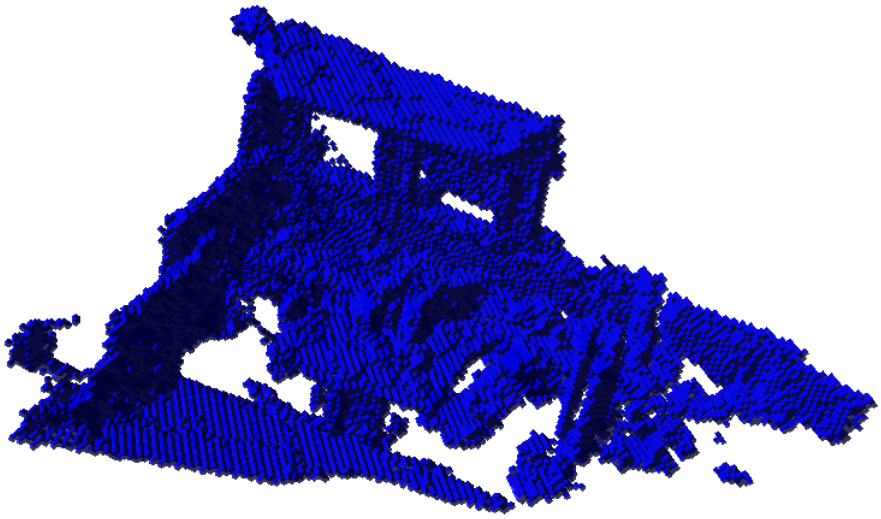
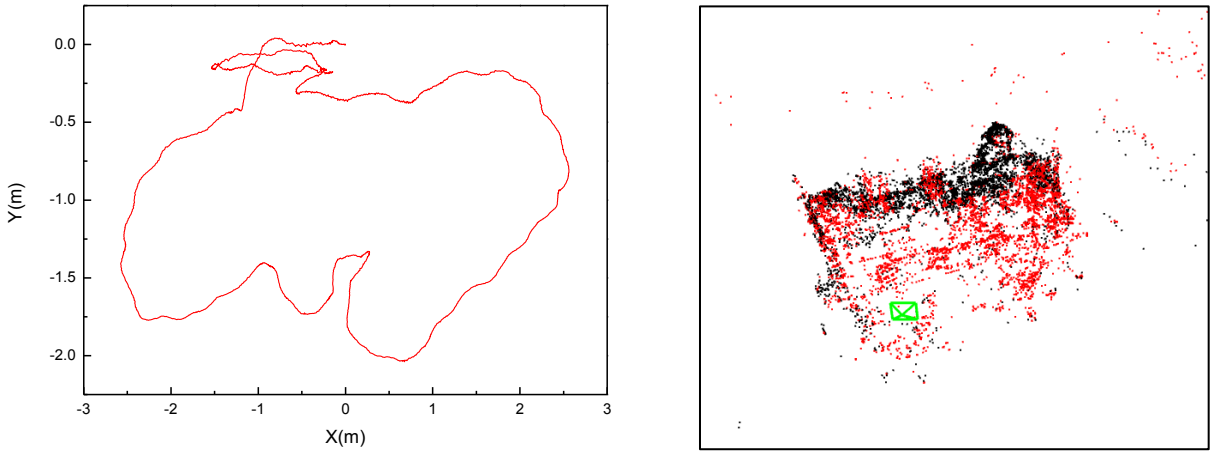


图 5.8　数据集 fr1_room八叉树地图

Fig. 5.8　Octreemapof datasefr1_room

图 5.9 为在数据集 fr3_office 上进行实验得到的相机运动轨迹和稀疏地图，在系统运行过程中，地图能够实时更新。从稀疏地图中可以看出房间内桌子的轮廓，办公桌一边的小熊玩偶的轮廓也很明显。



(a) 相机运动轨迹　　(b) 稀疏点云地图

图 5.9　fr3_office数据集实验结果

Fig. 5.9　Experimental resulof datasefr3_office

图 5.10和图 5.11分别是利用数据集 fr1_room得到的稠密点云地图和八叉树地图，从稠密的点云图中能够清楚地看出房间内的结构，办公桌以及旁边的小熊等物品都很清晰明显，证明系统对这个场景实现了准确的三维重建。图 5.11 中的八叉树地图与图 5.10 中的点云图相对应，在有物体的地方节点都被占据，可用于导航和避障。

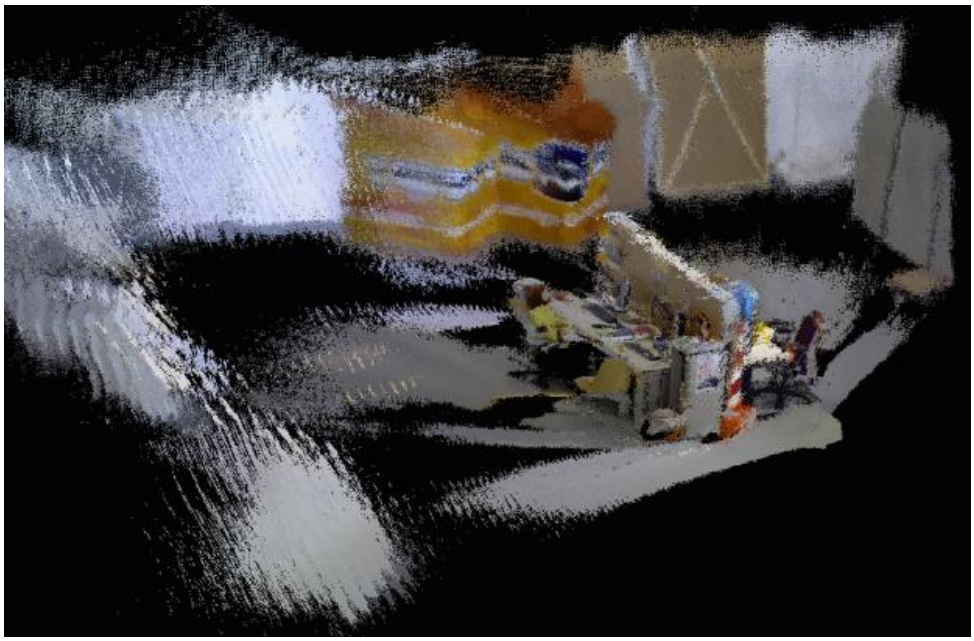


图 5.10　数据集 fr3_office 稠密点云地图

Fig. 5.10　Dense point cloud map of dataset fr3_office

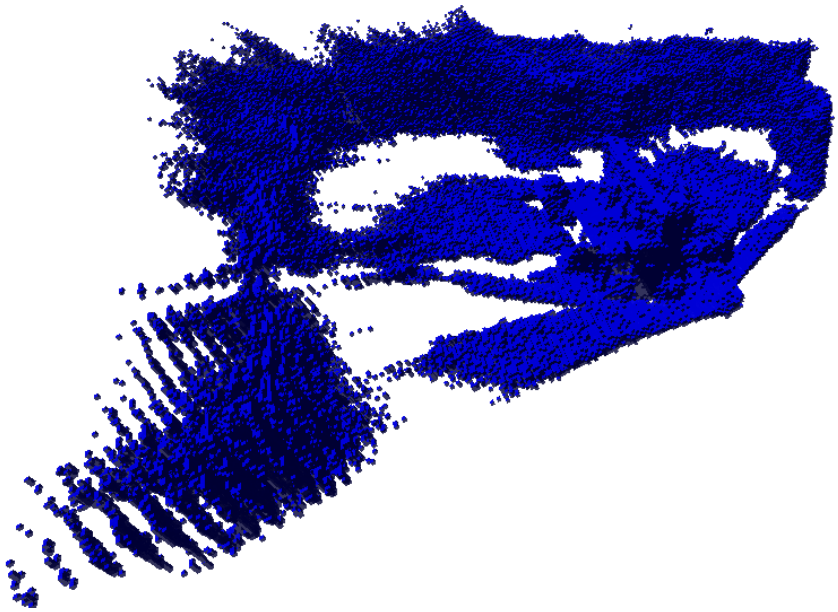


图 5.11　数据集 fr3_office 八叉树地图

Fig. 5.11　Octreemap of dataset fr3_office

5.4.2　真实环境实验

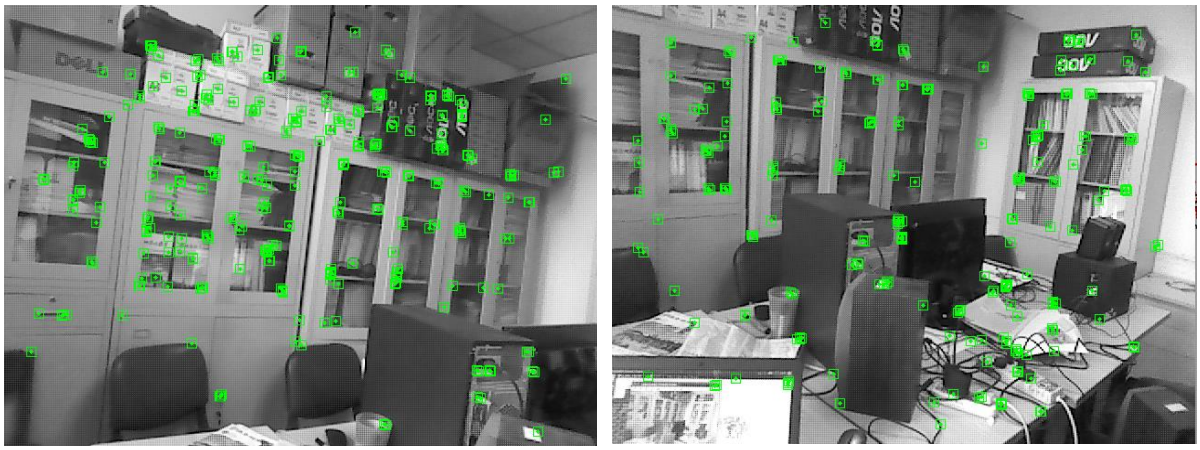


图 5.12　真实环境实验场景

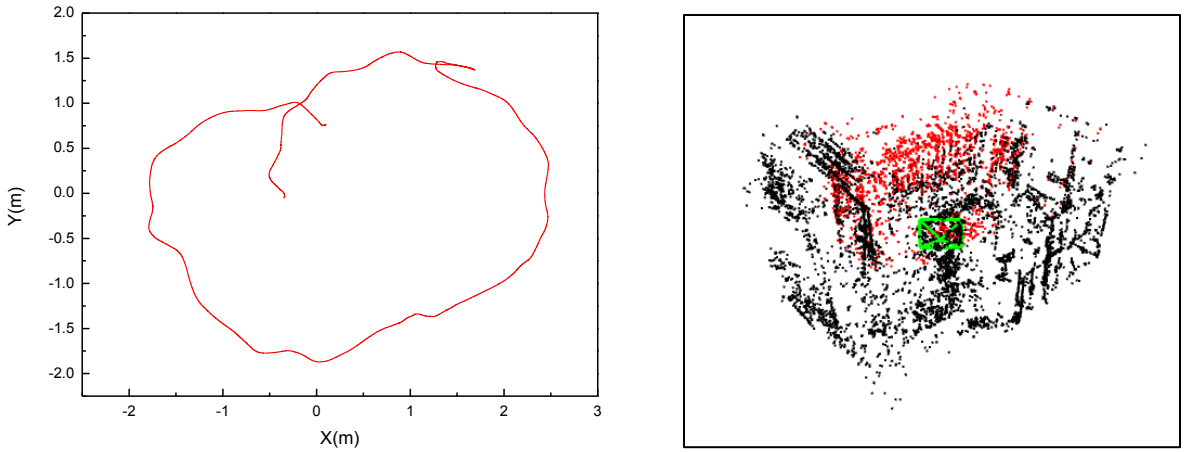
Fig. 5.12　Experimental scene of real environment

通过在数据集上进行实验得到的实验结果能够验证系统构建地图的效果，为了进一步验证系统的有效性，本文也在室内真实环境中进行了实验。真实环境的实验场景为创新园大厦的 A0521房间，房间内有一张长方形桌子，一侧墙壁处为紧密排列的书柜，实验场景如图 5.12 所示。图 5.12 为实验过程中采集到的图像的灰度图，绿色的方框为在该帧中提取到的特征点。

本文中的实验使用了机器人操作系统 ROS(Robot Operating System)，ROS是一个机器人软件平台，其代码为开源的，该平台可以兼容多种编程语言，包含许多实用性的工具，能够解决软件开发过程中移植性差的问题。ROS操作系统有多个版本，本文中使用的版本为 Indigo版本，系统的运行环境为 Ubuntu16.04，计算机的硬件配置为 Intel 的酷睿 i5 4200U处理器，系统内存为 4GB，笔记本的型号为 Dell Inspiron15-3537。将 KINECT 相机与笔记本电脑连接，手持相机在房间内绕着桌子一周匀速运动，利用相机采集到的室内的环境信息构建地图。

图 5.13 所示为在房间 A0521 进行实验得到的实验结果，左图为系统估计的相机运动轨迹，右图为实验过程中实时更新的稀疏点云地图，从稀疏地图中可以看出，房间的轮廓比较清晰，地图的一致性比较好，地图能够实时更新，证明本文中的 SLAM 系统在真实环境中也可以满足实时性的要求。图 5.14 所示为房间 A0521 的稠密点云地图，稠密的点云地图较好地重建了房间的内部环境，从地图中可以清楚地看出房间的布局和物体的形状轮廓等，说明本文中的 SLAM 系统构建的稠密点云地图能够准确地描述室内的场景，证明了系统的有效性和实用性。图 5.15 为房间 A0521 的八叉树地图，地图的分

辨率为 0.05m，可以看出八叉树地图与稠密的点云地图相互对应，可用作导航和避障等功能。



(a) 相机运动轨迹　　(b) 稀疏点云地图

图 5.13　真实环境实验结果

Fig. 5.13　Experimental result of real environment

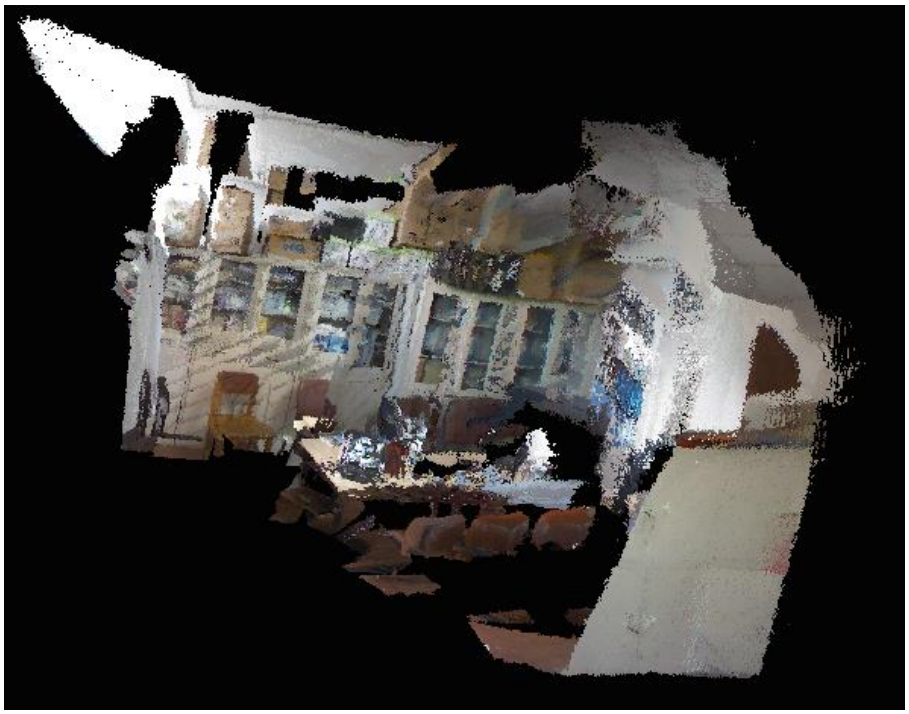


图 5.14　真实环境的稠密点云地图

Fig.5.14　Dense point cloud map of real environment

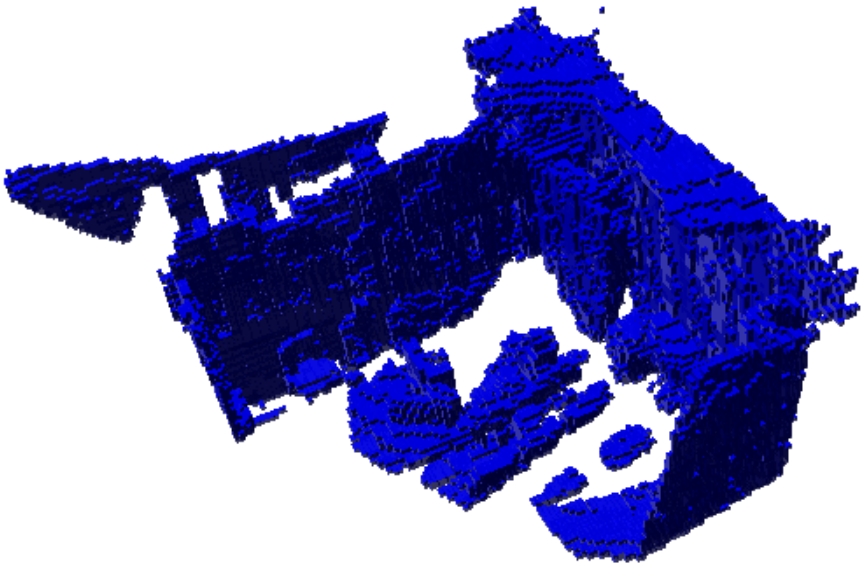


图 5.15　真实环境的八叉树地图
Fig.5.15　Octree map ofeal environment

5.5　实验结果分析

综合以上所有的实验，在剔除误匹配的实验中，本文方法得到的相机轨迹误差比传统方法得到的轨迹误差更小，可以证明本文剔除误匹配的方法优于传统的剔除误匹配的方法。在机器人的位姿跟踪实验结果中，在全局优化后系统得到的相机轨迹误差比全局优化之前更小，验证了系统的回环检测和全局优化的有效性。在验证系统性能的实验中，本文方法在数据集的场景序列中实验得到的相机轨迹误差较另外三种方法更小，说明本文方法的精度比另外三种基于 RGB-D 相机的 SLAM 方法更高，证明了本文方法的有效性。在系统地图构建实验中，不论是数据集中的场景还是真实的环境，本文的 SLAM 系统都可以进行实时的地图构建，并且可以重建环境的三维结构。

5.6　本章小结

为了验证本文的 SLAM 系统的有效性，本章利用数据集和真实环境进行了实验。首先通过 TUM 数据集中的序列 fr1_plant 和 fr2_desk 对本文剔除误匹配的方法进行验证，为了对系统的位姿跟踪效果以及全局优化效果进行评价，利用数据集中的序列 fr2_large_with_loop 和 fr2_large_no_loop 进行实验，对全局优化前后系统估计的相机运动轨迹和真实的轨迹分别进行对比，并计算全局优化前后轨迹的均方根误差，实验结果表明，在经过系统后端的全局优化之后，系统的定位精度增强，位姿跟踪效果较好。之

后利用序列之后利用序列 fr1_plant、fr2_desk、fr2_large_with_loop、fr1_room 和 fr3_office 对系统性能进行分析，将本文方法与 Kintinous、ElasticFusion 和 RGBD-SLAM 方法进行对比，结果表明本文方法的精度高于其它三种方法。最后利用序列 fr1_room 和 fr3_office 以及真实的室内环境对系统构建地图的效果进行评价。实验表明，无论是数据集还是真实环境，系统都能实时构建地图，其中稀疏地图能够用于定位，稠密的点云地图可以用于三维重建，八叉树地图可以用于导航和避障等功能。所有的实验结果验证了本文 SLAM 系统的有效性。

结　论

本文研究了基于 RGB-D 相机的视觉 SLAM 系统，SLAM 即同时定位与构建地图，它是一种能够为机器人提供定位和导航功能的底层系统，系统中包括前端位姿计算，回环检测，后端优化和构建地图这四部分。前端的功能是采集外部环境信息和计算机器人的位姿，回环检测的功能是解决误差累积带来的定位漂移问题，回环检测之后进行全局优化，然后构建机器人的轨迹和环境地图。

本文中 RGB-D 相机的选择为微软公司的 KINECT 相机一代。KINECT 相机既能获得场景的彩色图像，也能通过发射和接收红外光得到场景的深度信息，有利于系统的实时性。为了机器人位姿估计的准确性，需要对相机进行标定，本文棋盘格标定法对 KINECT 相机进行了标定，得到了相机的内参和外参。

系统前端视觉里程计的实现依赖于特征点法，本文中图像特征的选择为 ORB 特征，通过图像金字塔和灰度质心法为 ORB 特征添加了尺度不变性和旋转不变性，然后使用汉明距离作为度量进行特征匹配，通过设置距离阈值和利用次优匹配来剔除结果中的一些误匹配，之后利用 EPnP 的方法估计相邻两帧之间相机的运动，用图优化的方法对估计的位姿进行优化。

系统的回环检测部分采用了词袋模型的方法，首先定义关键帧的概念，然后利用词袋模型对图像进行描述，将图像转化为视觉向量，之后可以通过计算两个视觉向量之间的距离来度量图像间的相似性，确定回环候选帧，进行位姿校正。在回环之后，建立位姿图，进行全局位姿优化，然后构建轨迹和地图。

为了验证本文 SLAM 系统的有效性，分别利用数据集和真实环境进行了实验。数据集的选择为慕尼黑工业大学提供的 TUM 数据集，真实环境的实验场景为创新园大厦的 A0521 房间。从实验结果可以看出，(1)本文剔除误匹配的方法优于传统方法；(2)在全局优化后系统的定位精度和位姿跟踪效果有所提高，估计的相机轨迹与真实轨迹之间的误差较小；(3)与其它基于 RGB-D 相机的 SLAM 方法对比，结果表明本文方法的精度优于其它方法；(4)系统能够实现实时构建地图，其中稀疏地图可以用于定位，稠密的点云地图可以用于三维重建，八叉树地图可以用于导航和避障等功能。

综合所述，本文对基于 RGB-D 相机的视觉 SLAM 系统进行了研究，在之后的研究中，考虑进一步研究语义 SLAM，这就需要 SLAM 技术与深度学习相结合来处理图像，也是一个很有前景的研究方向。

参 考 文 献

[1] Dissanayake M W M G, Newman P, Clark S, et al. A solution to the simultaneous localization and map building (SLAM) problem[J]. IEEE Transactions on robotics and automation, 2001, 17(3): 229-241.

[2] Smith R C, Cheeseman P. On the representation and estimation of spatial uncertainty[J]. The international journal of Robotics Research, 1986, 5(4): 56-68.

[3] Teslić L, Škrjanc I, Klančar G. EKF-based localization of a wheeled mobile robot in structured environments[J]. Journal of Intelligent & Robotic Systems, 2011, 62(2): 187-203.

[4] Oh T, Lee D, Kim H, et al. Graph structure-based simultaneous localization and mapping using a hybrid method of 2D laser scan and monocular camera image in environments with laser scan ambiguity[J]. Sensors, 2015, 15(7): 15830-15852.

[5] Liu L, Chen H, Chu S, et al. The method of coordinate recognition for maize straws cantering by monocular vision[C]. 2016 2nd International Conference on Control, Automation and Robotics (ICCAR). IEEE, Hong Kong, China, 2016: 304-307.

[6] Liu M, Siegwart R. Topological mapping and scene recognition with lightweight color descriptors for an omnidirectional camera[J]. IEEE Transactions on Robotics, 2014, 30(2): 310-324.

[7] Kwon H, Yousef K M A, Kak A C. Building 3D visual maps of interior space with a new hierarchical sensor fusion architecture[J]. Robotics and Autonomous Systems, 2013, 61(8): 749-767.

[8] Zhang H, Liu Y, Tan J. Loop closing detection in RGB-D SLAM combining appearance and geometric constraints[J]. Sensors, 2015, 15(6): 14639-14660.

[9] Steinbrücker F, Sturm J, Cremers D. Real-time visual odometry from dense RGB-D images[C]. 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops). IEEE, Barcelona, Spain, 2011: 719-722.

[10] Whelan T, Johannsson H, Kaess M, et al. Robust real-time visual odometry for dense RGB-D mapping[J]. 2013.

[11] Endres F, Hess J, Engelhard N, et al. An evaluation of the RGB-D SLAM system[C]. 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 2012, 3(c): 1691-1696.

[12] Smith R C, Cheeseman P. On the representation and estimation of spatial uncertainty[J]. The international journal of Robotics Research, 1986, 5(4): 56-68.

[13] Lu F, Milios E. Globally consistent range scan alignment for environment mapping[J]. Autonomous robots, 1997, 4(4): 333-349.

[14] Thrun S, Montemerlo M. The graph SLAM algorithm with applications to large-scale mapping of urban structures[J]. The International Journal of Robotics Research, 2006, 25(5-6): 403-429.

[15] Davison A J, Reid I D, Molton N D, et al. MonoSLAM: Real-time single camera SLAM[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2007 (6): 1052-1067.

[16] Klein G, Murray D. Parallel tracking and mapping for small AR workspaces[C]. 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, IEEE, Nara, Japan, 2007: 225-234.

[17] Henry P, Krainin M, Herbst E, et al. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments[J]. The International Journal of Robotics Research, 2012, 31(5): 647-663.

[18] Endres F, Hess J, Engelhard N, et al. An evaluation of the RGB-D SLAM system[C]. 2012 IEEE International Conference on Robotics and Automation. IEEE, Saint Paul, MN, USA, 2012, 3(c): 1691-1696.

[19] Newcombe R A, Izadi S, Hilliges O, et al. Kinectfusion: Real-time dense surface mapping and tracking[C]. The 10th IEEE International Symposium on Mixed and Augmented Reality. Basel, Switzerland, 2011, 11(2011): 127-136.

[20] 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.

[21] 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.

[22] Forster C, Pizzoli M, Scaramuzza D. SVO: Fast semi-direct monocular visual odometry[C]. 2014 IEEE international conference on robotics and automation (ICRA). IEEE, Hong Kong, China, 2014: 15-22.

[23] Forster C, Zhang Z, Gassner M, et al. SVO: Semidirect visual odometry for monocular and multicamera systems[J]. IEEE Transactions on Robotics, 2017, 33(2): 249-265.

[24] Engel J, Schöps T, Cremers D. LSD-SLAM: Large-scale direct monocular SLAM[C]. European conference on computer vision. Springer, Cham, Zurich, Switzerland, 2014: 834-849.

[25] Labbe M, Michaud F. Appearance-based loop closure detection for online large-scale and long-term operation[J]. IEEE Transactions on Robotics, 2013, 29(3): 734-745.

[26] Endres F, Hess J, Sturm J, et al. 3-D mapping with an RGB-D camera[J]. IEEE transactions on robotics, 2014, 30(1): 177-187.

[27] 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.

[28] 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.

[29] Ullah S, Song B, Chen W. EMoVI-SLAM: Embedded Monocular Visual Inertial SLAM with Scale Update for Large Scale Mapping and Localization[C]. The 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN). IEEE, Nanjing, China, 2018: 880-885.

[30] Hu G, Huang S, Zhao L, et al. A robust rgb-d slam algorithm[C].IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, Vilamoura, Portugal, 2012: 1714-1719.

[31] 杨东方,王仕成,刘华平,等.基于 Kinect 系统的场景建模与机器人自主导航[J].机器人,2012 (2012 年 05): 581-589.

[32] Li M, Lin R, Wang H, et al. An efficient SLAM system only using RGBD sensors[C]. 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO). IEEE, Shenzhen, China, 2013: 1653-1658.

[33] 付梦印,吕宪伟,刘彤,等.基于 RGB-D 数据的实时 SLAM 算法[J]. 机器人,2015, 37(6): 683-692.
[34] Min H S, Yang J. Research of improved RGB-D SLAM algorithm fusing IMU[J]. Computer Engineering and Design, 2015: 6-15.
[35] Shen X, Min H, Lin Y. Fast RGB-D ICP with bionic vision depth perception model[C]. 2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER). IEEE, Shenyang, China, 2015: 1241-1246.
[36] Gao X, Zhang T. Robust RGB-D simultaneous localization and mapping using planar point features[J]. Robotics and Autonomous Systems, 2015, 72: 1-14.
[37] Xin G, Zhang X, Wang X, et al. A RGBD SLAM algorithm combining ORB with PROSAC for indoor mobile robot[C]. 2015 4th International Conference on Computer Science and Network Technology (ICCSNT). IEEE, Harbin, China, 2015, 1: 71-74.
[38] Cheng Z, Wang G. Real-Time RGB-D SLAM with Points and Lines[C]. 2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC). IEEE, Xi'an, China, 2018: 1119-1122.
[39] Han L, Xu L, Bobkov D, et al. Real-Time Global Registration for Globally Consistent RGB-D SLAM[J]. IEEE Transactions on Robotics, 2019, 35(2): 498-508.
[40] 徐冬云.基于 Kinect 的视觉同步定位与建图研究[D]:(硕士学位论文). 哈尔滨：哈尔滨工业大学, 2016.
[41] Weng J . Camera calibration with distortion models and accuracy evaluation[J]. IEEE Trans. on Pattern Analysis and Machine Intelligence, 1992, 14(10): 965-980.
[42] Zhang Z. Flexible camera calibration by viewing a plane from unknown orientations[C]. Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 1999, 99: 666-673.
[43] Lowe D G . Distinctive Image Features from Scale-Invariant Keypoints[J]. International Journal of Computer Vision, 2004, 60(2): 91-110.
[44] Bay H, Tuytelaars T, Van Gool L. Surf: Speeded up robust features[C]. European conference on computer vision. Springer, Berlin, Heidelberg, 2006: 404-417.
[45] Rublee E , Rabaud V , Konolige K , et al. ORB: An efficient alternative to SIFT or SURF[C]. 2011 International Conference on Computer Vision. IEEE, Barcelona, Spain, 2011: 2565-2470.
[46] Rosten E, Drummond T. Machine learning for high-speed corner detection[C]. European conference on computer vision. Springer, Berlin, Heidelberg, 2006: 443-430.
[47] Rosin P L. Measuring corner properties[J]. Computer Vision and Image Understanding, 1999, 73(2): 291-307.
[48] 高翔. 视觉 SLAM 十四讲: 从理论到实践[M]. 北京：电子工业出版社, 2017.
[49] Vrakking M J J. An iterative procedure for the inversion of two-dimensional ion/photoelectron imaging experiments[J]. Review of Scientific Instruments, 2001, 72(11): 4084-4089.
[50] Hartley R I . In defense of the eight-point algorithm[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002, 19(6): 580-593.
[51] Longuet-Higgins H C. A computer algorithm for reconstructing a scene from two projections[J]. Nature, 1981, 293(5828): 133.

[52] Chetverikov D, Stepanov D, Krsek P. Robust Euclidean alignment of 3D point sets: the trimmed iterative closest point algorithm[J]. Image and Vision Computing. 2005, 23(3): 299-309.
[53] Li S, Xu C, Xie M. A robust O (n) solution to the perspective-n-point problem[J]. IEEE transactions on pattern analysis and machine intelligence, 2012, 34(7): 1444-1450.
[54] Gao X S, Hou X R, Tang J, et al. Complete solution classification for the perspective-three-point problem[J]. IEEE transactions on pattern analysis and machine intelligence, 2003, 25(8): 930-943.
[55] Penate-Sanchez A, Andrade-Cetto J, Moreno-Noguer F. Exhaustive linearization for robust camera pose and focal length estimation[J]. IEEE transactions on pattern analysis and machine intelligence, 2013, 35(10): 2387-2400.
[56] 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.
[57] Triggs B, McLauchlan P F, Hartley R I, et al. Bundle adjustment-a modern synthesis[C]. International workshop on vision algorithms. Springer, Berlin, Heidelberg, 1999: 298-372.
[58] Kümmerle R, Grisetti G, Strasdat H, et al. g 2 o: A general framework for graph optimization[C]. 2011 IEEE International Conference on Robotics and Automation. IEEE, Shanghai, China, 2011: 3607-3613.
[59] Gálvez-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.
[60] Lloyd S. Least squares quantization in PCM[J]. IEEE transactions on information theory, 1982, 28(2): 129-137.
[61] Sivic J, Zisserman A. Video Google: A text retrieval approach to object matching in videos[C]. Proceedings Ninth IEEE International Conference on Computer Vision. IEEE, Nice, France, 2003: 1470-1477.
[62] Robertson S. Understanding inverse document frequency: on theoretical arguments for IDF[J]. Journal of documentation, 2004, 60(5): 503-520.
[63] Dubbelman G, Browning B. COP-SLAM: Closed-form online pose-chain optimization for visual SLAM[J]. IEEE Transactions on Robotics, 2015, 31(5): 1194-1213.
[64] Lee D, Myung H. Solution to the SLAM problem in low dynamic environments using a pose graph and an RGB-D sensor[J]. Sensors, 2014, 14(7): 12467-12496.
[65] Latif Y, Cadena C, Neira J. Robust loop closing over time for pose graph SLAM[J]. The International Journal of Robotics Research, 2013, 32(14): 1611-1626.
[66] Hornung A, Wurm K M, Bennewitz M, et al. OctoMap: An efficient probabilistic 3D mapping framework based on octrees[J]. Autonomous robots, 2013, 34(3): 189-206.

致　　谢

两年的时间很短，过得也很快，转眼间两年的研究生生活就要结束了，这两年对我来说是很有意义的一段经历，我学到了很多，成长了很多，尤其是在完成毕业论文的这段时间。从刚开始接触 SLAM 领域时的迷茫到最后完成所有的论文内容，我深深地感受到了全身心地投入一个领域进行研究的乐趣与充足。在这里，我要感谢在完成毕业论文期间所有帮助过我的人。

首先要感谢所有在视觉 SLAM 研究领域作出杰出贡献的研究者们，是他们奠定了视觉 SLAM 领域的基石，为我们提供了很多研究成果与研究方向，让我们这些后者能够快速并且容易地入门。

还要谢谢教研室的两位老师，殷福亮老师和陈喆老师，殷老师对待科研态度严谨，工作一丝不苟，总能在我迷茫的时候为我指点方向。陈喆老师不仅工作认真，对我们的各方面也都很关心，无论是学术上的难题还是生活中的琐事，都给了我很多帮助。在这里衷心的谢谢两位老师。

我还要感谢与我一起研究视觉 SLAM 的教研室同学王晓珊，从开始我们一起入门，一起查找资料，有不懂的地方我们能够一起讨论，相互鼓励。我总是会向她请教不懂的问题，她也会耐心地告诉我，在完成论文的过程中她给了我很多帮助。

最后要谢谢我的母校大连理工大学，在这里六年，我已经爱上了这里的一草一木，在这里的所学所得，是我一生的财富。

再次感谢所有给予我帮助的人，谢谢你们。