中国科学技术大学

国创成果报告

这里是标题
标题只能有三行
不能多

姓　　名：吴凡迪 王熠 朱思琦
院　　系：少年班学院
导　　师：黄章进 副教授
完成时间：二〇一七年五月

致　谢

感谢中国科学技术大学给我们此次宝贵的机会探索和创新，并提供经费支持，鼓励我们推动课题的进行和发展

感谢黄章进教授对我们项目的指导和大力支持，黄老师的意见和建议给我们的调研及相关工作带来了很大的启发。没有老师的支持就没有这个项目的今天。

感谢项目进行过程中帮助过我们的每一个老师，感谢王莉老师和薛佳老师对项目流程，经费管理的答疑解惑

感谢各位组员克服万难，奋力拼搏，实现了项目目标，成功完成了该份报告。

吴凡迪，王熠，朱思琦

2017 年 5 月 10 日

目 录

摘　要

增强现实是一种在现实场景中无缝地融入虚拟物体或信息的技术，能够比传统的文字，图像和视频等方式更高效、直观地呈现信息，有着非常广泛的应用。同时定位与地图构建作为增强现实的关键基础技术，可以用来在未知环境中定位自身方位并同时构建环境三维地图，从而保证叠加的虚拟物体与现实场景在几何上的一致性。文中首先简述基于视觉的同时定位与地图构建的基本原理并简单比较不同单目同时定位与地图构建技术的优劣，然后详细介绍了基于关键帧的单目相机实时重建的 PTAM 算法，而后探究了 PTAM 算法的改进 ORB-SLAM 算法，并提出了几点在家居设计和增强现实中的应用构想。最后讨论了动态环境下 SLAM 算法的改进，最后讨论近年来 SLAM 的发展趋势，并做总结和展望。

关键词： 同时定位与地图构建 增强现实 多视图几何 动态 SLAM 技术 PTAM 算法 ORB-SLAM 算法

第一章　同时定位与建图技术 (SLAM) 的介绍

1.1　增强现实技术

增强现实技术 (augmented reality) 是一种将真实世界信息和虚拟世界信息“无缝”集成的新技术，是把原本在现实世界的一定时间空间范围内很难体验到的实体信息 (视觉信息, 声音, 味道, 触觉等), 通过电脑等科学技术，模拟仿真后再叠加，将虚拟的信息应用到真实世界，被人类感官所感知，从而达到超越现实的感官体验。比起传统方式来说，它更加的直观，更加的高效，因此也有着更加广阔的应用前景。近年来，增强现实技术已在军事，生活，游戏等众多领域运营并取得了成功。例如，宜家家居公司已经开发了一个 APP 使得用户可以使用智能手机观察不同的家具在自己房间的摆放效果；而任天堂公司也开发了 Pokemon-Go 游戏，使得玩家可以通过智能手机在现实世界里发现精灵。

1.2　增强现实的定位方式

增强现实需要实时定位设备在环境中的方位，定位的方案虽然有许多种，但多数方案都存在局限或者代价太高难以普及，例如 GPS 无法在室内及遮挡严重的环境里使用，且精度较低，而基于无线信号的定位方案则需要事先布置场景。基于视觉的同时定位与地图构建技术（visual simultaneous localization and mapping V-SLAM）以其成本低廉、小场景精度较高、无需预先布置场景等优势成为比较常采用的定位方案。

1.3　V-SLAM 技术

V-SLAM 技术指的是使用图像作为外部信息的唯一来源，来定位一个机器人、一辆车或者一个移动的相机在整个场景中的位置，同时，重建环境的三维结构。

1.3.1　V-SLAM 的基本原理

V-SLAM 技术根据拍摄的视频、图像信息推断摄像头在环境的方位，同时构建环境地图，其原理为多视图几何原理 (Multiple view geometry theroy[?]) V-SLAM 的目标为同时恢复出每帧图像对应的相机运动参数 C_1,C_2 $\cdots C_m$ 以及场景三维结构 X_1,X_2 $\cdots X_n$, 每个相机运动参数 C_i 包含了相机的位置和朝向信息，通常表达为一个 3×3 的旋转矩阵 R_i 和一个三维位置变量 p_i。R_i 与 p_i 将一个世界坐标系下的三维点 X_j 变换至 C_i 的局部坐标系

$$(X_{ij}, Y_{ij}, Z_{ij})^{\mathrm{T}} = R_i(X_j - p_i) \tag{1.1}$$

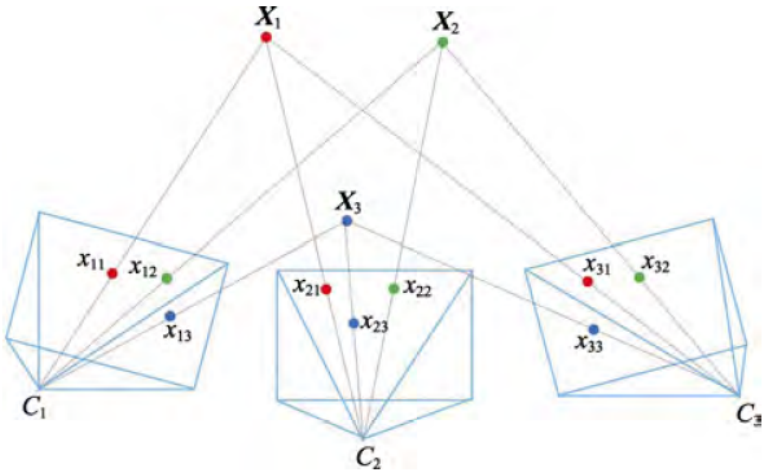

图 1.1　多视图集合的基本原理示意图

进而投影至图像中

$$h_{ij} = (f_x X_{ij}/Z_{ij} + c_x.f_y Y_{ij}/Z_{ij} + c_y)^{\mathrm{T}} \tag{1.2}$$

其中，f_x,f_y 分别为沿图像 x,y 轴的图像焦距,(c_x, c_y) 为镜头光心在图像中的位置，通常假设这些参数已实现标定且保持不变，由式 式 1.1 式 1.2，三维点在图像中的投影位置 h_{ij} 可表示为一个关于 C_i 和 X_j 的函数，记为

$$h_{ij} = h(C_i, X_j) \tag{1.3}$$

V-SLAM 算法需要将、对不同图像中对应于相同场景的图像点进行匹配，而这个过程是通过求解如下目标函数

$$\underset{C_1,C_m,X_1,X_n}{\arg\min} \sum_{i=1}^{m}\sum_{j=1}^{n} \|h(C_i, X_j) - \tilde{x_{ij}}\|_{\Sigma_{ij}} \tag{1.4}$$

得到一组最优的 C_1，$C_2 \cdots C_m$,X_1,X_2 $\cdots X_n$，使得所有 X_j 在 C_i 图像中的投影位置 h_{ij} 与观测到的图像点位置 x_{ij} 尽可能靠近，这里假设图像观测点符合高斯分布 $x_{ij} \sim \mathrm{N}\ (\tilde{x_{ij}},\Sigma_{ij})$，$\| e \| = e^{\mathrm{T}}\ \Sigma^{-1}\ e$ 求解目标函数式 1.4的过程也成为集束调整（bundle adjustment,BA），该最优化问题可利用线性方程的稀疏结构高效求解。

1.3.2　主流的单目 V-SLAM 方法

由于现阶段大多数 AR 产品都以智能手机以及平板电脑作为载体，而智能手机的摄像头大多以单目为主，双目、三目摄像头甚至深度摄像头都未得到普及，因此本文主要讨论基于单目视觉的同时定位与地图构建方法。目前，主流的 V-SLAM 方法主要为：基于滤波器、基于关键帧 BA 和基于直接跟踪，我们先来看看这三种方法。比较并分析其优劣，而后详细介绍基于关键帧 BA 的

V-SLAM 方法。其中比较具有代表性的有 MonoSLAM 以及 MSCKF 基于滤波器的 V-SLAM 的方法将系统每一时刻的状态 t 用一个高斯概率模型表达，$x_t \sim$ N $(\tilde{x}_t, \mathrm{P}_{ij})$, 其中 $\tilde{x}_t$ 为当前时刻系统状态估计值，P_t 为该估计值误差的协方差矩阵，系统状态由滤波器不断更新。而基于关键帧 BA 的 V-SLAM 方法 [?] 是近年来最流行的方法之一，他的主要思想是将相机跟踪（Tracking）和地图构建（Mapping）作为两个独立的任务在两个线程并行执行，而 Mapping 线程仅维护视频流中抽取的关键帧。PTAM 是最著名的基于关键帧 BA 的方法之一，也是我们介绍的重点基于直接跟踪的 V-SLAM 方法则是直接通过比较像素颜色来求解相机运动，具有代表性的算法有 DTAM 以及 LSD-SLAM。

1.3.3 不同 V-SLAM 方法的比较

比较三种方法，基于直接跟踪的 V-SLAM 方法对光照变化和动态干扰较为敏感，所以精度会若于基于关键帧 BA 的 V-SLAM 方法，而基于滤波器的 V-SLAM 由于容易在变量未精确的时候就进行消元，误差不断累积，因此精度最差。

而在定位效率与场景尺度方面，基于滤波器的 V-SLAM 方法计算复杂度较高，只适用于几百个点的小场景，因此定位效率和场景尺度较低，而基于关键帧的 V-SLAM 方法例如 PTAM，它只优化当前帧方位就因此具有最高的定位效率，但场景尺度受到 BA 的效率限制因此也不高，而基于直接跟踪的 V-SLAM 方法定位效率取决于图像，分辨率，且采用更高效的方位图优化替代全局 BA，因此具有最高的场景尺度限制。

而在近视纯旋转扩展鲁棒性上，由于在实际使用中，相机会近似纯旋转的方式转向拍摄旁边的场景内容，因此基于滤波器的 V-SLAM 方法每帧同时优化三维点和相机方位，因此对近似纯旋转具有很好的鲁棒性。而基于关键帧 BA 的 V-SLAM 方法在近似纯旋转扩展场景时容易因视差不够无法三角化新的三维点，从而无法加入新的关键帧导致关键帧丢失，鲁棒性较差。而基于直接跟踪的 V-SLAM 方法的鲁棒性则取决于后台场景地图的扩展及优化的效率，因此鲁棒性较好。

而对于场景变化的鲁棒性来说，无论哪种 V-SLAM 方法，都假设场景是静止不变的，如果场景变化很大都会导致跟踪的失败。因此对于场景变化的鲁棒性都不佳。在第四节中，我们会专门讨论运动情况下的 V-SLAM 方法。

总的来说，基于关键帧的方法具有更好的 V-SLAM 效果。

第二章　基于 BA 的单目相机实时重建的经典算法 (PTAM) 介绍

本章节将重点讨论基于 BA 的单目相机实时重建经典算法:PTAM。该算法是基于关键帧特征和 BA 的稀疏重建算法的开山鼻祖，具有重要的意义。

2.1　PTAM 算法的介绍

首先我们将简述 PTAM[?](Parallel Tracking and Mapping) 实现单目相机 SLAM 的原理。单目相机模型 (Monocular SLAM) 不同于多目相机模型，实时追踪时相机视界中的点不能和其它相机视界中的点进行匹配或者实现三角定位，只能和自己的关键帧匹配，从而加大了 3D 重建中定位 (即深度信息求解) 的难度。一般的 PTAM 算法首次提出利用双线程，将全局 BA 和相机姿态估计这两个任务分离，从而大幅提高了计算效率，使得单目相机实时追踪特征点，实现三维重建成为可能。

PTAM 算法主要思想是将 Tracking 和 Mapping 两个过程放在不同的管线 (进程) 中进行: Tracking 进程专门实现相机位置的估计，Mapping 进程则用于进行关键帧之间的误差消除。

2.2　PTAM 算法工作的基本流程

如果记 W 为真实世界的坐标系,PTAM 算法将维护一个关键帧集合: $Img = \{I_1, I_2, \ldots, I_m\}$，这 m 个关键帧分别对应 m 个相机坐标系 K_i，以及一个三维重建的结果。我们用 E_{K_iW} 表示从世界坐标系到相机坐标系的仿射变换 (Affine Transformation)。

2.2.1　PTAM 管线之一: Tracking

追踪进程需要解决如下的问题：

当读入了新的关键帧之后，原来算法在重建过程中提取的 3 维空间中的特征点现在在照片中的坐标是什么？现在的相机姿态应该怎么估计？

我们假定程序可以从映射进程得到一个关键帧集合 Img 以及 3 维重建的特征点的坐标集合 (相对于世界坐标系)$P = P_W = \{\mathbf{p}_{1W}, \ldots, \mathbf{p}_{sW}\}$，为了统一形式，将第 j 个点坐标记为 $\mathbf{p}_{jW} = (p_{jx}, p_{jy}, p_{jz}, 1)$。

根据已知结论，仿射坐标系变换对应公式为

$$\mathbf{p}_{jK_{t+1}} = E_{K_{t+1}W}\mathbf{p}_{jW} \tag{2.1}$$

为了把三维空间中的视界投影到二维空间，算法遵循 [?] 中的 FOV 相机模型，构建一个 $\mathbb{R}^3$ 到 $\mathbb{R}^2$ 的映射为:

$$f(x,y,z,1) = (u_0,v_0) + (x/z, y/z)\begin{bmatrix} f_u & 0 \\ 0 & f_v \end{bmatrix}\frac{r'}{r} \tag{2.2}$$

$$r = \sqrt{\frac{x^2+y^2}{z^2}} \tag{2.3}$$

$$r' = \frac{1}{\omega}arctan(2rtan\frac{\omega}{2}) \tag{2.4}$$

其中我们假定焦距 f_u, f_v，主点位置 (u_0, v_0) 和畸变系数 ω 已知。这时对于实时相机姿态的更新，相当于对于 式 2.2 求微分，在假定线性运动的情况下更新相机姿态的变换仿射矩阵。若我们设从上一关键帧到下一个关键帧的仿射变换矩阵为 T，则有以下的关系成立:

$$E_{K_{t+1}W} = TE_{K_tW} = exp(\mu)E_{K_tW} \tag{2.5}$$

其中 μ 为 6 维向量，代表矩阵 T 的 6 个自由度。所以问题转化为: 根据图像中特征点 (u,v) 的变化和在每个特征点三维空间中的预估位置，求解 μ, T 的取值。

我们假定点 p 为我们需要定位的特征点，则我们首先需要在当前的关键帧图像中找到该点的新投影坐标 $(\hat{u},\hat{v})$。为此，我们构造图像的尺度空间，利用的 FAST-10[?] 计算角点的方法提取可能的特征点。在假定相机移动很慢的情况下，特征点匹配算法从上一帧的位置开始，在一定的半径阈值内，在对应的视界空间上，依据设计好的评价函数 (scoring function) 进行搜索。算法会在搜索过程中得到一个坐标 $(\hat{u},\hat{v})$，以及 $\sigma = 2^l$ 作为视界空间金字塔中搜索对应的层数。

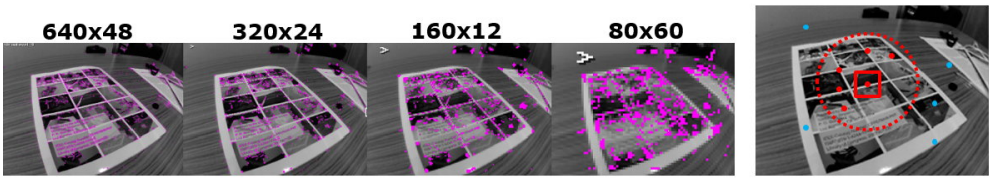

图 2.1 视界空间中利用 FAST 提取特征点和搜索对应特征点示意图，左图表示视界空间的 down sampling，以及基于 FAST 方法的特征点选取，右图为在对应的尺度空间中搜索对应的特征点

在得到所有特征点的信息之后，算法求解以下的优化问题计算相机姿态的更新 (设 P_{t+1} 为当前特征点集合):

$$\underset{\mu}{argmin} \sum_{j\in P_{t+1}} \psi(\frac{\|\mathbf{e}_j\|}{\sigma_j}, \sigma_T) \tag{2.6}$$

$$\mathbf{e}_j = (\hat{u}_j, \hat{v}_j) - f(exp(\mu)E_{K_tW}\mathbf{p_j}) \tag{2.7}$$

这里 ψ 为 Tukey loss[?]:

$$\psi(x,c) = \begin{cases} x(1-\frac{x^2}{c^2}) & \text{for } |x| < c \\ 0 & \text{for } |x| > c \end{cases} \tag{2.8}$$

为了保证算法的稳定性，追踪进程会进行两次：第一次会从三维模型中抽取 50 个特征点投影匹配，第二次则抽取 1000 个特征点进行匹配。同时，如果在特征点投影搜索配对的过程中，如果特征点搜索失败的比率大于某一个阈值的话，算法将会判定这一帧失效，并且自动舍弃（这可能是由于抖动，位移量过大特征点不在视界中等多种原因造成）。

2.2.2 PTAM **管线之二**: Mapping

在追踪进程持续运行的同时，映射进程会根据追踪进程估计的相机姿态，完成关键帧的选取及三维重建的主要任务。为此，算法首先在初始化阶段，需要用户指定视频流中的两帧 (第一帧一般为起始帧)，单独进行第一次特征点配对。这样做的目的有两个，一方面是为了利用五点法求解相机模型中的固有参数，另一方面是为了先定位特征点到三维空间中作为初始信息。

2.2.2.1 **关键帧的选择和插入**

在程序进程中，当视频流被读入之后，对于每一帧图像，算法需要判断是否需要把该帧图像加入关键帧集合。判断主要依据是距离上一关键帧的时间和预估的相机距离，以及图像的质量。在选定关键帧之后，根据追踪进程返回的信息，结合上一关键帧的对应特征点信息，定位深度，从而把特征点加入到三维重建模型中。

2.2.2.2 **利用** BA(Bundle Adjustment) **校正**

为了消除帧与帧之间的积累误差，映射进程还需要对整体的投影误差和进行一个优化，即所谓的 BA。算法对于所有关键帧求解一次优化问题，也会对局部的几个特征点求解局部的优化问题，它们有以下形式:

$$\underset{\{\mu\},\{\mathbf{p}\}}{argmin} = \sum_{i=1}^{N}\sum_{j\in P_i} \psi(\frac{\|\mathbf{e}_j\|}{\sigma_j}, \sigma_T) \tag{2.9}$$

这里的计算将会采用 Levenberg-Marquardt BA 算法 (详见 [?])。在完成校正之后，新的特征点的三维信息和关键帧信息将作为追踪进程的输入，传入 Tracking 进程，两个进程在工作的时候将会保持相互的通信，这样最大程度上保证算法运行的流畅度。上述算法的核心过程，在下方图 2.2中进行了简要的总结。

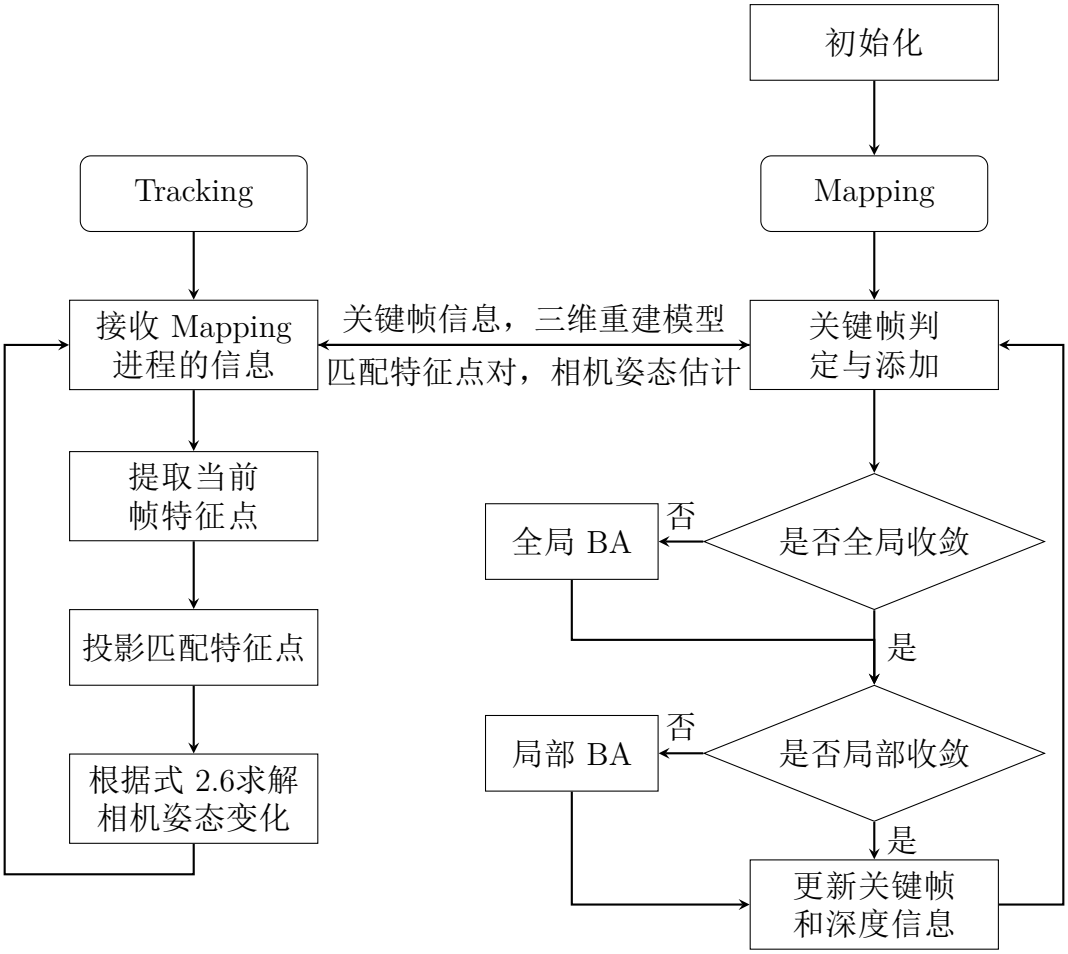

图 2.2 基于 BA 的单目经典算法 PTAM 的流程示意图

2.3 PTAM 算法的实际效果

本节重点比较和讨论对于 PTAM 算法运行效果。

2.3.1 在一般电脑上运行

在一般的笔记本电脑上运行的结果，如下图所示。对于开放的场景 (如寝室内部)，实际测试中帧率几乎达不到 30 帧/秒，能够在 20 帧每秒的情况也比较少见。对于特定的平面场景 (如桌面)，实际测试结果。

2.3.2 在智能手机上运行

智能手机的性能会是制约此类算法在手机上表现的一个重要因素，[?] 中尝试把 PTAM 的系统移植到 Iphone 中，得到了图 2.3所示的结果。

2.3.3 PTAM 算法的评价

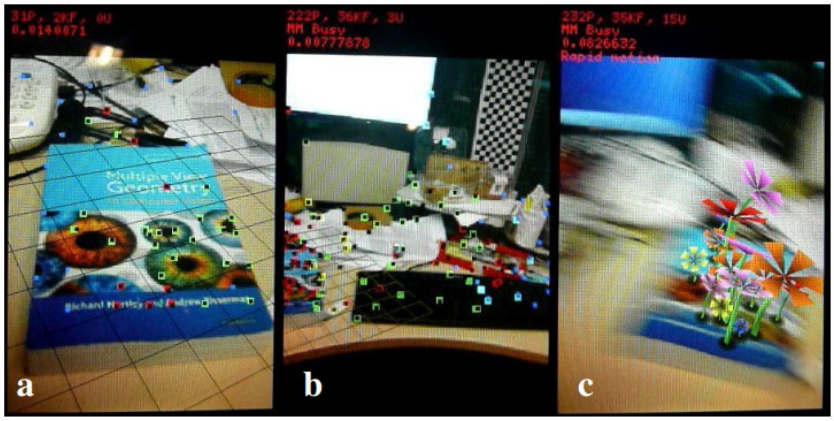

图 2.3 在智能手机上运行的结果，(a) 图为起始状态，(b) 图表现了特征点的追踪，(c) 图表示了在运动过程中的平面捕捉与 AR 渲染。由于手机运算能力限制，特征点数量较 PC 端大幅减少

第三章　ORB-SLAM 在家居设计和增强现实中的应用构想

3.1　ORB-SLAM 方法简述

ORB-SLAM [?] 是近几年比较火热的单目相机的 SLAM 方法，它在 PTAM 的基础上采用了效率更高的特征提取和优化算法，改进了 PTAM 的双进程为三进程，加入闭环检测 (loop closing)，并优化了代码结构，使得该算法较 PTAM 更稳定，更实用，也更有移植到移动端的潜力。

该算法的流程如图 3.1所示，算法主要维护三个进程：追踪进程 (Tracking) 与 PTAM 类似，主要用于相机姿态的估计，但不同的是采用了比 FAST 更优的 ORB [?] 特征点提取方法，同时改进了关键帧的判定。局部映射进程 (Local Mapping) 则与 PTAM 中的映射进程类似，在有了新的关键帧之后，算法需要维护关键帧集合，构建重建信息，同时进行 BA 优化。ORB-SLAM 算法维护的第三个进程，即闭环检测进程 (loop closing)，利用了 DBoW2[?] 技术，通过构建词典检测图片序列中的语义场景，从而判定相机轨迹中是否存在闭路的情形，同时对检测到的闭路进行优化，提高追踪精度并减少重复计算。

为了降低全局的时间复杂度，ORB-SLAM 维护一个基于关键帧到关键帧的图结构 (Covisibility Graph) [?] ，用图模型中边的权值 (共享特征点数量) 衡量关键帧的相似度，同时取稠密图中的 MST(Essential Graph) 作为子图实际参与优化，在保证鲁棒性的前提下尽可能提高效率。

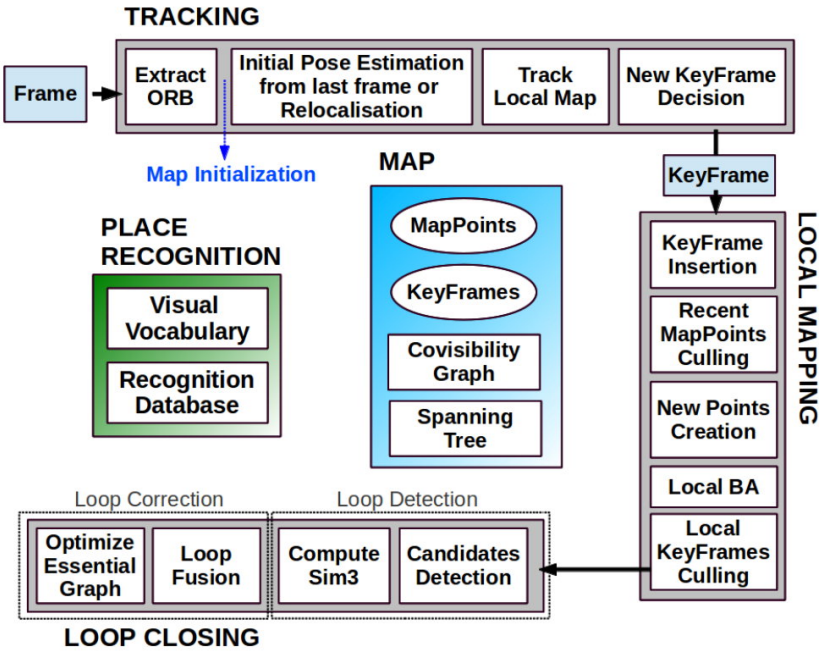

图 3.1　ORB-SLAM 算法的主要流程示意

第四章　动态环境中的 SLAM 算法介绍

实际应用中我们可能会遇到对于拍摄的物体有运动的情况，此时我们首先要区分出静态物体和动态物体，进一步再刻画两者的位置关系，两者的位置需要交错迭代来确认。另外为了提高位置的准确性，我们还需要提供标定点来给相机参考 [?]，或由相机位置计算出标定点 [?]，这里我们采用角落作为标定点。过程中使用两个集合 S 和 D，分别表示格点被静态部分和动态部分占用的概率，$S \cup D$ 则可表示出可推测部分的概率。当 S 或 D 中的某一个对应概率超过 0.9 时, 我们认为该位置是可以确定的。

4.1　物体类型的映射的迭代

记 o_t 为 t 时刻物体位置的深度信息,则我们希望求出 $P(S_t|o_1,,o_t), P(D_t|o_1,,o_t)$。为了得到迭代过程，我们期望 $P(S_t)$ 和 $P(D_t)$ 依赖于 S_{t-1}, D_{t-1}, 而由于 D_{t-1} 中信息会发生变化，我们在迭代过程中不利用 D_{t-1} 中的信息。即我们希望求出 $P(S_t|o_1,,o_t,S_{t-1}), P(D_t|o_1,,o_t,S_{t-1})$.

4.2　映射集 S 的迭代

利用 Bayes 公式, 可以得到

$$P(S_t|o_1,\cdots,o_t,S_{t-1}) = \frac{P(o_t|o_1,\cdots,o_{t-1},S_{t-1},S_t)P(S_t|o_1,\cdots,o_{t-1},S_{t-1})}{P(o_t|o_1,\cdots,o_{t-1},S_{t-1})}$$

在 $P(o_t|o_1,\cdots,o_{t_1},S_{t_1},S_t)$ 中，我们考虑 S_{t-1}, S_t 已经包含了 $\{o_1,\cdots o_t\}$ 中的所有信息，即 $P(o_t|o_1,\cdots,o_{t-1},S_{t-1},S_t) = P(o_t|S_{t-1},S_t)$，再次利用 Bayes 公式，可得

$$P(S_t|o_1,\ldots,o_t,S_{t-1}) = \frac{P(S_t|o_t,S_{t-1})P(o_t|S_{t-1})P(S_t|o_1,\cdots,o_{t-1},S_{t-1})}{P(S_t|S_{t-1})P(o_t|o_1,\cdots,o_{t-1},S_{t-1})}$$

类似可以求出 S 未占据格点个条件概率 $P(\widetilde{S})$,

$$P(\widetilde{S}_t|o_1,\ldots,o_t,S_{t-1}) = \frac{P(\widetilde{S}_t|o_t,S_{t-1})P(o_t|S_{t-1})P(\widetilde{S}_t|o_1,\cdots,o_{t-1},S_{t-1})}{P(\widetilde{S}_t|S_{t-1})P(o_t|o_1,\cdots,o_{t-1},S_{t-1})}$$

上两式相除，可得

$$\frac{P(S_t|o_1,\ldots,o_t,S_{t-1})}{P(\widetilde{S}_t|o_1,\ldots,o_t,S_{t-1})} = \frac{P(S_t|o_t,S_{t-1})}{P(\widetilde{S}_t|o_t,S_{t-1})}\frac{P(\widetilde{S}_t|S_{t-1})}{P(S_t|S_{t-1})}\frac{P(S_t|o_1,\cdots,o_{t-1},S_{t-1})}{P(\widetilde{S}_t|o_1,\cdots,o_{t-1},S_{t-1})}$$

由于 $P(S) = 1 - P(S')$, 上式可写为

$$\frac{P(S_t|o_1,\ldots,o_t,S_{t-1})}{1-P(S_t|o_1,\ldots,o_t,S_{t-1})} = \frac{P(S_t|o_t,S_{t-1})}{1-P(S_t|o_t,S_{t-1})}\frac{1-P(S_t|S_{t-1})}{P(S_t|S_{t-1})}\frac{P(S_t|o_1,\cdots,o_{t-1},S_{t-1})}{1-P(S_t|o_1,\cdots,o_{t-1},S_{t-1})}$$

由于 $P(S_t|S_{t-1})$ 不依赖于 t, 可记为 $P(S)$, 另外，由于 S^t 在给定 $o_1, \cdots, o_{t-1}, S_{t-1}$ 时，不发生改变，因此上式可写为

$$\frac{P(S_t|o_1,\ldots,o_t,S_{t-1})}{1-P(S_t|o_1,\ldots,o_t,S_{t-1})} = \frac{P(S_t|o_t,S_{t-1})}{1-P(S_t|o_t,S_{t-1})}\frac{1-P(S)}{P(S)}\frac{P(S_{t-1})}{1-P(S_{t-1})}$$

为便于计算，上式中取 $P(S) = 0.5$，可得

$$\frac{P(S_t|o_1,\ldots,o_t,S_{t-1})}{1-P(S_t|o_1,\ldots,o_t,S_{t-1})} = \frac{P(S_t|o_t,S_{t-1})}{1-P(S_t|o_t,S_{t-1})}\frac{P(S_{t-1})}{1-P(S_{t-1})}$$

上式即可完成迭代过程。

4.3 映射集 D 的迭代

类似于 S 的计算过程，我们可以得到

$$\frac{P(D_t|o_1,\ldots,o_t,S_{t-1})}{1-P(D_t|o_1,\ldots,o_t,S_{t-1})} = \frac{P(D_t|o_t,S_{t-1})}{1-P(D_t|o_t,S_{t-1})}\frac{P(D_{t-1})}{1-P(D_{t-1})}$$

由于 D 中的物体会随时间变化，迭代过程不需要依赖于 D_{t-1}, 计算时也不需要考虑 D 中的过去值，而只需要考虑当前的 t 来计算当前移动物体的位置情况。仅当 $P(S_{t-1})$ 未确定且 Po_t 较大时，$P(D_t)$ 会有较大的取值。[?]

4.4 定位

由于动态物体位置的确认依赖于原来区域的选取，因此静态位置误差会导致对动态物体估计的误差。位置标定的方法有很多种算法，如我们下面使用的地标法。若对环境有预先的信息，则这里可以在给定的信息中得到标定点，否则，要同时利用相机位置和图片信息 [?], 可以选取角作为标定点 [?]，为了避免标定点的相似影响结果，还需要相机的位置来辅助确定位置，并删去最近的标定点。这里除了上述的两种格点映射集，我们还需要第三类映射集来储存标定点的位置。同样，标定点位置可能会被动态物体影响，特别是将动态物体设置为标定点时，因此必须区分出静态标定点。我们在选取标定点时，必要的要从 S 中大概率的点来选取。