视觉指导下的路径规划

完成日期：_____

指导教师签字：_____

答辩小组成员签字：_____

_____

_____

_____

_____

视觉指导下的路径规划

摘要

视觉 SLAM 是目前热门的研究项目。相比较于一般使用激光等其他传感器的 SLAM 技术，它的优点有主要使用廉价的传感器，无论是室内还是室外都可以使用。目前许多 SLAM 技术和路径规划的协同工作的研究都是基于激光传感器等使用的 SLAM 下的路径规划工作，而关于针对视觉 SLAM 进行的路径规划的研究却不是很多。本文使用开源的视觉 SLAM 算法为基石，在此基础上配合路径规划算法，来完成视觉指导下的路径规划的研究工作。首先，考虑到稳定性和实用性，本文选择了 ORB-SLAM2 作为底层的 SLAM 系统。因而得以继承 ORB-SLAM 优秀的性能和稳定的效率。之后，按照算法的实用性，本文选择了 A* 算法搭载在该 SLAM 系统之上。最终，通过实时测试，实验结果显示，二者结合之后可以稳定地运行一段时间，并能够较为准确地指出可能的路线。

关键词：计算机视觉；人工智能；视觉 SLAM；路径规划；ORB-SLAM

Path Planning Through Visual Guidance

Abstract

Visual SLAM is a hot research project currently. Compared with the SLAM technology which generally uses other sensors such as lasers, advantages of the visual SLAM are that the sensors are much cheaper and can be used both indoors and outdoors. At present, the researches on the collaborative work of many SLAM technologies and path-planning algorithms are based on the path planning work under SLAM used by laser sensors, etc., but there are not many researches on path planning for visual SLAM. Using the open-source visual SLAM algorithm as the cornerstone, the paper tries to apply the path planning algorithm to complete the research work of path planning through visual guidance. First, considering stability and practicality, the paper chooses ORB-SLAM2 as the base SLAM system. Therefore, the paper lets the durable efficiencies and great stabilities of ORB-SLAM continue. Then, after comparing the practicability of path-planning algorithms, the paper chooses the A-Star algorithm to be mounted on the chosen SLAM system. By real-time testing, the final experimental results of this paper show that the combination of the two can run stably for a period and can indicate possible routes with a relatively high accuracy.

Keywords: Computer Vision; Artificial Intelligence; Visual SLAM; Path Planning; ORB-SLAM

目　录

1 绪论

1.1 课题研究背景及研究意义

在美国连续剧集《西部世界》（*Westworld*）中，人类创造了一个宏大的属于招待员(Host, 人形机器人)的世界。在那里人类可以纵情地操纵招待员的命运，享受招待员世界所带来的一切。而如今人工智能技术的兴起，人类不免会担心像剧中发生的事情，如果确实在人类社会发生了，人类的命运会何去何从。

抛开远景的事情，当代生活，人工智能的运用越来越广泛。随着这样的趋势，许多公司开发出了一些智能系统，服务个人或国家机关。例如新增的摄像头上都有人脸识别技术，配合个人信息的查询，就可以很轻松地知道一个人的行踪信息，而不再大量依赖人工排查[1]；在饮料机上，加装人脸识别系统，就不需再输入密码就可以从使用者的金融账户扣钱，完成交易[2]；只需动动嘴，社交软件就可以通过你的声纹信息，为你登录账户等等。

不过，人类对于人工智能的滥用，迫使一些国家的立法机关行动起来了。2019年的 5 月，作为世界计算机技术的前沿，旧金山成为了全球第一个禁止政府使用人脸辨识技术的城市。之后，相信会有更多城市，会对政府滥用人工智能的行为说不。

即便如此，正确、正当地使用人工智能，是对国家和社会有好处的。当今社会，视障人士的出行由于公共基础设施建设的不完善或者老化，逐渐成为社会关注的议题。

图1-1 导盲犬和导盲马

对于视障人士来说，目前最为主要的解决方式是通过社会福利机构培训的导盲动物，如图1-1 导盲犬和导盲马图1-1 所示的导盲犬和导盲马等，通过各种方

1 海康威视官方网站，平安社区监控系统解决方案，
https://www1.hikvision.com/cn/jjfajs_107_i474.html(2019-5-22)
2 经济日报，人脸识别技术正广泛用于线下支付取快递等，
http://www.xinhuanet.com/fortune/2019-04/16/c_1124373216.htm(2019-5-22)

式，训练它们日后成为视障人士的“眼睛”。不过，导盲动物也有缺陷，它们的寿命不如人类，因此无法永远陪伴和帮助残障人士。例如，导盲犬的寿命在十年左右，而尽管导盲马的寿命可以长达廿几年，但相对于人来说或许还是没有那么长。科技手段发展至今，需要一个能够代替动物，解决上述问题的，有效的新解决方案。

如今，通过 SLAM 技术，传统计算机视觉所不能解决的问题，现在得到了部分的解决之道。从扫地机器人，到未来可能会越来越普遍的家用机器人保姆，它们几乎都需要 SLAM 技术的帮助。SLAM 技术可以说，是机器人的眼睛。透过 SLAM 技术可以让机器人完成各种室内室外导航的任务，拥有和人相似的视觉功能。

1.2 国内外研究现状

关于视觉 SLAM 的研究，国外的专家学者的研究成果非常之丰硕和翔实。

Hauke Strasdat 等人在期刊文章中比较了两种不同进行视觉 SLAM 的方法[1]，关键帧方法(Keyframe Methods)和过滤方法(Filtering Methods)。作者通过多次蒙特卡罗实验(Monte Carlo Experiments)、熵减少和均方根误差测量精度数据等，比较了两者在准确性和计算代价上的优劣。最后，作者得出结论，由于前者能够提供较高的精度，关键帧的 BA(**B**undle **A**djustment)方法，在作者考虑的所有情形下，都胜过过滤方法。

Stefan Leutenegger 等人提出了一种新的使用视觉线索和惯性线索互补进行视觉 SLAM 的方案[2]。作者也和上一段所述研究一样，使用了关键帧方法取代了过滤方法。作者提出的新方法将视觉测量同 SLAM 中的 IMU (**I**nertial **M**easurement **U**nit, 惯性传感器) 读数有机结合；并且选择性地边缘化了部分状态，保证优化窗口的大小不会无限扩充，以确保系统的实时性。

Hauke Strasdat 等人还提出了在恒定时间中的视觉 SLAM 的双窗体优化算法(Double Window Optimization)[3]。这种方式采用双层级方法，将主要区域的精确姿势点约束，同姿势-姿势(Pose-Pose)软约束确定的稳定边界确立联系。并自动构建一个合适的关键字和约束的连通图，动态选择内外双窗体成员并同时优化。这种双窗口方式，无论用误差最大的单目摄像机，还是使用能够提供相对更多的信息的 RGB-D 摄像机，都能取得较良好的结果。

不过中国的学者对视觉 SLAM 系统方面的研究成果的数量也比较可观，并且一些学者和研究人员的研究成果也是富有成效的。

霍彦希等人[4]提出的《基于单目视觉 SLAM 的无人快递车的设计与实现》，通过在小型机器人上加装 OpenCR 控制器，并使用 ROS 内置的 Gmapping 算法做路径规划、导航以及避障功能。并且使用其他软件作为支持，完成无人快递车

应该完成的其他任务，例如预约取件、拿取货物以及其他的人机交互功能等。

无人机，或无人飞行器，是一种拥有自主决策能力的空中机器人系统，而有关无人机的研究在中国也发展得如火如荼。马勋举等人的《基于双目全景视觉的无人机避障技术探究》一文[5]，介绍了运用双目摄像机系统进行全景视觉的构建，并构建视差图获得障碍物位置进行避障等技术。

华中科技大学的万裕琳的硕士毕业论文以 ORB-SLAM2 作为基底[6]，借助以单目模式下运行的 ORB-SLAM2，结合 IMU 做成了一个完成度较高的新的视觉 SLAM 系统。这个新的 SLAM 系统在文章作者认为是只用于水平地面的导航与定位的，不过如果遇到需要上坡下坡的情况的话就显得有些局促。所以它的全局适用性相对较差，但对于地面导航是足够的。

哈尔滨工业大学邹谦的硕士毕业论文中[7]，提出了一项利用已经固定位置的二维码来做图优化的 SLAM 系统的技术。基于这种技术，SLAM 系统建图的准确度相较于纯激光传感器的 SLAM 系统得到了显著提升，得到了更好的地图一致性。

最后，国内有科技企业研发出了 SLAMWARE，一种商业化的模块化自主导航定位模块[1]。该模块号称是一种“一站式解决方案”，在它的加持下，机器人系统可以利用该模块提供的 API 完成环境见图和路径规划，还可以在此基础上通过厂商提供的 SDK 来扩展自己的功能。

1.3 研究目标和内容

本论文将采用开源项目 ORB-SLAM2 为基本框架[8]，辅以路径规划算法，来进行视觉指导下的路径规划的研究。

本文共设五个章节，内容的安排如下：

第一章绪论，即本章节，介绍有关本课题的研究背景、国内外有关研究现状的简单介绍以及对本文所述内容的简单说明。

第二章视觉 SLAM 系统和路径规划算法理论基础，首先将详细介绍有关本项目所使用的基础知识，包括有关空间刚体运动、基于此而求解相机位姿的方法以及 ORB-SLAM 系统的运作原理等。在此之后，将继续介绍目前比较主流的一些路径规划算法，这些算法有些是图论之中图搜索的一些经典算法，有些有其创新性而不同于传统的路径规划算法。

第三章路径规划同视觉 SLAM 的结合，将介绍本文在探索视觉 SLAM 系统和整合路径规划模块等所做的工作。

第四章在视觉 SLAM 指导下的路径规划，将会介绍本文利用上述理论基础

1 https://www.slamtec.com/cn/Slamware

所进行的实验详情。

第五章是总结与展望。这一章将对本文的所有工作进行总结，并阐述所做工作的下一步改善的方向等。

此外，本文所做的所有工作，使用和编写的代码托管在 GitHub 上：

https://github.com/catduck2010/path_planning_vSLAM

2 视觉 SLAM 系统和路径规划算法理论基础

2.1 视觉 SLAM 系统概述

SLAM (**S**imultaneous **L**ocalization **A**nd **M**apping, 即时定位和地图构建)的发展至今已经有了大约 30 多年的历史。它的出现，它是为了解决机器人如何在未知的环境中移动的问题，如何通过观察环境来确定自己的轨迹，同时建立一个环境地图。所以，SLAM 技术，是诸多技术的集合。

SLAM 技术涉及面广，根据传感器、应用场景和核心算法的不同，SLAM 有多种分类方法。按传感器分类，可以分为 1) 基于激光雷达的 2D/3D SLAM、2) 基于深度相机的 RGB-D SLAM、3) 基于视觉传感器的视觉 SLAM (vSLAM)、4) 基于视觉传感器和惯性单元的 VIO (**V**isual **I**nertial **O**dometry, 视觉-惯性里程计)。

基于激光雷达的 2D SLAM 相对成熟。2005 年，Sebastian Thrun 等人所著的经典书籍 *Probabilistic Robotics* 对 2D SLAM 进行了非常彻底的研究和总结[9]，基本上确定了激光 SLAM 的框架。目前，常用的网格映射方法也已有十多年的历史。2016 年，Google 开源了 Cartographer[10]，这个激光雷达 SLAM 系统，可以融合 IMU 信息，既可以处理 2D 又可以处理 3D SLAM。目前，2D SLAM 已经成功地应用于家用扫地机器人。

基于深度相机的 RGB-D SLAM 在过去这些年来也发展得很快。2010 年，微软的 Kinect 上市。从那时起，RGB-D 的研究就开始了。在短短几年内，几个重要的 RGB-D 算法相继出现。微软正在进行开发者测试的 AR 设备 HoloLens 的第一代和第二代也都集成了 RGB-D SLAM。

如图2-1 所示，基于视觉传感器的视觉 SLAM，目前也是研究的一大热点。视觉 SLAM 使用的传感器，包括单眼摄像机、双目摄像机、鱼眼摄像机等。它成为大热的原因，是因为它主要使用廉价的传感器，无论是室内还是室外都可以使用。在视觉 SLAM 中，根据视觉特征提取的方式，将其分为特征提取法、直接提取法等。目前，视觉 SLAM 的典型算法有 ORB-SLAM、SVO、DSO 等。

图2-1 从左至右分别为 Kinect(第一代)、USB(单目)摄像头和鱼眼相机

尽管本文并不涉及关于机器人的运动的研究，但实际上本文的研究可以看作是由本系统引导用户寻找路径，即把用户放在了机器人的位置上来进行考量的。

尽管这样说可能是对未来它的使用者的一种冒犯。

2.2 空间刚体运动与相机位姿

相机在空间中的运动是刚体运动。刚体运动保证了同一个向量在各个坐标系下的长度和夹角不变。与此同时，SLAM 系统使用的坐标系与现实生活中使用的坐标系有所不同。由于我们人类生来习惯这种坐标系，转换到新的坐标系的时候，我们理解它会出现一些困难和不适应。因此人类通过数学方式，找到了坐标系之间相互转换的方式。

求解相机位姿是单目 SLAM 系统中的经典问题。由于单目相机的本身的特点，在单目相机抓取的图像上获取特征点是不可能知道点的深度信息的。而实时 SLAM 系统运作的时候需要运动，所以存在尺度不确定性。因此，单目视觉 SLAM 系统需要在运行初期进行初始化，把图像的特征点三角化，获得它们的 3D 位置。此时如果出现了新的帧，就可以通过匹配求解这些算出来的 3D 位置在新的帧上的位置。根据这些对应关系就可以通过求解 PnP 问题来算出相机的位姿。

本小节主要介绍这两个方面的理论基础。

2.2.1 坐标系和旋转矩阵、变换矩阵

为了表示方便，SLAM 系统在一般情况下会使用相机坐标系进行计算。相机坐标系中，摄像机在原点，*x* 轴正方向指向右，*z* 轴正方向指向前(朝向屏幕内或相机方向)，*y* 轴正方向指向相机的上方。

相机坐标系和世界坐标系在各种方式下都有些不同，如果需要将二者转换需要一定的换算方式。相机坐标系只可以表示特征点在相机的视角下的位置信息，并不能描述相机本身的运动情况以及特征点在世界空间中的位置。所以需要将相机坐标系和世界坐标系之间进行相互转换。

可以提供这种转换的就是变换矩阵。在此之前先简单描述一下旋转矩阵。

设一个坐标系的单位正交基为$(\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3)$，经过旋转之后转化为另一个坐标系的单位正交基，为$(\boldsymbol{e}_1', \boldsymbol{e}_2', \boldsymbol{e}_3')$。则对同一个向量$\boldsymbol{a}$，在两个坐标系下的坐标为$[a_1, a_2, a_3]^T$和$[a_1', a_2', a_3']^T$，按照坐标的定义，有：

$$[\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3]\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = [\boldsymbol{e}_1', \boldsymbol{e}_2', \boldsymbol{e}_3']\begin{bmatrix} a_1' \\ a_2' \\ a_3' \end{bmatrix}$$

等号两边同时左乘$\begin{bmatrix}\boldsymbol{e_1^T}\\ \boldsymbol{e_2^T}\\ \boldsymbol{e_3^T}\end{bmatrix}$，得：

$$\boldsymbol{a}=\begin{bmatrix}a_1\\ a_2\\ a_3\end{bmatrix}=\begin{bmatrix}\boldsymbol{e_1^T e_1'} & \boldsymbol{e_1^T e_2'} & \boldsymbol{e_1^T e_3'}\\ \boldsymbol{e_2^T e_1'} & \boldsymbol{e_2^T e_2'} & \boldsymbol{e_2^T e_3'}\\ \boldsymbol{e_3^T e_1'} & \boldsymbol{e_3^T e_2'} & \boldsymbol{e_1^T e_3'}\end{bmatrix}\begin{bmatrix}a_1'\\ a_2'\\ a_3'\end{bmatrix}\triangleq \boldsymbol{Ra'}$$

则中间这个矩阵$\boldsymbol{R}$就是旋转矩阵。它的行列式为 1，表述了坐标系的旋转，并且它的逆或转置矩阵表示一个相对于它相反的旋转。

再设$\boldsymbol{t}$为平移向量，那么将旋转和平移联系到一起，有：

$$\begin{bmatrix}\boldsymbol{a'}\\ \boldsymbol{1}\end{bmatrix}=\begin{bmatrix}\boldsymbol{R} & \boldsymbol{t}\\ \boldsymbol{0^T} & \boldsymbol{1}\end{bmatrix}\begin{bmatrix}\boldsymbol{a}\\ \boldsymbol{1}\end{bmatrix}\triangleq \boldsymbol{T}\begin{bmatrix}\boldsymbol{a}\\ \boldsymbol{1}\end{bmatrix}$$

我们称矩阵$\boldsymbol{T}$为变换矩阵。

在现代的 SLAM 系统之中，我们一般只会使用两个坐标系——相机坐标系和世界坐标系。这里设某个点的世界坐标为$\boldsymbol{d}_w$，相机坐标为$\boldsymbol{d}_c$，则有：

$$\boldsymbol{d}_c=\boldsymbol{T}_{\mathbf{cw}}\boldsymbol{d}_w,\ \boldsymbol{d}_w=\boldsymbol{T}_{\mathbf{wc}}\boldsymbol{d}_c=\boldsymbol{T}_{\mathbf{cw}}^{\mathbf{-1}}\boldsymbol{d}_c$$

其中$\boldsymbol{T}_{\mathbf{cw}}$表示从世界系向相机坐标系的变换，$\boldsymbol{T}_{\mathbf{wc}}$反之。二者都可以表示相机的位姿。

2.2.2 PnP 问题简述

PnP(**P**erspective-**n**-**P**oint)问题指通过世界中的 n 个特征点与抓取的图像中的 n 个像点，计算出两者之间的投影关系，来获得相机位姿的问题。

假设相机光心的位置为点O，有P_1，P_2，… P_n这 n 个特征点。

此时如果相机O只见到P_1一个特征点，那么相机的位姿就在由向量$\overrightarrow{OP_1}$的模为半径，以P_1为球心的球上，有无数个解。

如果此时相机O又多见到一个特征点P_2，那么又可以确定一个球，是以P_2为球心，向量$\overrightarrow{OP_2}$的模为半径的另一个球。此时相机位姿的解在两球的相交线上，有无数个解。

以此类推，如果有三个特征点P_1，P_2，P_3，三个球相交的话，相机的位姿解有 4 个。显然这四个之中的一个是正解。但为了降低计算机的资源消耗，我们会先计算出这四个解，然后找到另外一个特征点，求它在图像中的 4 个投影。取投影误差最小的解，即为正解。

在 SLAM 系统之中，一般的做法是先使用 P3P 或 EPnP 等方法估计相机的位姿，然后构建最小二乘法优化问题进行 BA。

2.2.3 PnP 问题求解

PnP 问题的解法有很多种[11–15]。此处简要介绍通过 P3P 来求解 PnP 问题的方法。

P3P 利用了给定 3 个点的几何关系，还需要一对验证点。因此需要输入 3 对 3D-2D 点。如图2-2 所示，记空间坐标系中的点为 A, B, C，平面坐标系的点为 a, b, c，这 6 个点之中，小写字母表示的点为大写字母表示的点在相机成像平面上的投影，并且我们已经知道了 A, B, C 三点在世界坐标系中的位置。此外，设验证点对为 $D-d$，相机光心为 O。

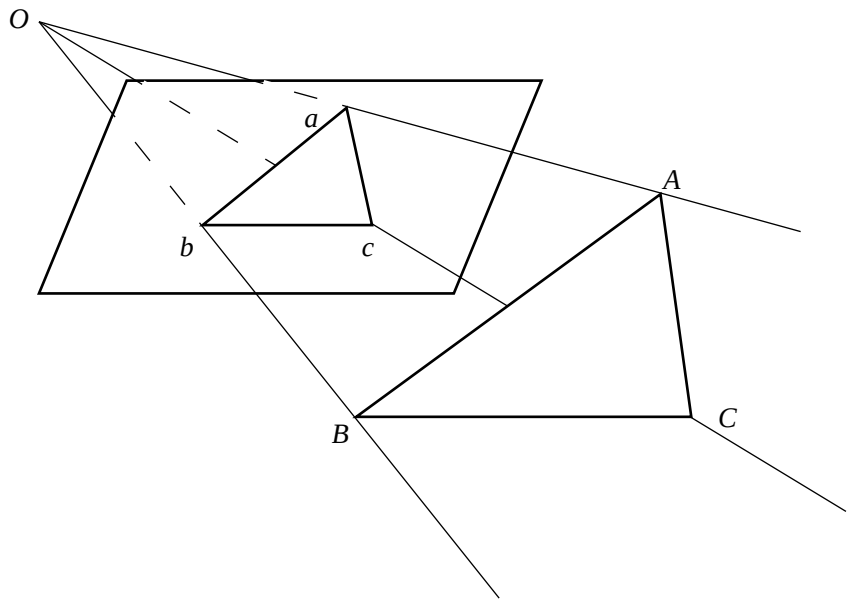

图2-2　P3P 问题示意[16]

显然，三角形之间有如下对应关系：

$$\Delta Oab \sim \Delta OAB,\quad \Delta Oac \sim \Delta OAC,\quad \Delta Obc \sim \Delta OBC.$$

考虑它们之间的相似性质，有：

$$OA^2 + OB^2 - 2OA \cdot OB \cdot \cos\langle a,b\rangle = AB^2$$
$$OA^2 + OC^2 - 2OA \cdot OC \cdot \cos\langle a,c\rangle = BC^2$$
$$OB^2 + OC^2 - 2OB \cdot OC \cdot \cos\langle b,c\rangle = AC^2$$

对上述三式同时除以OC^2并记$x = OA/OC$，$y = OB/OC$，$v = AB^2/OC^2$，$uv = BC^2/OC^2$，$wv = AC^2/OC^2$，就得：

$$x^2 + y^2 - 2xy\cos\langle a,b\rangle = v$$
$$x^2 + 1^2 - 2xy\cos\langle a,b\rangle = uv$$
$$y^2 + 1^2 - 2xy\cos\langle a,b\rangle = wv$$

消去v，得：

$$(1-u)y^2 - ux^2 - \cos\langle b,c\rangle\, y + 2wxy\cos\langle a,b\rangle + 1 = 0$$

$$(1-w)x^2 - wy^2 - \cos\langle a,c\rangle\, x + 2wxy\cos\langle a,b\rangle + 1 = 0$$

这之中，由于 2D 点的位置是已知的，那么$\cos\langle a,b\rangle$, $\cos\langle a,c\rangle$, $\cos\langle b,c\rangle$就是已知量，然后$u = BC^2/AB^2$, $w = AC^2/AB^2$可以通过 A, B, C 三点的世界坐标计算出来。因此两个式子中只有 x, y 是未知的。现在就构造出一个，x, y 的二元二次方程组，这个方程组最多有 4 组解。

用 $D-d$ 点对代入验证，误差最小的一组解，就是正解。

2.3 ORB 算法简述

ORB (Oriented FAST and Rotated BRIEF)算法由 Ethan Rublee[17]等人提出。本算法通过一种经过改良的 FAST (**F**eatures from **A**ccelerated **S**egment **T**est)方法检测特征点，并用 rBRIEF (旋转 BRIEF)算法计算特征点的描述子对特征点进行描述来获取视觉图像的特征。这种改良的方法修正了原本方法的不足之处和部分缺陷，改良之后的方法能够更为高效地提取视觉图像的特征。

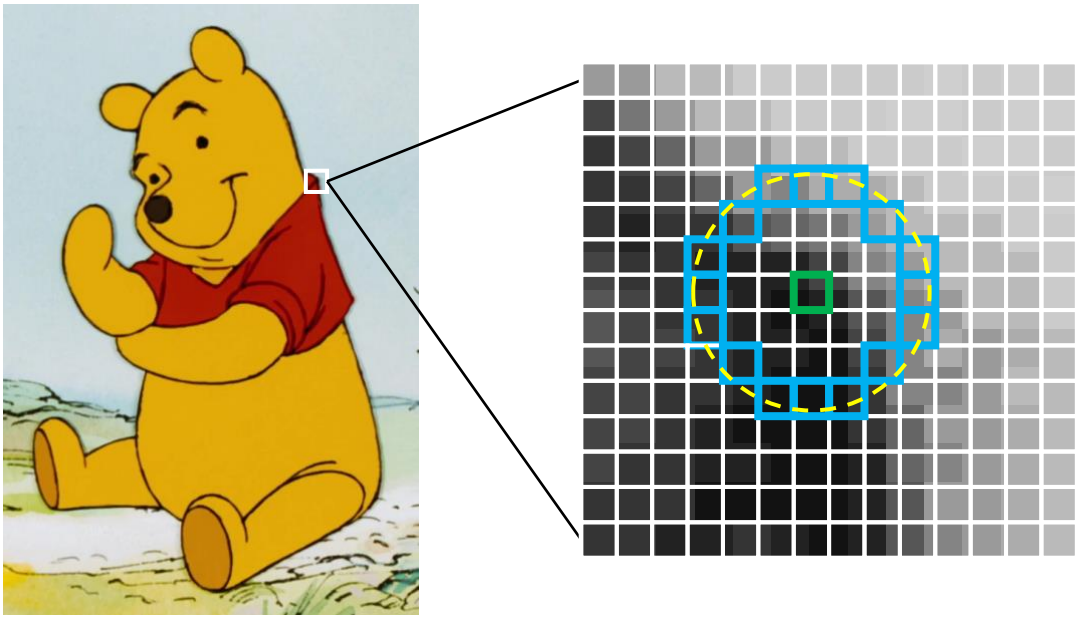

图2-3　ORB 算法示例

FAST 方法的核心就是任意取一个点，如果这一个点和周围其他的大部分点都不一样，就认为它是一个特征点。这种方法相对于 Harris 角点算法来说更为高效[18]。ORB 特征中使用不同尺寸图像构成的图像金字塔，并分别在每一金字塔层图像上提取 FAST 特征，这样每一个被提取的特征就都有了尺度属性。

ORB 使用的 rBRIEF 算法主要解决了 Michael Calonder[19]等人提出的 BRIEF (**B**inary **R**obust **I**ndependent **E**lementary **F**eatures)描述子不具备旋转不变性、对噪声敏感的缺点。

2.4 ORB-SLAM2 概述

ORB-SLAM2 是 Raul Mur-Artal 等人提出的[8,20]开源视觉 SLAM 算法，其和 ORB-SLAM 一齐在 GitHub 上开源提供，遵循 GPLv3 开源协议。

ORB-SLAM 是一个实时的、基于特征点的单目 SLAM 系统。ORB-SLAM2 则在前者的基础上，摆脱了对 ROS 的完全依赖，并又继续支持了标定后的双目相机和 RGB-D 相机。这样的话，在一些情况下可以不再需要调用 ROS 的函数或方法。作为一个具鲁棒性的视觉 SLAM 系统，它包含了所有视觉 SLAM 系统均包含的四个过程：跟踪(Tracking)、建图(Mapping)、重定位(Re-localization)和闭环检测(Loop Closing)。

在 SLAM 模式下，系统使用自行设计的 ORB 算法，进行特征提取，并有三个线程同时进行，分别是跟踪、局部建图(Local Mapping)和闭环检测。这三个线程的源代码分别放在三个不同的文件里，非常工整以及易于阅读。

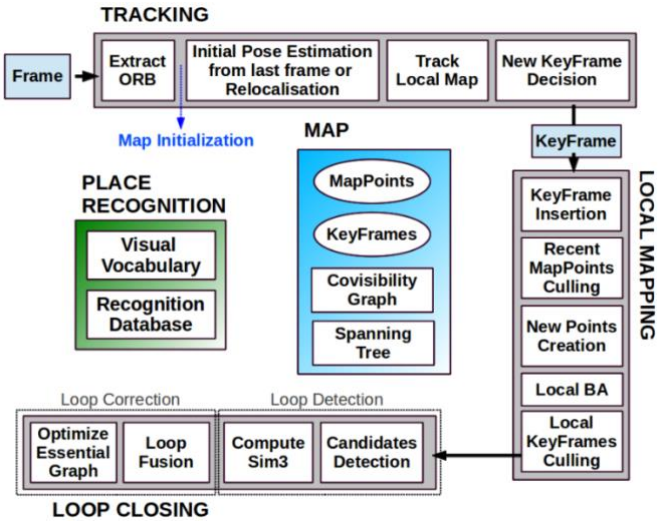

图2-4 ORB-SLAM2 系统流程图[20]

图2-4 是 ORB-SLAM2 整个系统的流程图。接下来本文将会分别简要介绍三个大板块所实现的功能。

2.4.1 跟踪线程

在 SLAM 系统之中，不可能把所有的帧都用来构图，如果这样做的话一定会拖累系统的效率。因此，需要借助跟踪(Tracking)这种方式来把所有的帧里面系统判定为有用的帧拿出来，作为关键帧来进行后续的构图操作。

跟踪线程还会跟踪计算每一帧视觉图像，寻找关键帧和局部地图之间的对应关系，优化帧位姿。

在跟踪之前，需要进行初始化过程，用 ORB-SLAM2 的 ORB 算法进行特征

提取。为了保证提取的点的质量，首先会提取出大量的特征点。和 OpenCV 的 ORB 算法不同的地方在于，为了使特征点，平均分布在整个图像的各个区域，提取时会先划分出数个划分格，并尝试保证各个划分格中至少有 5 个特征点。如果不能达标，系统会重新尝试更改这些限定条件。提取之后会计算各个特征点的方向信息和描述子，以便之后进行特征匹配。

2.4.2 局部建图线程

局部建图(Local Mapping)主要是对所有的关键帧来作出处理。

1) 插入关键帧

第一步先处理新插入的关键帧，将其作为新节点加入地图。然后，更新地图中地图点和当前关键帧的关联性，并更新这些新关系。最后，对当前关键帧计算单词，以便用于闭环检测。

2) 删除已不再有用的地图点

为了保证生成地图的质量，需要在 3 帧之内重新衡量地图点，这样才可以确保它们可以用于跟踪线程，而且三角化并创建地图点时的过程不出错。因此，满足条件 a) 被观测的帧数量比上应被观测的帧数量的比值小于 0.25 或者 b) 从这个地图点创造了多个关键帧，但可以观测到它的关键帧不多于 3 个的地图点，就不再需要了，它将被移除。

3) 创建并增加新地图点

相机视角运动过程中，新关键帧会和共视图中其他的关键帧三角化，并产生新地图点。对在当前帧没匹配到的特征点，要在别的关键帧中继续寻找。如果新的地图点投射到其他关键帧上，查到了新的对应关系的话，也会删掉重复的地图点。

4) 删除冗余关键帧

如果某关键帧九成以上的地图点可以被 3 个以上的别的关键帧检测出来，则此关键帧会被看作冗余关键帧，系统会删去它以保证关键帧数量不会疯狂增长，进而影响局部 BA 运算的复杂度。

2.4.3 闭环检测线程

闭环检测(Loop Closing)线程会获取上述局部建图线程的最后一个关键帧，尝试进行闭环检测以及调整检测到的闭环。下述是这个线程主要完成的功能。

1) 查找候选闭环检测所用的帧

第一步，先判定距离上次的闭环检测是否超过了 10 帧，然后通过计算出与当前帧相连的帧的 BoW 的最小得分，以计算出阈值。之后，搜索与当前帧有相

同单词的不与其相连的其他帧，统计出与当前帧所有的最多单词数，将此作为阈值来过滤出一个或多个候选闭环帧。之后是下述的计算 Sim3 的过程。

2) 计算 Sim3

为了成功闭合检测到的闭环，需要计算 Sim3 变换。Sim3 是一种自由度为 7 的变换，其中，旋转自由度为 3，平移自由度为 3，尺度自由度为 1. 闭环检测线程会对各个循环候选帧进行变换尝试，并用 RANSAC(**Ran**dom **Sa**mple **C**onsensus, 随机抽样一致性)方法，进行计算迭代[21]。若经过这种迭代优化之后依然存在足够多的内点，那就将此闭环候选帧作为进行融合闭环数据使用的帧。

3) 融合闭环数据

第一步，先刷新当前帧的共视关系，用之前已经计算出的 Sim3 来调整当前关键帧的位姿信息，并根据传递获得和当前的关键帧相连的帧的 Sim3 变换信息，调整相连帧的位姿。之后，进行数据融合，将和当前帧相连的地图点和闭环帧的地图点融合。这样做的原因是，这些点中有大量冗余和重复。最后，调整关键帧之间的连接关系，再进行整体 BA 来优化更新后的关键帧以及地图点。

2.4.4 其他功能

ORB-SLAM2 中的其他板块实现的功能主要有地图绘制、相机位姿求解等。相对于前辈系统使用两个线程的方案，ORB-SLAM 的三线程方案现在受到了普遍的认可[16]。由于上述三个同时运行的线程有可能出现并发访问同一个变量的情况，因此在一些需要互斥访问的类中引入了线程锁，来确保线程之间的运作关系趋于稳定。

2.5 路径规划算法综述

路径规划算法是机器人实现路径导航的基础技术。它所解决的主要问题是，找到一个路线让机器人能够完好、无伤地找到通往指定地点的路线，而不触碰到可能出现的障碍物、或者突然在其面前倒下的人等。路径规划主要涉及三个方面的问题：1) 明确起点和终点的位置；2) 避障；3) 尽可能优化路径。

根据对外界环境的掌握情况的范围大小，路径规划还可以区分为两种，全局路径规划、局部路径规划。

全局路径规划，就如同它的字面意思，即在已知的环境中，规划出一条路径出来。它的精度取决于环境的准确度，如果现实中出现了地图中不存在的障碍物，这种方法就无能为力了。由于它是事前规划，因此对机器人的计算能力要求略低。虽然规划结果是全局性的、较好的，但对当前环境中的模型误差和噪声的鲁棒性较差。

全局路径规划的应用主要有城市地图的导航、游戏内地图的导航等。在游戏地图很大的情况下，路径规划的作用主要是指引玩家到指定位置等。如图2-5，天蓝色的箭头指导玩家朝着任务地点方向行进。由于整个地图是已知的，也不存在很多障碍物(即使存在，多数情况下都是要求玩家捡的材料等)，因此开发者使用了全局路径规划。

图2-5　某手机游戏中开放世界的路径规划

而局部路径规划，对机器人来说环境信息可能完全未知或有小部分可知，侧重考虑机器人目前所处的局部环境信息，这样使得机器人能够具有良好的避障能力，通过传感器等方式对机器人的工作环境进行探测，以获取障碍的世界位置和几何性质等信息。这种规划方案需要收集环境信息，并对环境模型的动态更新随时进行修改、校正。局部规划方法能把环境的建模同搜索相结合，要求机器人系统具有较强的计算能力，需要它对环境误差和噪声具有很强的鲁棒性，能实时反馈和修正规划结果。然而，由于缺乏全局环境信息，规划结果可能不是最优的，甚至可能无法找到正确的路径。

本章余下内容主要简单介绍一些主流的路径规划算法以及对它们的一些简要分析。

2.6　经典的路径规划算法

2.6.1　DFS 和 BFS 算法

DFS(Depth First Search, 深度优先搜索)和 BFS(Breadth First Search, 广度优先搜索)算法，都是非常经典的图的基本算法。二者主要的目的，是遍历图中的所有节点，同时也可以用于搜索。

DFS 算法在搜索过程中，1) 访问图的某一个顶点后，2) 需要递归地访问此

顶点的所有没有访问过的相邻顶点。过程描述如下：

深度优先搜索(DFS)算法

输入：开表和关表，前者保存所有已生成而未考察的节点，后者保存已访问过的节点；
输出：节点序列，即最短路径。

初始条件下，选择一个点，作为起始顶点，并按照下述步骤遍历：
a) 选择起始顶点放入开表；
b) 把它的邻接节点放入开表；
c) 如果点的邻接节点都在开表中，则放节点进入关表，并回溯上一个节点；
d) 返回 b)直至开表为空。

BFS 算法在 1) 进一步遍历图中顶点之前，2) 先访问当前顶点的所有邻接结点。过程描述如下：

广度优先搜索(BFS)算法

a) 将起始节点放入队列尾部；
b) *while*(队列不为空)：
取得并删除队列首节点 *n*；
处理节点 *n*；
把 *n* 的未处理相邻节点加入队列尾部。

2.6.2 Dijkstra 算法

Dijkstra 算法是寻找最短路径的典型算法，此算法在 1956 年，由荷兰王国的计算机科学家 E W. Dijkstra 最初提出[22]。具体过程描述如下：

Dijkstra 算法

输入：开表和关表，前者保存所有已生成而未考察的节点，后者保存已访问过的节点；
输出：节点序列，即最短路径。

a) 访问图中里起始点最近且没有被检查过的点，把这个点放入开表中待查；
b) 从开表中找出距起始点距离最近的点，揪这个点的所有子节点，把这个点放到关表中；
c) 遍历考察这个点的子节点，求出和起始点的距离值，置子节点到开表中；
d) 回到步骤 b 继续，直到找到目标点或开表为空。

不过，Dijkstra 算法的缺点也非常明显。由于需要进行大规模的节点遍历，从起始点开始向周围的点群层层扩展、计算之后，才能到达目标点。因此速度慢并且效率很低。

2.7 A* 算法

A*(A-Star)算法由 P. E. Hart 等人在 1968 年提出[23]。它的核心是如下的公式：

$$f(n) = g(n) + h(n)$$

其中$f(n)$是表示，每个可能试探的点的估价函数，$g(n)$则表示，从初始搜索点到当前的点的代价的估价函数，$h(n)$是当前点到目标点的估价函数。

A*算法克服了上一节所述的三个算法的不足之处，描述过程如下：

A* 算法

输入：开表和关表，前者保存所有已生成而未考察的节点，后者保存已访问过的节点；
输出：节点序列，即最短路径。

a) 将起始点 *S* 放入开表，关表置空；
b) 如果开表不为空，从表头取节点 *n*，如果为空则停止；
c) *n* 若是目标解，则输出一条路径；
d) 展开 *n* 的子节点，如果不在关表里，就把它们放入开表，并把 *n* 放入关表，计算 *n* 个子节点的$f(n)$值；

e) 将每个开表内的节点按$f(n)$值的大小排序，最小的在开表表头；
f) 转到 b)。

A*算法同 Dijkstra 算法，主要区别在于是否有估价值，后者相当于前者中估价为 0 的情况，即：

$$f(n) = g(n)$$

其中$f(n)$是每个可能试探的点的估价函数，$g(n)$表示从初始搜索点到当前的点的代价的估价函数。

A*算法是最为有效的在静态路网中求最短路径的直接搜索方法。算法核心公式中的$h(n)$与实际值越接近，算法就能越快搜索。尽管它是静态方案，但是它也是可用于动态路径规划当中的。不过当环境发生变化时，就需要重新计算、规划路线，消耗计算资源。

2.8 其他路径规划算法

2.8.1 D*和 D* Lite 算法

D*算法，是目前比较主流的动态规划算法。它和 A*算法主要的区别是，遇到障碍时，它会在受困位置起重新计算部分路线，而不是完全重新计算。

D* Lite 算法由 Sven Koenig 提出[24]。它的核心思想与 D*算法相反，它是从终点开始，求到起点的路线的算法。二者的实现方式比较类似，可以看作是一样的算法。

由于二者都是动态规划算法，内容较为复杂，本文在此不再详细介绍。

2.8.2 RRT 算法

由 Steven M. LaValle 等人提出的 RRT(**R**apidly Exploring **R**andom **T**ree, 快速扩展随机树)算法方式[25,26]，通过地图碰撞检测，省去了建图步骤。而且，这种方式能够有效解决高维空间和在复杂的约束条件中的路径规划问题。

RRT 算法的缺陷是，如果在比较狭窄的空间或存在大量障碍物的空间中，能找到通向狭窄通道的路需要“碰运气”——有的时候能很快找到，但有的时候怎么都找不到；其次，RRT 虽然能找到路径，但它找出的路径的形状会有一点奇怪，也不是最短路径。

2.9 本章小结

本章详细介绍了关于 SLAM 系统和路径规划算法所涉及的基础知识等。例

如有关 SLAM 系统的一些基本概念、相机位姿的计算和表示法、图像的特征点的提取和表示等；以及关于路径规划算法的基础，简要介绍了几种常用的路径规划算法。本章是本文工作的基础，涉及的数学知识、算法知识等较为深奥，需要细细理解和运用。

3 路径规划同视觉 SLAM 的结合

本章主要介绍将路径规划算法运用于视觉 SLAM 系统之上，本文所做的主要工作。

3.1 坐标系和相机位姿

ORB-SLAM2 在使用单目镜头的时候是一个所谓“循迹”系统，意思是系统并不清楚它所描绘的 3D 空间的尺度如何。换句话说，如果不经过适当的转换，它描绘的 3D 空间是不能看作与我们人类所处的 3D 空间等价的。

不过，由于 SLAM 系统一般是实时进行的，不会涉及读写存档和对其重新匹配的问题。因此此处暂时不去考虑尺度的对应关联。

由于 ORB-SLAM2 的坐标系的特殊性，需要使用特定方式来获得地图点的三维坐标。为了实现地图和相机位姿表示的可视化效果，ORB-SLAM2 提供了一些访问地图点和相机位姿中的世界坐标的方法。

获得相机位姿在本课题之中的一个目的是，是为了让系统在规划路径时有一定的指向性。如果 SLAM 模式开始检测时镜头模组没有看后面，而一直看着前面，那么系统会误认为后面的路也可以走，造成错误。

3.2 地图和障碍物

3.2.1 地图降维

为了降低寻路难度，需要用一些简单的方法将地图降维，使得求出路径的效率更高。我们生活中摆放物品的时候，一般情况下都是横平竖直的，而且本文只考虑人的行走，不考虑例如无人机等的飞行避障。因此可以将 3D 地图转化为 2D 地图进行路径规划。

我们生活中一般使用的都是空间直角坐标系。如果从某一个轴的正方向到负方向对所有地图点进行投影，就能够得到一个平面直角坐标系的地图。

ORB-SLAM2 使用 OpenGL 进行地图的绘制，而 OpenGL 使用右手系。伸出右手，将食指指向上方，中指弯曲，和拇指相互垂直。此时拇指的位置就是 x 轴正方向，食指是 y 轴正方向，而中指就是 z 轴正方向。

3.2.2 避障问题

还有，如果 ORB-SLAM2 在地面上也检测到了特征点，不让这些点不被误认为是障碍也是一个问题。笔者曾在家中，运行 ORB-SLAM2，由于家中的地砖是

带纹理的，因此在地面上检测到了数量庞大的特征点。但地砖上的纹理显然并不是障碍。

3.3 虚拟目标

设定虚拟目标点的原因，是因为要寻找路线，首先要订立一个目标。起始点是相机的光心，终点也要确定下来才可以。不然就没有方向，更不用说规划路线。不过使用者未必知道自己到底究竟要去哪里，所以问题变成了系统自己找一个终点，指导使用者往那个终点走。

3.3.1 虚拟目标的计算

根据相机的位姿数据，通过相机位姿的矩阵表示法，即变换矩阵$\boldsymbol{T}_{\mathbf{wc}}$或$\boldsymbol{T}_{\mathbf{cw}}$，不仅可以知道相机光心的世界坐标，还可以通过变换法向量，计算出当前视线的方向，从而获得大体的方向信息。这样一来，朝哪里走的问题就得以解决。

本文对确定目标点的解决方式是 1) 先用在相机坐标系中指向镜头前方的法向量$[0,0,1]^{\mathrm{T}}$补齐维数，左乘变换矩阵$\boldsymbol{T}_{\mathbf{wc}}$算出世界坐标并降维之后；2) 算出该点和相机光心所在的直线的斜率和截距，并找到该直线和地图边界的交点，即为假定的目标点。

如果仍无法用上述方法确定目标点的世界位置，可以考虑旋转法向量后再计算，即不再在镜头直视位置的延长线上找终点，而是在镜头的视角范围内找一系列可能的终点。取一个离起点最远的点作为终点，再进行路径的计算求解。

3.3.2 虚拟目标的局限

如上所述的确定虚拟目标的方法尽管能确定一个目标，但还是有一定的局限性。这样做的好处就是保证了目标点一定在使用者视线之内，但这样做的缺点就是如果前边没有路就不能找到路径。如果使用者在一个类似迷宫的地方里使用此系统，那就需要靠使用者自行确定哪里能走了。

3.4 路径规划算法和度量单位

本文使用 A* 算法编写程序。 A* 算法使用整形变量来表述路径，而 ORB-SLAM2 使用浮点数米(meter)来表示地图的尺度。尽管不能和现实世界的米完全对应，但这样的尺度对于整形变量表示的米来说就不太一样了。因此如果地图不够大，会产生终点和起点非常接近，乃至重合的错误。

笔者为了缓解这种情况，程序中将地图的尺度扩大之后，再进行搜索路径。

确定一条或多条路径之后，再还原回原来的尺度。不过这种方法并不能根除这个问题，第五章会详细解释。

3.5 本章小结

本章详细阐述了本文面对路径规划问题时采用的解决方式，是本文下一章阐述开发进程，进行试验的一些主要思想。下一章主要探讨，根据本章对这些可能遇到的问题的分析，进行再开发以及实验的具体内容等。

4 在视觉 SLAM 指导下的路径规划

4.1 准备工作

笔者利用一台运行着 Ubuntu 16.04 LTS 的 Linux 发行版的便携式计算机，通过计算机屏幕上方的摄像头来采集图像信息。为了调试方便，笔者安装了 ROS Kinetic 以在 ORB-SLAM2 需要的情况下得以使用它。不仅是 ROS，其他需要的依赖也安装上了，按照自述文档简单配置后，笔者完成了对项目自带的 Examples 文件夹下的范例的编译工作。此时意味着，它已可以使用，并可以开始工作了。

按照 ORB-SLAM2 源代码关于地图绘制的部分，笔者经过简单探索后找到了在 ORB-SLAM2 的 Map Viewer (地图查看器，如图4-1)上绘制简单路径的方式。

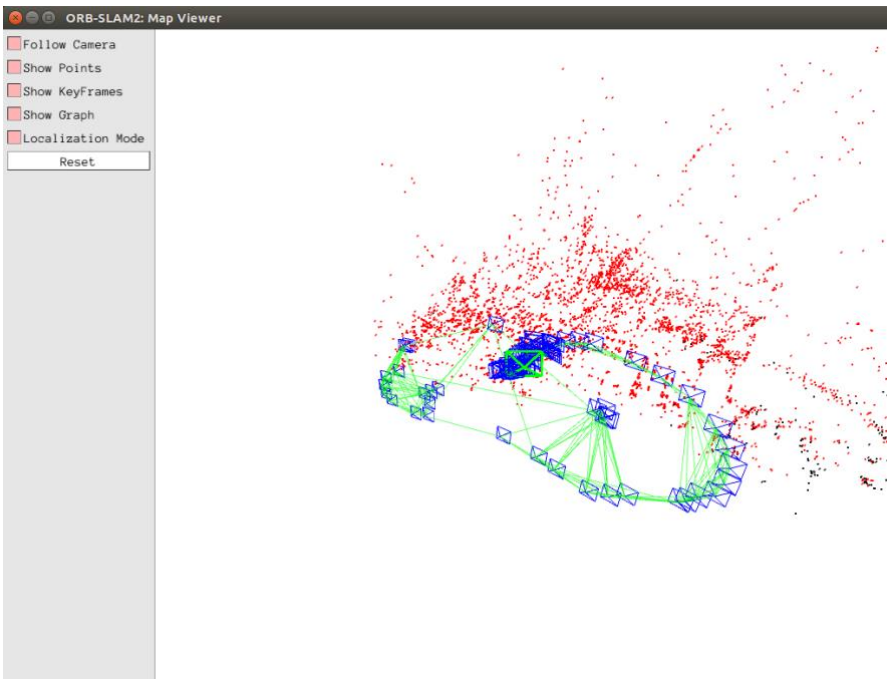

图4-1 ORB-SLAM2 的 Map Viewer

笔者使用紫色的点来在地图上表示规划的路径。

4.2 开发和测试

4.2.1 路径规划模拟测试

笔者在 Windows 系统下使用 Visual Studio 2019 对将要使用在 ORB-SLAM2 上的路径规划算法进行测试。考虑到地图点的数据结构、地图尺度的不确定性以及 ORB-SLAM2 选用的编程语言，笔者没有像一般使用数组来表示地图，而是采用了点集合的方式来表示，用 C++ 语言来编写程序。它的优点是地图的尺度可

以随意调整，如果用二维数组，那么所有的点和障碍都是连在一起的，不能做到点集合的无序和随机；其次是和 ORB-SLAM2 的地图点的数据结构几乎保证了一致性，有利于后续的代码移植工作。

但这样存储地图的缺点也是存在的。如果地图过大而系统定义的栈大小却没有那么大，那么调用递归方法的时候就会报错；方法中有过多指针的调用，一旦出现了空指针也会出错，这也是不熟悉 C 或 C++ 语言的人容易犯的错。因此笔者决定先做模拟测试，调试方法中的错误，再将其运用在 SLAM 系统上。

4.2.2 整合测试

考虑到笔者电脑的重量，并且为了方便调试，笔者决定自行录制视频进行测试。笔者首先使用在曾在家中录制的一段大约不到 20 秒的视频进行测试。

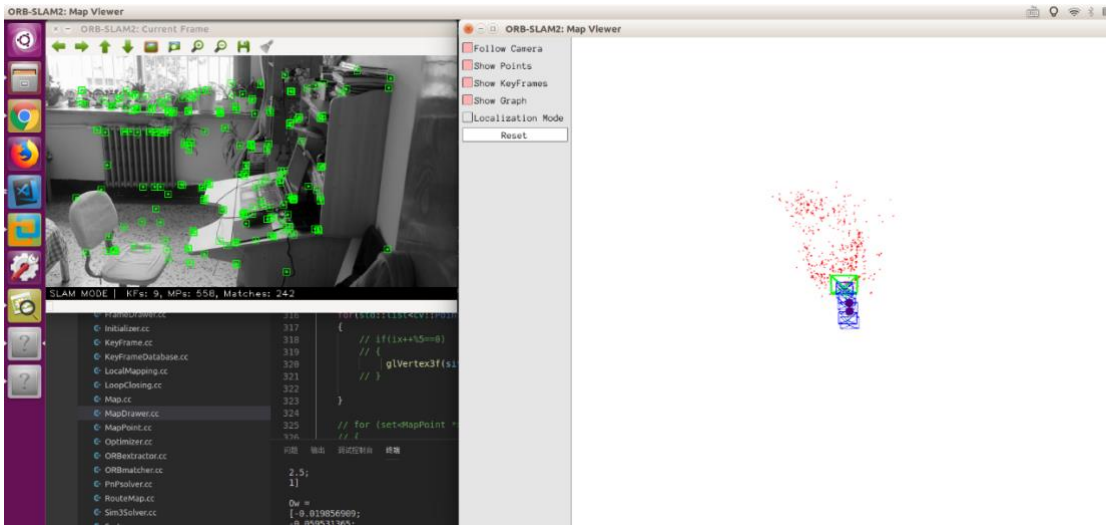

图4-2　程序运行的初始阶段

在运行的最开始，由于所知的特征点比较少，算法找出了一段路径，如图4-2中的紫色点所示。但随着运行持续进行，视频中检测出的特征点越来越多并且相当密集，因此如图4-3 所示，算法并没有找出路径。这是实时路径规划的通病。并且显而易见，终点和起点的位置非常接近。

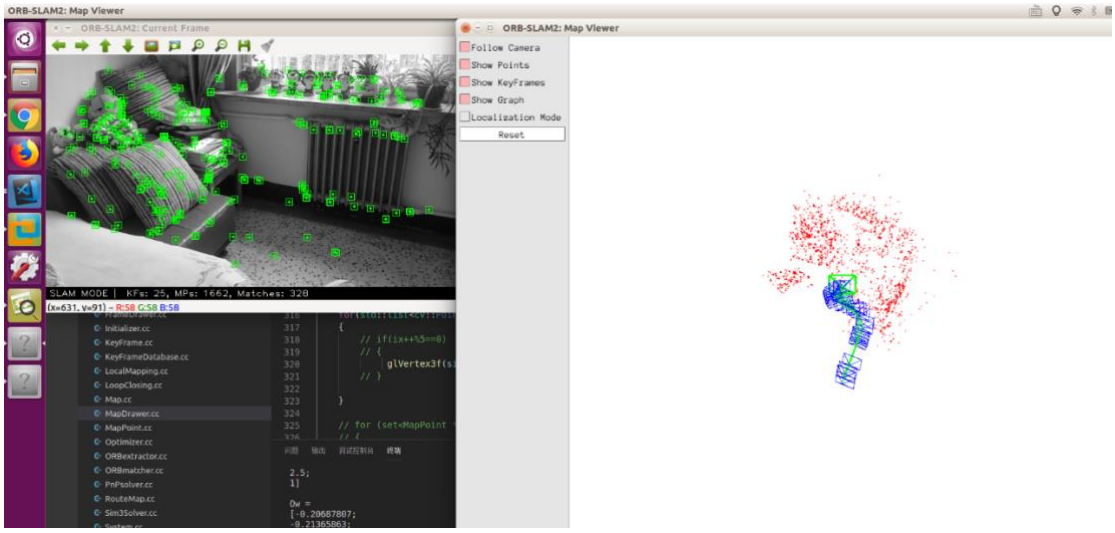

图4-3　程序运行的中后期阶段

其次，在程序运行的后期，如果参考点(如图4-3 中的红色点)太少，或者参考点分布不均匀，就会出现终点位置出现错误等问题。参考点表示在所有的关键帧中能够匹配到对应关系的地图点。程序只提取了参考点作为构图的参照，如果参考点分布不均可能会导致相机光心不在构图范围内等错误出现。

并且，单目视觉 SLAM 的缺点就是如果相机的运动幅度或旋转幅度太大，位姿信息容易出现估计偏差过大等问题，所以会出现一大堆地图点突然失去了和关键帧的连接关系，导致此问题的发生。

笔者在图书馆的阅览区一角录制了一段约 30 秒的视频，重新实验。

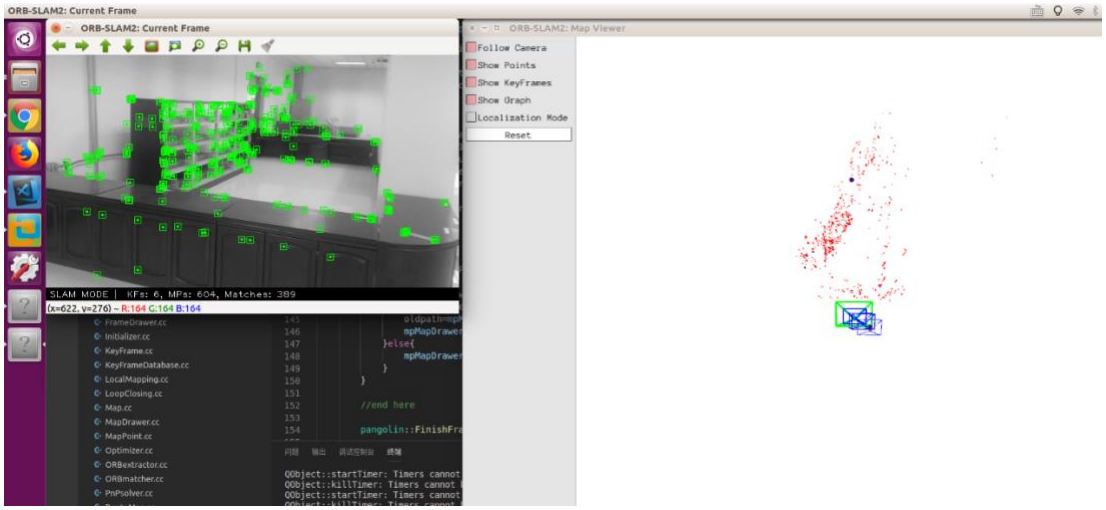

图4-4　在阅览室一角进行的实验

可以见到，规划出的路径出现了一些问题。尽管巧合的是，路径的终点表示的地方确实有一扇门可以通过，但对于此 SLAM 系统来讲却在一个不能达到的地方。之后，笔者行进到门外面，系统立刻出现了跟踪丢失的情况。

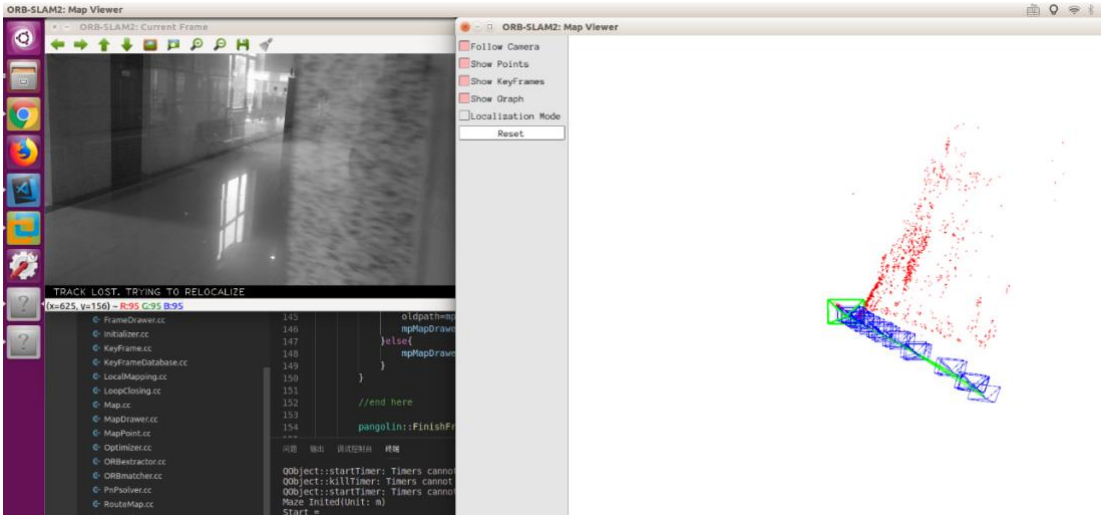

图4-5　出现跟踪丢失(Track Lost)的情况

笔者又在图书馆阅览区的书架之间录制了一段约 30 秒的视频，并再次进行实验。这次实验取得了比较好的效果，可以见到，路径的指向没有太多的偏差。不过由于地图的尺度太小，系统还是只算出了两个点。

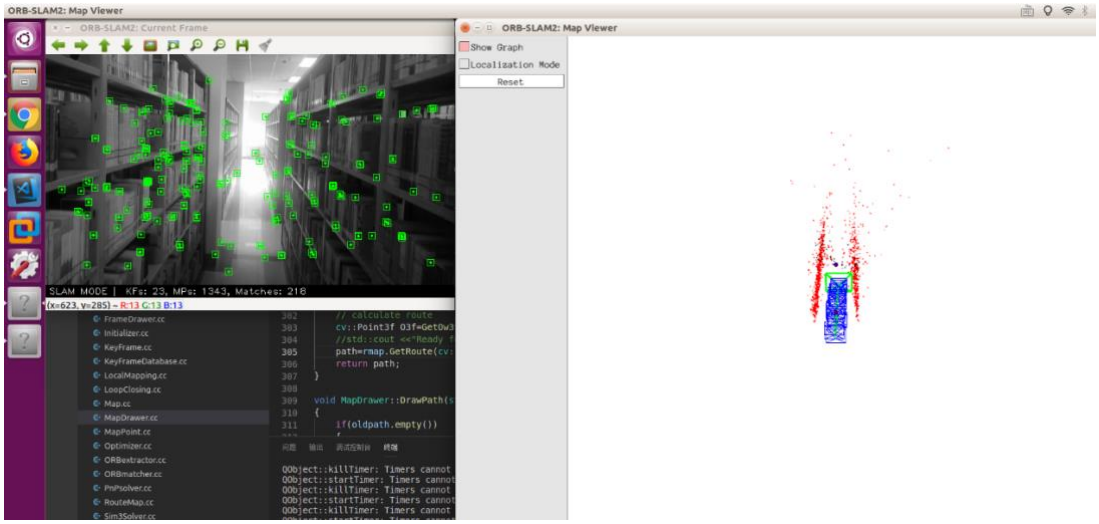

图4-6　在书架之间进行的实验结果

4.3　本章小结

本章，笔者通过录制视频进行了搭载在视觉 SLAM 系统上的路径规划实验。笔者在自己的笔记本电脑上，进行的实验的结果显示，在较为挤逼的空间内，系统很难找到路径；但在较为宽阔的地方是有一定的可能找到路径的。

5 总结与展望

5.1 总结

本文所述的研究通过 ORB-SLAM2 为基底，配合路径规划算法，完成了视觉指导下的路径规划的研究。

首先，系统具有普适性。只要内存大小足够，不论室内或室外条件都可以稳定地运行一段时间。此系统还发挥了 ORB-SLAM2 本身的优势，比如特征提取的速度快、几乎不依赖 GPU 等。

其次，通过一些适当的配置，系统可以在任何有相当性能的计算机上运行，这套系统是没有感情的，因此需要建立感情的步骤全部省略掉了。因此，不再需要和动物一起适应环境、培养感情，后期更不需要面对和动物的生离死别。如果系统出了问题，只需要维修即可。

5.2 未来工作和展望

由于时间仓促，笔者所做的工作还是有一些可见的缺陷产生。

首先，ORB 算法本身的缺陷。它对于没有纹理的物体很难甚至无法提取到特征点，因此会导致有些地方的地图点或路线会出现偏移现象。ORB-SLAM2 对于带纹理的地面会被识别成障碍物，例如笔者家中的地砖等，路径算法进行处理的时候会有些棘手。在一些较为狭窄的空间内，它规划的效果不太理想；在一些比较宽敞的空间里，ORB 很难获得特征点，导致整个系统都不能初始化。

其次，单目 SLAM 的缺陷也比较显著，会出现旋转相机时出现视角偏移、不能移动太快等不足。如果笔者考虑使用双目相机或者 RGB-D 相机，就可以减少这种偏移现象的发生；或者依然继续使用单目相机，再配合其他传感器例如 IMU 等，也可以解决移动速度和精度的问题，实现比较好的效果。

其三，ORB-SLAM2 对计算机运算能力要求较高，因此移动端的应用目前还不太普遍。这主要是因为它独有的三线程程序结构对 CPU 的要求高，而且此 SLAM 系统对每一个帧都会进行运算，对能源的消耗比较大，也需要移动平台 CPU 有着强大的计算能力。鉴于此情况，为了移植到移动设备上，未来笔者可能会选用其他的开源 SLAM 系统进行研究。

其四，笔者使用的 A* 算法，在该 SLAM 系统中运行的效果，并没有想象中的好。笔者未来会针对 SLAM 实时地图的尺度大小进行优化，甚至考虑全新的地图映射方式，来解决由于数据结构导致的尺度问题。此外，系统自动选择可能的终点的方式，也需要一些改良。

最后，ORB-SLAM2 目前用于室内的情况比较多。如果应用于室外就需要和

现有的地图系统进行结合，进行双重导航。未来笔者也会试图将两种导航系统进行结合，发挥各自的功效。目前的算法可能会出现找不到路的情况，这是目前实时路径规划的通病，也亟待解决。

参考文献

[1] STRASDAT H, MONTIEL J M, DAVISON A J. Visual SLAM: why filter?[J]. Image and Vision Computing, 2012, 30(2): 65–77.

[2] LEUTENEGGER S, FURGALE P, RABAUD V, 等. Keyframe-based visual-inertial slam using nonlinear optimization[J]. Proceedings of Robotis Science and Systems (RSS) 2013, 2013.

[3] STRASDAT H, DAVISON A J, MONTIEL J M, 等. Double window optimisation for constant time visual SLAM[C]//2011 International Conference on Computer Vision. IEEE, 2011: 2352–2359.

[4] 霍彦希, 张诚诚, 董振宇, 等. 基于单目视觉 SLAM 的无人快递车设计与实现[J]. 科技传播, 2019, 11(07): 117–118.

[5] 马勋举, 张瑞珠, 马子领, 等. 基于双目全景视觉的无人机避障技术探究[J]. 机电工程技术, 2019(03): 16–19.

[6] 万裕琳. 用于水平地面导航的 VI-SLAM 算法的设计与实现[D]. 华中科技大学, 2019.

[7] 邹谦. 基于图优化 SLAM 的移动机器人导航方法研究[D]. 哈尔滨工业大学, 2017.

[8] MUR-ARTAL R. Real-Time SLAM for Monocular, Stereo and RGB-D Cameras, with Loop Detection and Relocalization Capabilities:raulmur/ORB_SLAM2[M]. https://github.com/raulmur/ORB_SLAM2, 2017.

[9] THRUN S, BURGARD W, FOX D. Probabilistic robotics[M]. MIT press, 2005.

[10] Cartographer is a system that provides real-time simultaneous localization and mapping (SLAM) in 2D and 3D across multiple platforms and sensor configurations.:googlecartographer/cartographer[M]. https://github.com/googlecartographer/cartographer, 2016.

[11] LENG D, SUN W. Finding all the solutions of PnP problem[C]//2009 IEEE International Workshop on Imaging Systems and Techniques. IEEE, 2009: 348–352.

[12] 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.

[13] MORENO-NOGUER F, LEPETIT V, FUA P. Accurate non-iterative o (n) solution to the pnp problem[C]//2007 IEEE 11th International Conference on Computer Vision. IEEE, 2007: 1–8.

[14] HESCH J A, ROUMELIOTIS S I. A direct least-squares (DLS) method for

PnP[C]//2011 International Conference on Computer Vision. IEEE, 2011: 383–390.

[15] GAO X-S, HOU X-R, TANG J, 等. Complete solution classification for the perspective-three-point problem[J]. IEEE transactions on pattern analysis and machine intelligence, 2003, 25(8): 930–943.

[16] 高翔, 张涛, 刘毅, 等. 视觉 SLAM 十四讲: 从理论到实践[M]. 电子工业出版社, 2017.

[17] RUBLEE E, RABAUD V, KONOLIGE K, 等. ORB: An efficient alternative to SIFT or SURF.[C]//ICCV. Citeseer, 2011, 11: 2.

[18] HARRIS C G, STEPHENS M. A combined corner and edge detector.[C]//Alvey vision conference. Citeseer, 1988, 15: 10–5244.

[19] CALONDER M, LEPETIT V, STRECHA C, 等. Brief: Binary robust independent elementary features[C]//European conference on computer vision. Springer, 2010: 778–792.

[20] 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.

[21] FISCHLER M A, BOLLES R C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography[J]. Communications of the ACM, 1981, 24(6): 381–395.

[22] DIJKSTRA E W. A note on two problems in connexion with graphs[J]. Numerische Mathematik, 1959, 1(1): 269–271.

[23] HART P E, NILSSON N J, RAPHAEL B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths[J]. IEEE Transactions on Systems Science and Cybernetics, 1968, 4(2): 100–107.

[24] KOENIG S, LIKHACHEV M. D* Lite[J]. Aaai/iaai, 2002, 15.

[25] LAVALLE S M. Rapidly-exploring random trees: A new tool for path planning[J]. 1998.

[26] LAVALLE S M, KUFFNER JR J J. Rapidly-exploring random trees: Progress and prospects[J]. 2000.

致谢

我是一个不善交际、不善言辞，也不善于表达感情的人，因此可能用任何形式的语言和文字表达我的感激之情都显得比较局促。尽管有很多开心和不开心的事情发生，但对于我来说曾是挑战，而将要如梭消逝的这四年来的大学生活，值得我去感谢和留恋。

首先，感谢我的班主任、以及我的导师仲国强老师的指导。刚来海大的时候我就问仲老师有关计算机科学的专业方向的问题，老师的回答也让我坚定了自己在这条道路上一直走下去的愿望。仲老师给了我很多的指导，有问题的时候老师也会悉心解答。仲老师的声望在学长学姐的中间有口皆碑，我也非常荣幸能跟着仲老师完成今次的毕业论文。

其次，感谢我这四年来在大学认识的朋友，尽管一个手就能数完，但我不会忘记朋友们在我无助、迷茫甚至是伤心、绝望的时刻对我的帮助和爱。感谢好闺蜜刘恬恬，在我开心和不开心的时候带我去吃去玩，有时也会坐在图书馆或者咖啡厅开心地聊天八卦，所以认识你大概是我 20 多年来做过的最为冲动和最不后悔的事情了。感谢李美霞，无论是学习上工作上，还是作为朋友，她都是一个优秀的人，是一个值得交心的朋友；多谢你善待这个曾在准备出国考试时非常迷茫的我，多谢你能够耐心地听我发牢骚。

再次，感谢我前 317 的室友们植志元、尹路、张珂、苏原，现 603 的室友赵树，谢谢你们对我的鼾声的忍耐，谨谢不杀之恩！

最后，感谢父母和家人对我的支持，无论是物质上的也好还是精神上的也好，作为他们的独生子，我感激不尽。我想，我的父母也希望我自己的人生，也像父母为我起的名字那样，像利于航行的一叶扁舟，在自由的大海里遨游。

我还要感谢青岛这座美丽的海滨城市。成年之前没受过多少挫折的我，刚来这里的时候就觉得这里是一个充满着不幸的地方，因为四年之中自己受了不少的打击。不过后来我就开始喜欢这个地方了，大概是自己比较喜欢繁华都市的原因吧。经历着这些挫折的我，成长了很多，也不再怎么觉得这里怎么不幸了。

其实，也要感谢自己，最困难的时候也不自暴自弃；尽管有时候会对生活“小嘴抹了蜜”，但还是不放弃对生活的热忱，对这片土地深沉的爱。