第1章

三维重建

1.1 概述

在计算机图形学、计算机视觉领域中，人们习惯于用图像来表示一个对象或是一个场景，然而在结构表示中，三维模型相比二维图片，对于局部细节与结构的描述更为精细。完整的三维模型，可以避免二维图片中对象的自遮挡、结构缺失等问题，进而提高其他视觉任务（如对象识别、分割）的精度。可见，相比于图像，三维模型拥有更强大的表达能力，在机器人导航与交互、数据可视化等领域中拥有图像所不可替代的作用。

三维重建正是为场景或对象建立一个这样的三维模型。其中最常见的方法就是使用相对容易获取的图片，通过将不同视角下拍摄的图片整合到全局的三维坐标系中，从而重建其三维形状。本章主要讨论的就是如何从图像中恢复三维的结构，包括深度信息的估计、单目标的三维重建以及场景的三维重建，并针对这些问题介绍基本原理，展示相关的实践结果。

传统的图像三维重建方法通常使用大量已知拍摄位置的图像（或是通过算法估算拍摄位置），来生成场景的三维几何表示，这种几何表示可以是视差图（或深度图）、以体素构成的三维体、距离场或三维点云等。三维重建的基本原则很简单，需要满足输出的三维点在投影回二维图像时，与对应视角的输入图像保持一致。这其中体现了三维重建的两个至关重要的因素，即图像观察点的位置以及观察点与重建模型之间的对应关系，或者进一步称为定位和建图。在没有相机位置标定的拍摄场景中，需要在拍摄过程中对相机定位，以判断像素点与三维模型的转换关系，最终在全局坐标系下构建三维地图。

基于上述基本原则，三维重建可以利用视觉信息和几何特征的对应关系进行匹配，从而完成位姿的优化和三维信息的恢复，本章中称其为基于多视图几何的三维重建；也可以使用最近流行的基于深度学习的三维重建方法，将三维重建问题变成一种数据驱动的网络训练模式，并可在端到端的网络中加入一些投影几何关系，提高重建网络对几何关系的约束。

然而，三维模型相比于二维图片，要占用更多的存储空间，耗费更大的计算量，

再加上三维模型本身的结构复杂性，导致三维结构的重建与分析更具有挑战性，上述的两种方法在某些方面具有局限性。本章会从基于传统多视图几何的三维重建和基于深度学习的三维重建这两个方向讲解每种方法的基本原理、国内外研究现状，分析其优缺点和适用情景，并介绍一些与三维重建相关的技术案例（包含深度信息的估计、单目标的三维建模以及场景的三维重建），最后对本章内容进行总结。


第 1 章资源

本章中所涉及的基础算法与技术案例可以参考左侧二维码中所给出的链接，这些资源都是在 Github 上公开的资源，读者可以通过研读相关的论文和代码来加深理解。

1.2 国内外主要研究工作

1.2.1 基于多视图几何的三维重建

基于多视图几何的三维重建是一种比较传统的方法，然而传统并不代表其不会再有新意，时至今日，每年的顶级会议和期刊上依旧有不少相关工作和问题出现。这类方法通常使用大量已知拍摄位置的图像（或是通过算法估算拍摄位置），来生成场景的三维几何表示，这种几何表示可以是视差图（或深度图）、以体素构成的三维体、距离场或三维点云等。其基本原则是需要满足输出的三维点在投影回二维图像时，与对应视角的输入图像保持一致，也就是根据图像与三维全局结构之间的投影关联，从一个图像集合中恢复出其三维的几何结构。比如实时地从一个单目相机视频中重建表面的立体视觉方法[1-2]，这种方法假定表面都是理想散射，场景中点的深度由能量函数最小化来确定，因此重建结果鲁棒性较差。随着消费级深度相机的面世[3]，融合 RGB-D 信息（RGB 表示彩色图，D 表示深度图）的三维重建法越来越受欢迎，相比于单目重建，深度信息可以帮助确定图像点在相机坐标系中的 z 坐标，在一些环境下增强了重建鲁棒性。比如使用 RGB-D 图片的 RGB-D SLAM[4-5]，它通过提取图片 SURF（加速鲁棒特征）特征点，并匹配帧间的特征点用于估算 RGB-D 帧的相机位姿，最后使用图优化方法优化位姿轨迹，实现一个全局上最优的重建结果。

KinectFusion 算法[6-7]则仅使用深度信息，迭代更新截断符号距离函数（truncated signed distance function，TSDF）最终通过 3D 渲染反馈给用户。KinectFusion 算法相对于前述方法，大幅降低了场景创建过程的计算量，基本满足用户交互的实时性，但由于显存限制，只能在一个小范围的场景进行重建。基于这一局限，有团队使用空间分层[8]和哈希体素[9]来进行改善，或是利用点云[10-11]这一更一般化的数据形式来提高分辨率。

近几年比较有代表性的三维重建工作还有 DynamicFusion 算法[12]，该算法在 KinectFusion 算法的基础上，解决了如何在被测物体运动的情况下进行实时

的表面重建；BundleFusion算法[13]是目前该领域效果最好的方法，不需要任何显式的回环检测，支持在图形处理单元(graphics processing unit，GPU)上的实时鲁棒跟踪；来自我国国防科技大学的ROSEFusion算法[14]，适用于快速移动相机拍摄的场景重建。

本章接下来只简述比较经典的视觉即时定位与地图构建(simultaneous localization and mapping，SLAM)，和KinectFusion算法的原理，相信读者对这些有了了解后，对其他更先进的算法也能通过阅读相关文献而理解和掌握。

1. 视觉SLAM算法


视觉SLAM算法

SLAM指在没有环境的先验知识的条件下，驱动机器或人手持传感器，在环境中运动并感知信息，并在对自身运动进行定位估计的同时，建立这一未知环境的地图模型。接下来从SLAM的数学表述与经典视觉SLAM框架来介绍基本的视觉SLAM算法流程。

在定位方面，如果将传感器的轨迹按照离散的时间步 $t=1,\cdots,K$ 分离成对应的位置序列，用 x 表示机器人或其他代理的位置，那么这个位置序列可表示为 $x_1,\cdots,x_K$；在建图方面，如果假设地图是由 N 个路标组成(用 $y_1,\cdots,y_N$ 来表示)，路标在平面地图中可以看作一些具有标志性的建筑点，在三维场景中可以看作一些三维点。

在传感器沿着轨迹移动时，在每个时刻对应的位置上会测量到一部分的路标点，并得到其观测数据。SLAM问题就变成了研究传感器在轨迹上位置的变化和对路标的测量问题，前者称为运动，后者称为观测。SLAM模型正是将运动和观测两部分用数学形式表述并估计其中状态的算法。

假设在 $t=k$ 时，运动传感器在位置 x_k 上观测得到的数值为 u_k，传感器噪声为 w_k，则定义运动方程：

$$x_k=f(x_{k-1},u_k,w_k) \tag{1.2.1}$$

运动方程表示了当前时刻与上一时刻相比位置的变化。假设在 x_k 位置上观测到某个路标点 y_j，得到观测数据 $z_{k,j}$，则定义观测方程：

$$z_{k,j}=f(y_j,x_k,v_{k,j}) \tag{1.2.2}$$

其中，$v_{k,j}$ 为观测的噪声。运动方程和观测方程以数学形式描述了SLAM的过程，而定位与建图正是求解这两个方程中的位置 x 和路标点 y，那么SLAM问题的本质变为求解两个方程的状态估计问题，即通过有噪声的观测数据来估计方程内部的位置与路标点变量。

关于这一状态估计问题的求解有多种方法，包括线性高斯系统、卡尔曼滤波器、粒子滤波器以及在视觉SLAM中常用的图优化方法等。由于本章的主要目的是应用三维重建方法，因此接下来主要阐述使用视觉信息的视觉SLAM技术。

视觉 SLAM 主要用视觉传感器(比如单目相机、双目相机、深度相机等)采集数据并对自身定位,同时构建出场景的地图。由于其采集的数据大多是图像的形式,因此可以利用一些计算机视觉的方法辅助定位方程与观测方程的状态估计。

经典的视觉 SLAM 中可以拆分成 5 个模块,分别是信息读取、视觉里程计、回环检测、后端优化和建图。5 个模块的关系如图 1.2.1 所示。

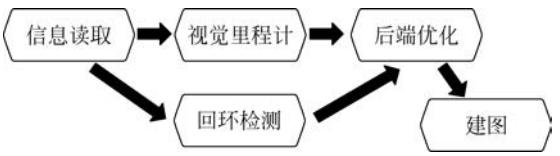

图 1.2.1 SLAM 5 个模块的关系

(1) 信息读取:这一模块主要对传感器拍摄得到的图像信息做读取和相应的处理操作,其作用在于为建立一个视觉里程计提供必要的视觉信息。实际在机器人上操作时,可能不局限于视觉信息的读取,可能还包含惯性传感器信息的读取与同步等。

(2) 视觉里程计:这一模块的主要作用是针对读取到的视觉信息,估计相机的运动并根据相机的运动建立地图,虽然建立的地图存在误差,但会在后端进行优化。完成一个视觉里程计,可以使用特征点法或直接法。以特征点法举例,首先对图像的特征进行提取并对相邻帧间进行特征的匹配,然后基于匹配的特征使用对极几何法或是最近点迭代等方法求解相机的位姿变化,最后根据世界坐标系下的齐次坐标与像素坐标系之间的变换公式来将二维特征点转化为三维地图上的路标点。转化关系如式(1.2.3):

$$P_{uv} = \boldsymbol{KT}P_{w} \tag{1.2.3}$$

其中,$\boldsymbol{K}$ 为相机的内参矩阵,在相机出厂后固定不变;$\boldsymbol{T}$ 为相机的外参矩阵,表示了相机的旋转与平移,随着相机位置的变化而变化。

在得到相邻帧的位置变化后,累计计算整个序列中每个图片的位置并将二维特征点转化到三维路标点后,即可得到一个里程的地图。

(3) 回环检测:一个视觉里程计能够建立一个地图,然而由于噪声的存在,相邻帧之间的位姿估计存在误差,一个里程下来,误差不断累积,导致相机在回到之前去过的位置时,重建的地图与原先不同。因此,检测先前去过的位置,形成一个回环,这样就能在相邻帧间特征匹配基础上又增加回环上的约束,约束相机的定位与地图的构建应该满足全局上的一致。关于回环检测,一个常用的方法是对图像建立一个词袋模型(BoW),这一模型描述了图像上有哪几种特征,比如图像上有桌子、椅子和书,那么这些物体就构成了 BoW 中的“单词”,许多个单词组成在一起通过聚类就形成了一个字典,如果某个单词存在于图像中则用 1 来表示,不存在则用 0 来表示,那么如果字典中有 N 个单词,一个图像就可通过

一个 N 维的由 0 和 1 组成的向量来表示。计算向量之间的距离就可以判断两幅图像之间的相似性，当相似性足够高时就可以认为检测到了回环从而为后端优化提供更多的约束。

(4) 后端优化：后端优化接收视觉里程计计算得到的相机位置与回环检测提供的约束，通过优化最终得到在全局上一致的轨迹和地图。后端优化有多种方法，包括线性高斯系统、卡尔曼滤波器、粒子滤波器以及图优化等方法。以现在比较常用的图优化方法为例，首先将世界坐标系中的点根据式(1.2.3)投影到像素坐标下，得到视觉 SLAM 的观测方程，将其抽象为

$$z = h(x, y) \tag{1.2.4}$$

其中，位置 x 指代相机位姿，即旋转矩阵与平移向量，假设其对应的李代数为 ξ。当待重建的地图是三维点云时，路标 y 指代的是三维点 p，z 指代观测方程得到的值。那么观测的误差可定义为传感器观测值与观测方程预估值的差：

$$e = z - h(\xi, p) \tag{1.2.5}$$

假设一共有 m 个位姿，n 个路标点，那么整体的代价函数可按最小二乘法来定义：

$$L = \frac{1}{2}\sum_{i=1}^{m}\sum_{i=1}^{n} \| z_{ij} - h(\xi_{ij}, p_{ij}) \|^2 \tag{1.2.6}$$

其中，下标 ij 表示在第 i 个位置，第 j 个路标的数据。接下来需要对这一误差进行优化，通过最小二乘法的求解，也就相当于对方程中的隐藏状态(位姿和路标)进行调整，最终得到在全局上一致的位姿轨迹和地图。

(5) 建图：地图的表现形式有多种，包括二维栅格地图、拓扑地图、点云地图、网格地图、八叉树地图等。以三维点云地图为例，它用三维坐标的点集来表示，数据无序，精确地表示了地图中物体的具体空间位置，同时分为稀疏地图和稠密地图。稀疏地图只选择了一些具有代表性的部分来组成地图，相当于路标，其他则忽略；稠密地图则对所有该传感器能够看到的东西都进行了建模，而用于导航的地图通常按照某个分辨率将其转化为许多个小方块，也就是体素。体素表示了其中是否有三维结构占据。

经过这 5 个模块，系统能够通过视觉传感器获得图像信息，即时地进行定位与地图构建，然而实际应用中可能远远比这要复杂，所使用的信息可能也不仅局限于视觉信息，对此本章不做考虑。

2. KinectFusion 算法


KinectFusion 算法

KinectFusion[6-7]算法是一种使用深度传感器拍摄的深度信息来对相机轨迹进行跟踪并实时重建场景的三维建模算法。相比于视觉 SLAM 方法，KinectFusion 算法可以划分为 4 个阶段：第一阶段是深度图的转换，深度信息将二维像素转化到三维坐标上；第二阶段使用最近点迭代法计算相机位姿的变化；第三阶段相当于 SLAM 中的建图，不过在三维数据形式上，KinectFusion 算法使用

的也不再是点云而是距离场；第四阶段则是面向用户所做的实时三维渲染。接下来对这 4 个阶段做介绍。

(1) 深度图转换：深度相机在获得深度图后，通过校准的相机内参矩阵 $\boldsymbol{K}$ 可以将像素点转换为相机坐标系下的一个三维点，其转换公式为

$$V_{x,y}=D_{x,y}\boldsymbol{K}^{-1}[x,y,1]^{\mathrm{T}} \tag{1.2.7}$$

其中，$D_{x,y}$ 为深度图在图像坐标(x,y)处的深度值，$V_{x,y}$ 为对应的相机坐标系下的三维点。考虑临近的投影点，还可计算出三维点的向量：

$$n_{x,y}=(V_{x+1,y}-V_{x,y})\times(-V_{x,y}) \tag{1.2.8}$$

不同相机坐标系下的点需要统一转化到一个全局的世界坐标系下，这就需要计算图像对应相机的旋转和平移，也就是相机的外参矩阵 $\boldsymbol{T}$，关于外参矩阵的计算在下一阶段的相机跟踪中进行。

(2) 相机跟踪：KinectFusion 算法使用最近点迭代法进行相机位姿的实时计算。算法首先利用投影关联将当前相机坐标系下的顶点 V 按照上一时刻计算的相机位姿投影到它的相机坐标系下，并透视投影到图像坐标中，按照顶点和法线去查找图像中对应点的位置；其次经过剔除异常值、判断顶点和对应点在全局坐标系下的距离角度是否在一定阈值内，得到对应点的集合；最后，为了得到全局上最优的外参矩阵 $\boldsymbol{T}$，计算当前帧上每个点与先前帧上对应点切平面的距离，通过迭代最小化这个距离来求得最优的外参矩阵 $\boldsymbol{T}$。KinectFusion 算法通过假定帧间仅有一个增量变换来线性近似这个系统，并通过树规约在 GPU 上求解。整个过程中，KinectFusion 算法在 GPU 上求解，既没有提取特征，也没有对点稀疏采样，实时性较好。但是其前提是需要密集的相机跟踪，也就是需要密集的深度图序列来假设帧间仅有一个增量变换。

(3) 体表示：对于跟踪计算的相机位姿轨迹，KinectFusion 算法[6-7]用体表示(volumetric representation)来整合轨迹上的深度信息，将其统一到一个全局坐标系上。这一体表示实际上是一个距离场，表示其到实际表面的距离，在表面前时，值为正，在表面后则为负，在表面上则为 0。如果只对表面附近做考虑，则可定义一个截断区域，在截断区域下的距离函数，称为截断符号距离函数(truncated signed distance function，TSDF)。如果用 512^3 的体素来表示一个场景，每个体素中保存着 TSDF 值，KinectFusion 算法需要实时地对 TSDF 值进行更新，更新后的距离场可以进一步做三维渲染呈现给用户。

(4) 三维渲染：在得到更新的距离场后，KinectFusion 算法使用光线跟踪法进行渲染和跟踪三维体表示中的隐式表面，直观地呈现给用户。对于渲染图像中的像素，GPU 首先并行地沿着射线进行渲染，射线会穿过三维体中的体素；其次，在射线上 TSDF 值为 0 的两侧体素上，用三线采样点的线性插值法计算隐式表面交界点，并通过对零点交界处求导计算曲面法线；最后，使用计算得到的信息做像素的光照计算，从而完成渲染。

经过这4个阶段，系统能够通过深度相机获得的深度信息，在GPU下即时地跟踪相机的轨迹，并建立一个全局上的体表示，这个体表示相当于一个三维的建图，最后可以面向用户进行渲染。

这些基于多视图几何的三维重建算法经过多年的研究，已日趋成熟，然而这些算法最开始一般都是提取特征点和匹配帧间的特征点，并进行位姿的优化，这就不可避免地面临以下几个问题：

第一，算法需要传感器感知的信息稳定而不受到环境的影响。无论是使用彩色相机还是深度相机，采集的数据都会受到环境的干扰。对彩色图来说，由于需要对其进行特征的提取，因此要求图像有足够的纹理，而场景中像墙面等照片的纹理并不丰富，导致特征的匹配变得困难；同时对强反射物体、透明物体的彩色图和深度图质量均会有影响。此外，移动过程中的不稳定与扰动都可能促使扫描过程中丢帧或是使拍摄的图片变模糊，这些都会影响匹配的准确性，从而影响定位与建图。

第二，算法需要稠密的视角。SLAM算法对临近帧的匹配与KinectFusion算法中的最近点迭代法都要求拍摄视角密集，这样才能够有足够的重叠区域，进行匹配与位姿估算。这一点与人类智能不同，人类往往能通过几个视角的观察就确定其三维的结构，使用如此稠密的视角扫描一个对象在这个人工智能的时代着实有些“浪费”。特别是在一些场景中，由于遮挡或是不便到达等原因，对象的一些视角无法获得，这个时候对其重建可能会丢失关键结构。

第三，算法需要人为地规划路径。这两种方法都是对场景或对象进行稠密视角的扫描，需要获得涵盖场景或对象的稠密图像信息，并没有算法上的扫描轨迹规划，这就需要人手持相机进行扫描，来确保覆盖足够多的信息。

尽管这些方法拥有诸多局限，但由于其算法相对成熟，实时性较好，在一般环境下鲁棒性和精度都满足需求，因此它们依然是目前最流行的三维重建算法，并将随着硬件和算法的更新，拥有更强的适用性。

1.2.2 基于深度学习的三维重建


单视角
重建算法


多视角
重建算法

如前所述，基于多视图几何的三维重建方法存在多方面的局限性，包括算法需要稠密的视角，需要假定场景中对象没有镜面反射，需要拥有丰富的纹理，以及需要优化相机配准误差等。这些局限性导致在光照条件不好、拍摄视角存在遮挡、噪声较强的环境下，重建的结果可能存在关键结构缺失的问题。相比之下，深度学习在大量数据的支持下，能够通过先验知识对对象的全局结构进行预测，以及对局部细节进行补全。深度学习的引入为三维建模的研究带来新思路，为稀疏信息的三维建模提供新可能，因此是学术界和工业界的研究热门。

如果将三维重建问题看作一个基于图片特征的预测和生成问题，则可以使用卷积神经网络搭建二维编码网络和三维解码网络，从而将二维像素编码成特征向

量并解码成三维体素。比如，Wu 等[15]使用自编码的三维对抗生成网络，Rohit 等[16]使用的 T-L 网络，能够将输入的单张图片生成对应的三维模型；Dai 等[17]则将自编码网络结构与分类网络相结合，并通过图形合成技术的后处理，输入单一视角的图片，输出预测和合成的三维模型；普林斯顿团队使用扩张卷积网络通过单一视角下的深度图，实现场景的补全和语义分割[18]。

除了用三维体素来表示一个对象的重建形状，还可以通过生成八叉树体素、点云、三维网格和隐式来重建对象。比如，八叉树生成网络[19]使用更高效的八叉树来减少显存占用和计算量，生成更高分辨率的三维体素；点云生成网络[20]则第一个将无序点云这一更方便的数据形式引入深度学习中，通过单张彩色图预测出其三维点的分布；AtalsNet 算法则在生成点云的基础上，判断点所在的面片，重建出具有表面的三维网格[21]；DeepSDF 算法和 OccNet 算法使用隐式来表示三维形状，在连续的三维空间上输出距离函数或是决策边界，并通过采样和等值面提取来生成三维网格[22-23]。

上述方法有一个共同点，是通过输入单张彩色图片或深度图片，基于大型的三维模型数据库（比如 ShapeNet[24]，SUN3D[25]，ScanNet[26]等）训练神经网络，并通过“二维编码-三维解码”的思想预测三维信息。然而，即便训练出强大的深度学习模型，也无法弥补视角的缺失问题，特别是对于结构复杂的三维模型，只获得其中一个视角，难以表达全部的几何结构信息，使得最终重建的结果可能存在较大的误差。基于这一问题，Choy 等[27]提出了三维的循环神经网络，通过搭建三维的循环神经网络，接收多个视角下拍摄的图片，同时利用网络隐藏层，隐性表示当前重建的几何结构，从而建立了单视角与多视角联合的深度学习三维重建框架。Kar 等[28]又在这一网络模型的基础上，将不同视角下的语义特征投影到对应的三维空间中，来实现图片几何特征的提取，进一步提高体素重建质量；大连理工大学团队将多视角的三维重建和视角规划相结合，利用三维注意力机制动态选择视角，并通过深度学习恢复三维信息[29]。

还有很多使用到深度学习技术的三维重建算法，比如通过匹配数据库中的部件并进行组装实现高质量重建的模板匹配方法[30-31]，通过基元拟合实现带有基元表示的三维重建[32-33]等。笔者恐怕无法对所有相关方法进行细致介绍，只是在接下来从基于深度学习方法的三维重建中最基本的算法出发，阐述其流程和优缺点，帮助读者对其有基本的了解，相信通过这些了解，读者能够更轻松地阅读最新的相关文献。

1. 深度学习方法简述

深度学习是近 10 年来非常热门的研究领域。2013 年 4 月，《麻省理工学院技术评论》（*MIT Technology Review*）杂志将深度学习列为 2013 年十大突破性技术之首，在 AlphaGo 战胜围棋世界冠军后，深度学习再次掀起狂潮，并应用于医疗、娱乐、交通等多个方面。深度学习仿造人脑的工作过程，接收信号并处理，并不断抽象这一信号，最后完成任务。人的感知过程是由浅入深的，深度学习也是利用这

一特点，由浅入深地将底层特征抽象为高层特征（比如一个物体的类别），在形式上像一个神经网络，通过神经元处理输入的信号，随着网络的加深不断抽象特征进行更深层的认知。

而在计算机视觉中，卷积神经网络和循环神经网络是两个最受欢迎的深度学习模型。前者用于识别位移、缩放及其他形式扭曲不变性的二维像素或三维体素，后者处理当前输出与先前输出有关的时间序列问题。

（1）卷积神经网络：一般的卷积神经网络主要由输入层、卷积层、激活函数、池化层和全连接层组成。卷积层的计算有两个关键的操作：建立滤波器和窗口滑动。

如图 1.2.2 所示，以图像的二维卷积为例，输入层处理输入的图像，并送入卷积层中。卷积层的作用在于对输入进行更深层的特征提取。在卷积计算中，首先要设定卷积核，也就是滤波器。二维卷积的滤波器大小为 fH × fW × inCh × outCh，其中 fH 和 fW 分别表示卷积核的高和宽，inCh 和 outCh 表示输入特征的深度（通道数）和输出特征的深度。在输入和输出的每个通道上，相当于有一个 fH × fW 的像素窗口，窗口下有输入特征的像素值，计算窗口下的权重与像素值加权和即为输出特征对应通道上窗口中心位置的特征值。通过滑动窗口与计算窗口下的加权和，就可以获得这个输出特征的所有像素值，当然这么说并不准确，在特征边缘会没有窗口中心覆盖，但可通过填充数值来实现整个完整特征的输出。

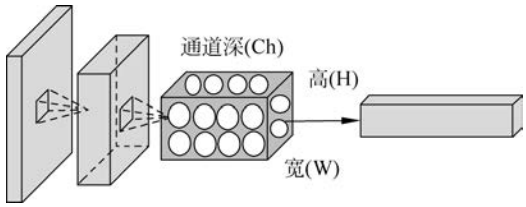

图 1.2.2 卷积神经网络逻辑结构

激活函数根据需求不同有多种形式，比如 Sigmoid 激活函数、Relu 激活函数、Tanh 激活函数等，关于不同函数的特点与用法这里不再赘述。

卷积操作的作用是使特征变深，池化层的作用则是对特征图进行压缩。与卷积类似，池化操作也有一个滑动的窗口，不过池化操作没有权重，而是对窗口内的像素进行下采样，比如取均值或是最大值，将窗口内的特征图下采样为一个像素。卷积神经网络随着网络层数加深，特征图的长、宽逐渐变小，深度逐渐加深。

全连接层的本质是向量乘积，设 $\boldsymbol{x}$ 为全连接层的输入特征向量，W 和 b 分别表示全连接层的权重和偏移量，则输出的特征向量 $\boldsymbol{y}$ 为

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b} \tag{1.2.9}$$

全连接层在卷积神经网络中起到“分类器”的作用，“压缩”地表示了原输入特征的语义信息，这一特征可用于分类回归等不同的任务。

综上可以看出，卷积神经网络使用深层的网络结构，对特征图进行局部感知，随

后在更高层去综合局部信息，最终得到全局信息，符合人类视觉皮层神经元局部接受信息的特点，而卷积核的权重共享特点使得其相对于全连接参数更少，运算量更小。

(2) 循环神经网络：卷积神经网络的输出都只考虑当前的输入，而很多时候要解决的问题的输入是一个序列，比如视频流和文本。而序列中每个时刻的结果可能不止受当前时刻输入的影响，还受先前时刻输入的影响，这个时候只受当前时刻输入影响的卷积神经网络便不再胜任，可选用专门处理时间序列问题的循环神经网络(recurrent neural network，RNN)。

循环神经网络是一个具有“记忆功能”的神经网络，它在每一个时间步都会通过隐藏层保存前面的信息并用于当前时刻的计算。在结构上隐藏层之间的节点是有连接的，而隐藏层输入也不再是当前输入，还包括上一时刻的输出。

图 1.2.3 中直观地展示了 RNN 的逻辑结构，网络由输入层、隐藏层和输出层组成，从图中可以看出，隐藏层保留上一时刻的输出，并与当前输入共同输入到隐藏层中，最后得到网络的输出。其数学表达可见式(1.2.10)：

$$\begin{aligned} S_t &= f(WS_{t-1} + UX_t) \\ O_t &= g(VS_t) \end{aligned} \tag{1.2.10}$$

其中，S_t 为 t 时刻的隐藏层状态，X_t 和 O_t 分别为 t 时刻的输入和输出，f 和 g 为激活函数，W、U、V 为网络中的权重。循环神经网络的权重在每个时刻共享参数。

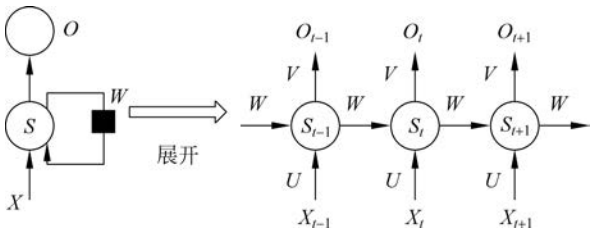

图 1.2.3 RNN 逻辑结构

然而这样的结构存在一些问题，当时间序列较长时，反向传播求梯度时可能会发生梯度消失或梯度爆炸。长短期记忆(LSTM)网络和门控循环单元(GRU)可以很好地解决这一问题。

LSTM(逻辑结构见图 1.2.4(a))在普通 RNN 基础上，加入 4 个门来控制去除或者增加信息到网络隐藏层中。这 4 个门分别是决定丢弃什么信息的遗忘门、决定存放什么信息的输入门和细胞门，以及更新隐藏层中旧细胞状态的输出门，4 个门的数学表示为

$$\begin{aligned} g_f &= \text{sigmoid}(W_{i,f}x_t + W_{h,f}h_{t-1} + b_f) \\ g_{in} &= \text{sigmoid}(W_{i,in}x_t + W_{h,in}h_{t-1} + b_{in}) \\ g_c &= \tanh(W_{i,c}x_t + W_{h,c}h_{t-1} + b_c) \\ g_o &= \text{sigmoid}(W_{i,o}x_t + W_{h,o}h_{t-1} + b_o) \end{aligned} \tag{1.2.11}$$