Visual SLAM Learn Notebooks Chapter 1

Introduction

First Author: Taolinyinjiu

Version: 1.0

我这样死板的山，也会为你哗然

Foreword

仅以笔记，记录那些陷入低谷的日子，以及致不曾放弃的自己

Preface

该笔记主要讲记录了个人在学习视觉 SLAM 中所遇到的各种问题，以及对应的解决方法，主要的参考资料为高翔老师的视觉 SLAM 十四讲

Abstract

Introduction 中主要介绍了 SLAM 以及视觉 SLAM 的发展历史，主要变化，特征结构，以及对学习路线的指导

Key words

Visual Simultaneous Localization and Mapping

Table of Contents

表格

插图

1. 预备知识

1.1 什么是 SLAM？

SLAM，全称为 Simultaneous Localization and Mapping，中文翻译为即使定位与地图构建。它是指搭建特定传感器的主体，在没有环境先验信息的情况下，于运动过程中建立起环境的模型，同时估计自己的运动。如果在这个过程中，主要使用的传感器为相机，则称为视觉 SLAM，也就是 Visual Simultaneous Localization and Mapping

对 SLAM 的定义中提到了大量的名词，或者说专业术语，也可以说是暂时对大家晦涩的定义。这是我们刻意而为之的，我们希望大家对于 SLAM 有一个明确的概念。首先，SLAM 的目的是同时解决”定位”和“地图构建“两个问题。也就是说，一边要估计传感器自身的状态，一边要建立起周围环境的模型。但是怎么做到呢？这就需要用到传感器的信息。传感器能够以一定的形式去观察外部的世界，不过不同的传感器所观察的方式是不同的。而我们将会花很大的篇幅去讨论与之相关的问题，就是因为它很难，特别是在我们希望能够实时的，在没有先验知识的基础上去进行 SLAM 的情况下。当我们以相机为主要的传感器去解决 SLAM 问题时，我们要做的就是根据相机得到的一张张连续的图像去估计相机的运动和周围的情况。

1.2 SLAM 的发展历史

在计算机视觉 (Computer Vision) 创建之初，人们就想象着有朝一日，计算机将同人类一样，通过眼睛 (摄像头) 去观察世界，理解周围的物体，探索未知的世界–这是一个伟大又浪漫的梦想，它吸引了无数的人夜以继日，前仆后继地做出无与伦比的贡献。我们曾以为这件事情并不困难，然而现实的进展远不如预想的那么顺利。我们眼中的花草树木，虫鱼鸟兽，在计算机中只是一个个由数字所排列而成的矩阵 (Matrix)。让计算机理解图像的内容，就像让我们自己来理解这些数字的内容一样的困难，我们既不了解自己如何了解图像，也不理解计算机该如何理解，探索这个世界。于是这个领域的发展停滞了很久，直到多年以后，才有了一点成功的迹象: 随着人工智能 (Artifical Intelligence -> AI) 和机器学习 (Machine Learning -> ML)，计算机渐渐的能够辨别出物体，人脸，声音，文字，尽管他们所使用的方式 (通过概率学建模) 与我们是如此的不同，但至少已经有了一份可能。另一方面，SLAM 发展了近 30 年之后，我们的相机才开始能够逐渐认识到自身的位置，发觉自己处于运动当中，即使方式也同人类有着巨大差异。不过，研究者们已经搭建出了各种的实时 SLAM 系统，有的可以快速跟踪自身位置，有的可以做到实时的三维重建。

自 1986 年提出以来，SLAM 一直是机器人领域中的热点问题，关于它的文献数以千计，想要对 SLAM 发展史上的算法做出一个完整的说明，就我的水平而言是不太可能的。笔记中会涉及到的内容主要是 SLAM 十四讲中涉及到的内容，以及部分机器人学中的状态估计，机器人学中的 SLAM 等一系列由高翔老师参与 (我是高翔老师的小迷弟) 的书籍中所涉及到的内容。

1.3 SLAM 的参考资料

目前，与 SLAM 有关的参考资料主要有，概率机器人，计算机视觉中的多视图几何，机器人学中的状态估计等

2. 初识 SLAM

2.1 引子

假设我们组装了一台叫做” 小萝卜” 的机器人，大概样子如下图所示

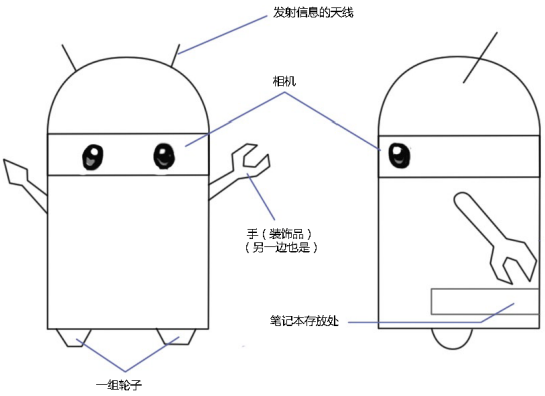


Fig. 1. 小萝卜设计图 (从 SLAM 十四讲书中直接摘的嘻嘻)

虽然他长得或许有点像 Android 的图标 (或者叫吉祥物)，但它并不是依赖于安卓系统来进行计算的，我们在他的身体里安装了一台边缘计算设备，通常会搭载 Linux 系统，比如 Ubuntu。

接着我们希望” 小萝卜 “能够拥有自主移动的能力，虽然世界上也有放在桌面上当摆件的机器人，但我们还是希望” 小萝卜 “能够在房间里四处走动，不管我们在哪里打招呼，他都能自己移动过来。小萝卜想要移动，就需要有轮子和电机，所以我们在小萝卜的下面安装上了轮子，有了轮子，小萝卜就可以四处的移动了，但是如果不加以控制的话，小萝卜就不知道该往哪里走，如果更糟糕一点，前面是一堵墙，而小萝卜毫不犹豫的就撞了上去的话，就会造成损坏。为了避免这种情况的发生，我们在小萝卜的脑袋上安装了一个相机。

安装相机的主要动机，是考虑到小萝卜和人类非常的相似 (从画面上一眼就能看出来)，有眼睛，大脑和四肢的人类，可以在任意环境中轻松的行走，探索，我们也天真的认为，机器人也能够完成这件事情。为了使小萝卜能偶探索一个房间，它至少需要知道两件事：

- 我在什么地方? –> 定位
- 周围的环境是什么样的 –> 建图

”定位”和“建图”，可以看成感知的”内外之分“；作为一个“内外兼修”的小萝卜，一方面要明白自身的状态 (也就是位置), 一方面也要了解外在的环境 (也就是地图)。当然，解决这个问题的方法非常的多，比如我们可以在房间上铺设导引线，在墙壁上贴上二维码，在桌子上放置 RTK 定位系统。如果在室外环境，还可以在小萝卜的脑袋上安装 GPS 定位设备。

我们不妨把这些传感器分为两类，一类是安装在机器人本体上的，例如机器人的轮式编码器，相机，激光传感器等等；另一种是安装与环境中的，例如导轨，导引线，二维码等。安装在环境中的传感设备，通常能够直接测量到机器人的位置信息，简单有效的解决了机器人的定位问题，但是由于他们必须设置在环境中，因此也限制了机器人的使用范围。

我们会发现，这些安装在机器人上的传感器约束了外部环境，只有在外部环境满足他们的要求时，才能够使用基于它们的定位方案。因此我们说，这些传感器提供了简单有效的定位方案，但是无法给出一套普遍的，通用的解决方案。相对的，那些安装在机器人本体上的传感器，比如激光传感器，相机，轮式编码器等，他们测量的都是间接的物理量而不是直接的位置数据，比如 IMU(惯性测量单元) 测量的是线加速度和角加速度，轮式里程计测量的是轮子转过的角度。通过这样的一些数据，来估计机器人的位置和姿态，虽然看上去来说像是一种迂回战术，但相应的，他们没有对环境提出任何要求，因此这种方案得以适应未知的环境。

视觉 SLAM 是我们所讨论的主题，所以我们尤其关心小萝卜的眼镜能够做什么事。SLAM 中使用的相机和我们平时见到的单反摄像头并不是同一个东西，它往往更加简单，不携带昂贵的摄像头，而是以一定的速率拍摄周围的环境，形成一个连续的视频流。普通摄像头能以每秒 30 张图片的速度采集图像，而高速相机则更快一些。

按照工作方式的不同, 相机可以分为单目相机 (Monocular), 双目相机 (Stereo) 和深度相机 (RGB-D) 三大类。直观的来说，单目相机只有一个摄像头，双目有两个，而 RGB-D 的原理比较复杂，除了能够采集到彩色图片外，还能读出每个像素和相机之间的距离。深度相机通常携带多个摄像头，工作原理也和普通的相机不尽相同。此外，在 SLAM 中还有全景相机，Event(事件) 相机，鱼眼相机等许多的种类。

2.1.1 单目相机

只是用一个摄像头进行 SLAM 的做法称为单目 SLAM(Monocular SLAM)。这种传感器的结构特别简单，成本特别低，所以单目 SLAM 特别受到研究者的关注。单目 SLAM 使用的数据是什么呢? 照片，是的，作为一张照片，他有什么特点呢?

照片本质上是拍照时的场景 (Scene) 在相机的成像平面上留下的一个投影。它以二维的形式反映了三维的世界。显然，这个过程丢失了场景的一个维度，也就是我们常说的深度 (或者距离)。在单目相机中，我们无法通过单张图像来计算场景中的物体与我们之间的距离 (远近)，而这是 SLAM 中非常关键的信息。由于人类见过大量的图像，形成了一种天生的直觉，对大部分场景都有一个直观的距离感 (或者叫空间感)，它可以帮助我们判断图像中物体的远近关系。比如说，我们能够辨识出图像中

的物体，并且知道其大致的大小；比如，近处的物体会挡住远处的物体，而太阳，月亮等天体一般在很远的地方；再比如，物体受到光照后会留下影子，等等。这些信息都能够帮助我们判断物体的远近。

由于单目相机拍摄的图像只是三维空间的二维投影，因此如果想要恢复三维结构，就必须改变相机的视角。在单目 SLAM 中也是同样的原理，我们必须移动相机，才能估计它的运动，同时估计场景中物体的远近和大小 (不妨称之为结构)。那么，怎么估计这些东西的运动和结构呢? 从生活经验我们可以知道，如果相机向右移动，那么图像里的东西就会向左移动-> 这就给我们的推测带来一些信息。另一方面，我们还知道，近处的物体移动的快，远处的物体移动的慢。于是，当相机移动时，这些物体在图像上形成了视差，通过视差，我们就可以判断哪些物体离得远，那些物体离得近。

然而，即使我们知道了物体的远近，他依然只是一个相对的值。比如我们在看电影时，虽然知道电影场景中某些建筑相对于另一些建筑更大一些，但是无法确定电影中那些建筑的 “真实尺度”；那些建筑是真实存在的高楼大厦，还是桌上的特制模型？直观的说，如果把相机的运动和场景的大小同时方法数倍，单目相机所看到的像是一样的。同样的，把这个大小乘以任意倍数，我们都将看到一样的场景。这说明，单目 SLAM 估计的轨迹和地图与真实的轨迹和地图相差一个因子，也就是所谓的尺度 (Sclae)，由于单目相机无法仅凭图像确定这个真实尺度，因此又称为尺度不确定性。

平移之后才能计算深度，以及无法确定真实尺度，这两件事情给单目 SLAM 的应用造成了很大的麻烦，其根本原因是无法通过单张图像确定深度，所以，为了得到这个深度，人们开始使用双目相机和深度相机。

2.1.2 双目相机和深度相机

正如上文所说，使用双目相机和深度相机的目的是，我们期望以某些手段来获取单目相机所无法得到的深度信息。而一旦知道了距离，场景的三维结构就可以通过单个图像恢复出来，也就消除了尺度的不确定性。尽管都是为了测量距离，但是双目相机和深度相机测量深度信息的原理是不同的。

双目相机由两个单目相机组成，这两个相机之间的距离称之为基线 (Baseline)，通常基线是已知的，我们通过这个基线来估计每个像素点的空间位置。这和人眼非常相似，人类可以使用左眼和右眼图像的差异来判断物体的远近，计算机使用双目相机也能做到同样的效果。计算机上的双目相机需要大量的计算能力才能估计每一个像素点的深度，相比于人类来说非常笨拙。双目相机测量到的深度范围与基线有关，基线越大，则测量到的深度越远，所以通常无人车上搭载的双目相机会是一个大家伙。

双目相机的距离估计是比较左右眼的图像获得的，并不依赖其他传感设备，所以它既可用于室内，也可用于室外。双目相机的缺点是配置与标定均非常的复杂，其深度量程和精度受双目的基线和分辨率所限制，而且视差的计算非常的消耗资源，再用 GPU 和 FPGA 加速后，才能实时输出整张图像的距离信息，因此在现阶段的应用中，计算量是双目相机的主要问题之一。

深度相机 (又称为 RGB-D 相机，在后文中多用 RGB-D 代指深度相机) 是 2010 年左右开始兴起的一种相机，它最大的特点是可以用过红外结构光，或者 Time of Flight(TOF) 原理，像激光传感器一样，主动地向外发射红外光或不可见光，并接受返回的光，测量出物体与相机之间的距离。这部分并

不像双目相机一样通过软件计算来解决，而是通过物理的测量手段，所以相比于双目相机，RGB-D 可以节省更多的时间和资源。目前常用的 RGB-D 相机有:Kinect/Kinect V2 , Xtion PRo Live , RealSense 等，不过目前 RGB-D 还存在有测量范围窄，噪声大，视野小，易受日光干扰，无法测量透射材质等多个问题，在 SLAM 方面 RGB-D 主要应用于室内，室外则比较难用。

2.2 经典视觉 SLAM 框架

下面我们来介绍经典的视觉 SLAM 框架，如下图所示，来了解一下视觉 SLAM 由那些模块组成

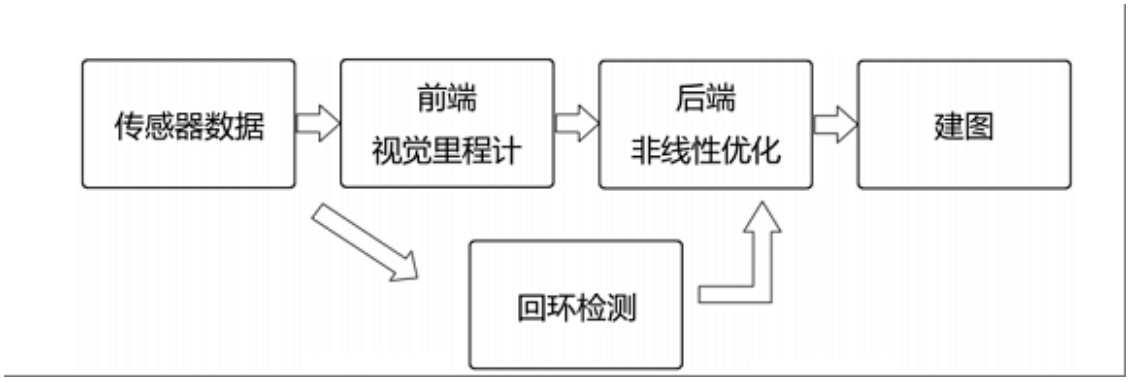


Fig. 2. 整体视觉 SLAM 流程图 (从 SLAM 十四讲书中直接摘的嘻嘻)

整个视觉 SLAM 流程包括以下几个部分

1. 传感器信息读取: 在视觉 SLAM 中，主要为相机图像信息的读取和预处理。如果是在机器人中，还会有码盘，IMU 等信息的读取和同步。

2. 视觉里程计 (Visual Odometry -> VO): 视觉里程计的任务是估算相邻图片之间的运动，以及局部地图的样子，VO 也被称为前端。

3. 后端优化 (Optimization): 后端接收不同时刻 VO 测量的相机位姿，以及回环检测的结果，对他们进行优化，得到全局一致的轨迹和地图。由于优化接在 VO 的后面，因此又被称为后端。

4. 回环检测 (Loop Closing): 回环检测判断机器人是否达到过先前的位置，如果检测到回环，则会把信息提供给后端进行处理

5. 建图 (Mapping): 建图是根据估计的轨迹，建立与任务要求对应的地图。

基本的视觉 SLAM 框架，已经是过去十几年的研究成果了，这个框架本身及其所包含的算法已经基本定型，并且已经在许多视觉程序库和机器人程序库中提供。依靠这些算法，我们能够构建一个视觉 SLAM 系统，使之在正常的工作环境里实时定位与建图。因此我们说，如果把环境限定在静态，刚体，光照变化稳定的，没有人为干扰的场景中，那么 SLAM 系统是一个非常成熟的系统。

接下来我们开始详细的介绍每个模块

2.2.1 **视觉里程计** Visual Odometry

视觉里程计关心的是相邻图像之间的相机运动，最简单地情况就是两张图像之间的运动关系。例如，当看到下图的时候，我们会下意识的反映出，右图是左图向左旋转一定角度的结果 (在视频流的情况下会更加自然)。

Fig. 3. 相机拍到的图片与人眼反应的运动方向 (从 SLAM 十四讲书中直接摘的嘻嘻)

我们不妨思考一下：自己是怎么知道“向左旋转”这件事情的呢？人类早已习惯于用眼睛去探索世界，估计自身位置，但又往往难以用理性的语言去描述我们的直接。在图 Fig.3 中，我们会自然的认为离我们最近的是吧台，尔远处是墙壁和黑板。当相机向左旋转时，吧台离我们近得部分出现在视野中，而右侧远处的柜子则淡出了视野。通过这些信息，我们判断相机应该是向左旋转了。

但更进一步来讲，我们能否确定旋转了多少度？平移了多少厘米？我们就无法给出一个确定的答案了，因为我们的直觉对于具体的数字并不敏感。但是，在计算机中，又必须精确的测量这段运动的数据。所以我们要思考：计算机是如何通过图像来确定相机的运动的呢？

在前面我们提到过，在计算机视觉领域，人们在直观上看起来十分自然的事情，在计算机视觉中却非常困难。图像在计算机中只是一个数值矩阵，至于这个矩阵表达了什么，计算机对此一无所知 (当然这正是目前 ML 所在解决的问题)。而在视觉 SLAM 中，我们只能看到一个个像素，知道他们是空间中某些点在相机的成像平面上投影的结果，所以，为了定量的估计相机的运动，必须先要了解相机与空间点的几何关系。

考虑到要讲清楚这个几何关系以及 VO 的实现方法，需要先铺垫有关坐标变换的一系列基础知识，因此我们将这一点放在后面来进行讲解。

现在我们只需要知道，VO 能够通过相邻的图像来估计相机的运动，并恢复场景的空间结构。称它为“里程计”是因为它和实际的里程计一样，只计算相邻时刻的运动，而和再往前的过去的信息没有关联。在这一点上，VO 就像是只有 7 秒记忆的鱼 。

假定我们已经拥有了一个视觉里程计，估计两张图像之间的相机运动。那么，是不是只要我们把相邻时刻的运动“串”起来，就够成了机器人的运动轨迹，从而解决了定位问题。另一方面，我们根据每个时刻相机的位置，计算出各个像素对应的空间点位置，就得到了地图。那么是否，我们拥有了

VO，就可以解决 SLAM 问题了呢?

答案是否定的，视觉里程计确实是视觉 SLAM 的关键，但是只通过视觉里程计来估计轨迹的话，将会不可避免的出现累积漂移 (Accumulating Drift) 的问题。这是由于视觉里程计只估计两个图像之间的运动造成的。一个显然的情况是，在每次估计时，都会带有一定的误差，由于里程计的工作方式，先前时刻的误差会传递到下一时刻，导致经过一段时间的估计后，视觉里程计估计出的轨迹将会不再准确，如下图所示。

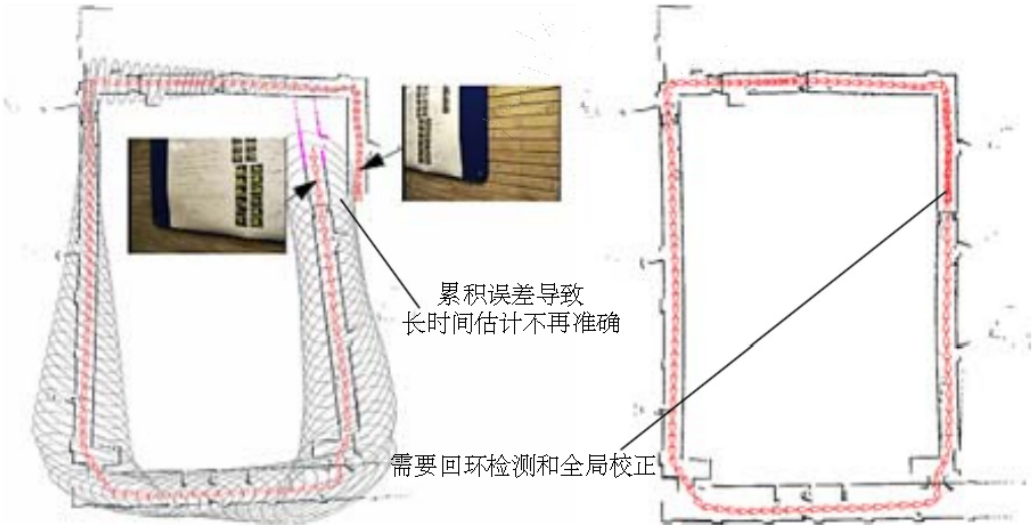


Fig. 4. 累积误差与回环检测 (从 SLAM 十四讲书中直接摘的嘻嘻)

假如机器人先向左转 90 度，再向右转 90 度，由于估计时的误差，我们将第一个 90 度估计成了 89 度，那么在后面的估计中都会有这-1 度的误差。这也就是所谓的漂移 (Drift)，他将导致我们无法建立起一致的地图。为了解决这样的问题，我们引入了两种技术：后端优化和回环检测。回环检测负责把“机器人回到原始位置”这件事情检测出来，而后端优化通过回环检测的信息，对 VO 估计的轨迹进行优化。

2.2.2 **后端优化** Optimization

简单地来说，后端优化主要处理是 SLAM 过程中的噪声问题。虽然我们都希望得到的数据是准确无误的，但是再昂贵的传感器所采集的数据也会有噪声和误差。所以在解决了“如何从图像中估计出运动”后，我们还要关心这个估计带有多大的误差，这些误差是如何从上一个时刻传递到下一个时刻的，而我们对当前的估计又有多大的信心。后端优化所要考虑的问题，就是如何从这些带有噪声的数据中估计整个习题的状态，以及这个状态估计的不确定性有多大–这也被称为最大后验概率估计 (Maximum-a-Posteriori -> MAP)。这里的状态既包括有机器人自身的轨迹，也包含地图。

相对的，视觉里程计有时也被称为“前端”，在 SLAM 框架中，前端给后端提供待优化的数据，以及这些数据的初始值。后端则负责整体的优化过程，它面对的往往只有数据，而不关心数据从何而

来。简单地来说，在视觉 SLAM 中，前端和计算机视觉研究领域更加相关，比如图像的特征提取与匹配，后端则主要是滤波与非线性优化算法。

2.2.3 **回环检测** Loop Closing

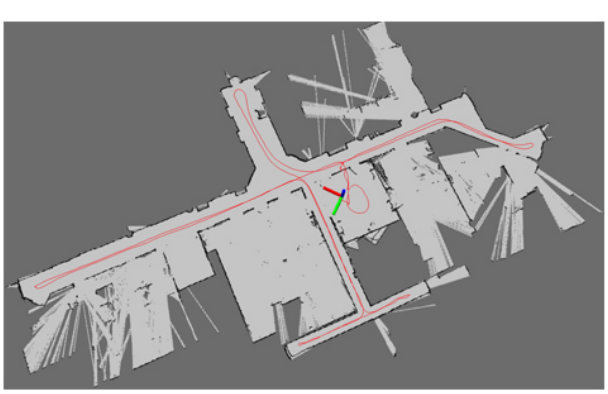
回环检测，又称之为闭环检测 (Loop Closing Detection)，主要解决位置估计随着时间漂移的问题。假设机器人在一段时间的运动后回到了原点，但是它的位置估计值 (比如坐标 x，y) 并没有回到原点，怎么办呢？我们希望能够有一种手段，让机器人知道“回到原点”这件事情，或者说把”原点”识别出来，我们再把位置估计值更新过去，就可以消除掉漂移了。这就是所谓的回环检测。

回环检测与”定位”和“建图”息息相关，事实上我们认为，地图存在的意义主要是让机器人知道自己所到达过得地方。为了实现回环检测，我们需要机器人拥有能够识别到达过场景的能力。比如，在某些地方设置一个二维码图片，只要机器人看到这个图片就知道自己到达了什么地方。但是，这样的办法等价于我们先前所讨论的，基于环境中的传感器估计机器人自身的位置，因此会对应用环境作出一定的限制。我们更希望机器人能够通过自身所携带的传感器 (也就是摄像头所采集到的图像本身) 来完成这件事，例如判断两张图片之间的相似性来完成回环检测，这一点和人是相似的。当我们看到两张相似的图片时，容易辨认它们来自同一个地方。

所以，视觉回环检测本质上是一种计算图像数据相似性的算法，由于图像所蕴含的信息丰富，使得回环检测的难度降低了一些。在检测到回环之后，回环检测会将类似 “A 和 B 是同一个点” 这样的信息给到后端优化，然后后端根据这些新的信息，把轨迹和生成的地图调整到符合回环检测结果的样子。如果我们有着丰富的回环检测数据，就能够消除累积误差，得到全局一致性的轨迹和地图。

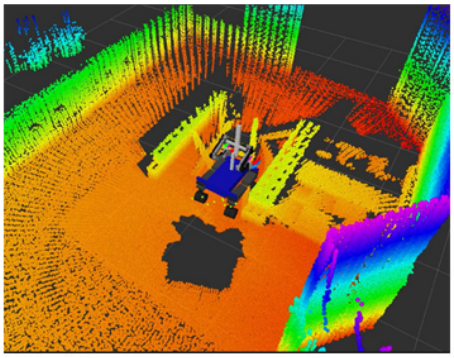
2.2.4 **建图** Mapping

建图，指的是构建地图的过程。地图是对环境的描述，但这个描述并不是固定的，根据应用领域的不同，地图也分为不同的种类。

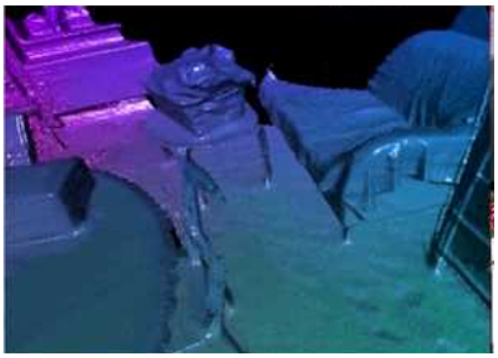


2D 栅格地图

2D 拓扑地图



3D 点云地图



3D 网格地图

Fig. 5. 形形色色的地图 (从 SLAM 十四讲书中直接摘的嘻嘻)

对于家用扫地机器人来说，这种主要存在于低矮平面的机器人，只需要一个二维地图，标记哪里可以通过，哪里存在障碍物，就可以满足它在一定的范围内导航了。而对于一个相机，它有着六自由度的运动，因此至少需要一张三维的地图。有的时候，我们想要一个漂亮的重建结果，不仅是一组空间点，还需要带有纹理的三脚面片。另一些时候，我们不关心地图的样子，只需要知道 A 点和 B 点之间能否通过这样的事情。

对于地图，我们有着太多的想法和需求。因此，相比于前面提到的视觉里程计，后端优化，回环检测，建图并没有一个固定的形式和算法。一组空间点的集合可以称为地图，一个漂亮的 3D 模型也可以是地图，一个标记着城市，村庄，铁路的图片也是地图。

地图的形式根据 SLAM 的应用场合而决定，大体上可以分为度量地图和拓扑地图。

- 度量地图

  度量地图强调精确的表示地图中物体的位置关系，通常用稀疏 (Sparse) 与稠密 (Dense) 对其进行分类。稀疏地图进行了一定程度的抽象，并不需要表达所有的物体。例如，我们选择一部分具有代表意义的东西，称之为路标 (Landmark), 那么一张稀疏地图就是由路标所组成的地图，

而不是路标的部分就可以忽略掉；相对的，稠密地图着重于建模所有看到的东西。对于定位来说，稀疏路标地图就足够了。而用于导航时，则需要稠密地图。稠密地图往往按照某种分辨率，由许多小块组成。对于二维地图是许多小格子 (Grid)，对于三维地图是许多小方块儿 (Voxel)(因此也被称为栅格地图)。一般的，一个小块儿含有占据，空闲，未知三种状态，以表达该格内是否有物体。当查询某个空间位置时，地图能够给出该位置是否可以通过的信息。这样的地图可以用于各种导航算法，如 A*,D* 等，为研究人员所重视。

- 拓扑地图

  相对于度量地图的精确性,拓扑地图更强调地图元素之间的关系。拓扑地图是一个图 (Graph)，由节点和边所组成，之考虑节点之间的连通性。它放松了地图对精确位置的要求，去掉了地图的细节问题，是一种更为紧凑的表达方式。然而拓扑地图不善于表达具有复杂结构的环境，如何对地图分割形成边和点，又如何使用拓扑地图来进行导航与路径规划，还是待研究的问题。

2.3 SLAM **问题的数学表达**

通过前面的表述，我们对 SLAM 中的各个模块的组成和主要的功能有了一个直观的了解和感性的认识，但仅仅是这样并不能让我们写出可以运行的程序。我们要将这份认识上升到理性的层次，也就是用数学语言来表述 SLAM 过程。

假设我们之前提到的小萝卜机器人携带着某种传感器 (假设为相机) 在未知的环境里运动，怎么用数学语言来表述这个过程呢？首先，由于相机通常实在某些时刻采集数据的，我们也只关心这些时刻的位置和地图。因此我们就将连续的运动抽样成了离散时刻 $t = 1,2,\cdots,K$ 当中发生的事情。在这些时刻中，用 x 来表示小萝卜机器人自身的位置。于是各个抽样的离散时刻中小萝卜机器人的位置就可以记为 $x_1,x_2,\cdots,x_K$, 这些位置，就够成了小萝卜机器人的运动轨迹。在地图方面，我们假设地图是由许多个路标 (Landmark) 所组成的，在每个时刻，传感器会观测到一部分路标点，得到他们的观测数据。不妨假设一共有 N 个路标点，用 $\mathrm{y}_1,\mathrm{y}_2,\cdots,\mathrm{y}_N$ 来表示。

在这样的设定中，“小萝卜机器人携带着传感器在位置的环境中运动”，由如下两件事来表述:

1. 什么是运动？我们要考虑从 $k-1$ 时刻到 k 时刻，小萝卜机器人的位置 x 是如何变化的
2. 什么是观测？假设小萝卜机器人在 k 时刻于 x_k 处探测到了一个路标 y_j, 我们要考虑这件事情如何用数学语言来描述。

先来看运动，通常，机器人会携带一个测量自身运动的传感器，比如说码盘或者是惯性传感器。这个传感器会测量有关运动的参数，不一定就是位置差值，也可能是加速度或者角加速度信息。不过，无论是什么传感器，我们都能使用一个通用的，抽象的数学模型:

$$x_k = f(x_{k-1}, u_k, w_k) \tag{1}$$

这里，u_k 是运动传感器的读数 (当然通常我们称之为输入)，w_k 称之为噪声。注意到，我们用一个一般的函数 f 来描述这个过程，而未指明 f 的作用方式。着使得这个数学模型可以作用于任意的运动传感器，成为一个通用的方程，而不必局限于某个特定的传感器，我们将之称为运动方程。

与运动方程相对应的，是观测方程。观测方程描述的是，当机器人在 x_k 位置上看到某个路标点 y_j 时，产生一个观测数据 $z_{k,j}$。同样地，可以用一个抽象的函数 h 来描述这个过程:

$$z_{k,j} = h(y_j, x_k, v_{k,j}) \tag{2}$$

$v_{k,j}$ 是这次观测里的噪声，由于观测所用的传感器形式更多，因此这里的观测数据 z 以及观测方程 h 也会有许多不同的形式。

或许你会问，我们所用的函数 f,h 似乎并没有具体的说明运动和观测是怎么回事? 同时，这里的 x,y,z 又是什么东西呢? 事实上，根据小萝卜的真实运动和传感器的种类，存在着若干的参数化的方式 (Parameterization)。什么是参数化呢? 举例来说，假设小萝卜在平面中进行运动，那么它的位姿由两个位置和一个转角来描述，即 $x_k = [x, y, \theta]_k^T$，同时，运动传感器能够测量到小萝卜在任意两个时间间隔位置和转角的变化量 $u_k = [\Delta x, \Delta y, \Delta\theta]_k^T$，于是，此时运动方程就可以具体化为:

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_k = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{k-1} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix}_k + w_k. \tag{3}$$

这是简单地线性关系，不过并非所有的传感器都能直接测量出位移和角度，因此通常运动方程和观测方程会更加的复杂。针对不同的传感器，这两个方程有不同的参数化形式，如果我们保持通用性，将它们设集成通用的抽象形式，则 SLAM 过程可以总结为两个基本方程:

$$\begin{cases} x_k & = f(x_{k-1}, u_k, w_k) \\ z_{k,j} & = h(y_j, x_k, v_{k,j}) \end{cases}$$

这两个方程描述了最基本的 SLAM 问题: 当知道运动测量的读数 u，以及传感器的读数 z 时，如何求解定位问题 (估计 x) 和建图问题 (估计 y)? 此时，SLAM 问题建模为了一个状态估计问题: 如何通过带有噪声的测量数据，估计内部的，隐藏着的状态变量?

状态估计问题的求解，与两个方程的具体形式，以及噪声服从哪种分布有关。按照运动和观测方程是否线性，噪声是否服从高斯分布进行分类，分为线性/非线性和高斯/非高斯系统。其中线性高斯系统（Linear Gaussian，LG 系统）是最简单的，它的无偏最优估计可以由卡尔曼滤波器（Kalman Filter，KF）给出。而在复杂的非线性非高斯系统（Non-Linear Non-Gaussian，NLNG 系统）中，我们会使用以扩展卡尔曼滤波器（Extended Kalman Filter，EKF）和非线性优化两大类方法去求解。直至 21 世纪早期，以 EKF 为主的滤波方法在 SLAM 中占据了主导地位。我们会先在工作点处把系统线性化，并以预测—更新两大步骤进行求解。最早的实时视觉 SLAM 系统即是基于 EKF 开发的。随后，

为了克服 EKF 的缺点（例如线性化误差和噪声高斯分布假设），人们开始使用粒子滤波器（Particle Filter）等其他滤波器。时至今日，主流视觉 SLAM 使用以图优化（Graph Optimization）为代表的优化技术进行状态估计。目前，我们认为优化技术已经明显优于滤波器技术，只要计算资源允许，通常都偏向于使用优化方法。