SLAM 中的几何与学习方法

Geometric Approaches and Neural Networks for SLAM Systems

李言

Federico Tombari

updated 2024 年 7 月 20 日

目录

说明：文章中的实验结果图片，除特殊说明以外，均来自我们自己的实验。读者可以引用，引用时，请说明出处。

```
@misc{gl-SLAM,
author = {Yanyan Li},
title = {SLAM中的几何与学习方法},
year = {2020},
publisher = {GitHub},
howpublished = {\url{https://github.com/yanyan-
    li/SLAM-BOOK}
}
```

Part II

SLAM 基础理论

4 多视图几何基础（Multiple View Geometry）

计算机视觉基础是一个很庞大的课题，为了更贴近 SLAM 的研究主题，我们聚焦在视觉几何的部分，而这一章主要介绍 SLAM 中常用的图像/数据处理方法。

相机位姿估计和地图问题是系统性的，需要许多模块。在本节中，我们重新审视这个过程的核心模块，从图像输入开始，最终生成如图 1 所示的因子图。该过程可以总结如下。当我们将图像序列输入系统时，首先从帧 F_i 中提取 2D 特征 $f_{c_i}^j$。然后，通过不同的特征跟踪策略，包括描述符 [5,10] 和光流 [2]（详见第 4.1 节），获得其在下一帧 F_{i+1} 中的对应特征 $f^{j'}c_{i+1}$。基于这些对应关系，我们可以估计相机坐标 C_i 到 C_{i+1} 的相对位姿 $Ti+1,i$，以及世界坐标中的 3D 地标 P_w^j（详见第 **??** 节）。在关联步骤 **??** 之后，我们将得到观测关系，显示了帧 F_i 在像素位置 $\mathbf{p}_{c_i}^j$ 处测量的 3D 地标 P_w^j。因此，最终可以生成图（见图 1）

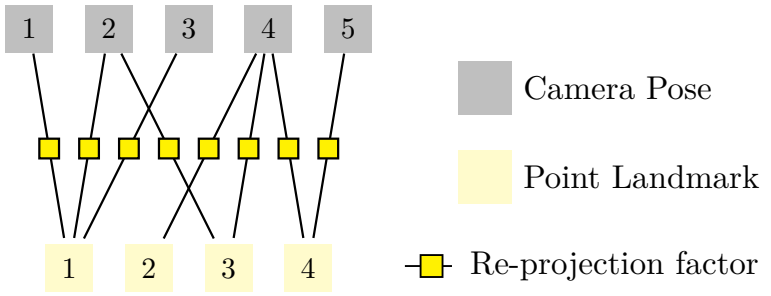


图 1: SLAM 问题的因子图表示。

4.1 特征提取与匹配 (Feature Detection and Matching

通常，在 SLAM 中，特征指的是像素（如角点）或像素组（如线和平面），它们可以直接从 RGB（深度）图像中提取，因为它们与周围的像素有明显的区别。在本节中，我们首先介绍在 SLAM 系统中常用的特征，包括点、线和平面，而相应的匹配方法将在后续段落中继续介绍。

特征提取（Feature Detection）。 特征检测操作是 SLAM 系统中最关键的步骤之一，其目标是找到当前输入的独特线索。因此，这个模块经常被调用，几乎在每个输入帧上都执行。有许多方面可以用来表示输入帧的特殊

性。然而，我们将那些可以在后续的姿态估计和重构模块中使用的线索称为特征。本节详细介绍了三种常见的特征：点、线和表面。点、线和平面检测方法的详细信息在第 **??** 节中进行了说明。

特征跟踪。 在从帧 F_i 和 F_{i+1} 中提取特征之后，我们接下来需要解决的问题是从两幅图像中选择相同的特征。如果两幅图像观察到场景中的相同位置，则这两幅图像上的三维点的投影被称为对应关系。因此，获取对应关系的过程被称为特征跟踪。本节的目的是介绍如何匹配这些特征。

描述符是 SLAM 方法中常用的方式 [8,9]，用于在不同帧之间跟踪对应关系。该方法的核心思想是使用抽象的代码来唯一表示每个特征。SIFT [1] 算法生成一个大小为 256×1 的描述符向量，对光照变化、方向和均匀缩放具有不变性。ORB 算法中使用的 BRIEF 方法 [3] 为点提供了二进制描述符，可以在对应关系匹配过程中通过汉明距离来衡量。

光流是基于灰度不变性假设的一种有效的特征匹配方法，即 $\mathbf{I}(u+du, v+dv, t+dt) = \mathbf{I}(u,v,t)$，其中 $\mathbf{I}(u,v,t)$ 是时刻 t 时像素 (u,v) 的灰度值，像素的新位置在时刻 $t+dt$ 移动到 $(u+du, v+dv)$。光流算法通过计算每个像素从一帧到下一帧的位移矢量 (du, dv) 来跟踪这些特征。Lucas-Kanade 方法 [7] 广泛用于稀疏光流估计。基于灰度不变性假设，我们可以得到如下公式

$$\begin{bmatrix} \mathbf{I}_u & \mathbf{I}_v \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix} = -\mathbf{I}_t \tag{4.1}$$

这里，V_x 和 V_y 分别是像素在 x 轴和 y 轴上的运动速度，$\mathbf{I}_u$、$\mathbf{I}_v$ 和 $\mathbf{I}_t$ 是图像 I 关于位置 u、v 和时间 t 的偏导数，在当前时刻点 (u,v) 处进行评估。当以 (u,v) 为中心的 $w \times w$ 窗口中的像素具有相同的运动时，我们可以获得窗口内像素的 $\begin{bmatrix} V_x & V_y \end{bmatrix}^T$。为提高算法的鲁棒性，光流跟踪过程中使用了关于加权窗口和图像金字塔的策略。

4.1.1 点特征提取与匹配

经常被使用的特征有 SIFT(Scale Invariant Feature Transformation) [19] 、SURF(Speeded up robust feature) [20] , ORB(Oriented fast and rotated BRIEF) [21] 等。这些特征通过不同的算法获得, 也有不同的应用场景。

SIFT 特征 SIFT 特征匹配算法是早期非常著名的特征描述和匹配算子, 为计算机视觉的发展起到了巨大的推动作用。这种特征可以很好的应对图像对之间的平移、旋转、尺度变换, 众多的实验表明了 SIFT 在上述情况下的鲁棒性。使用 SIFT 算子惊醒特征提取工作主要包括两个步骤: 分别是特征描述向量的生成, 如图 4 所示; 另一个是特征描述向量之间的匹配。同时这种两步方法也被后来众多的特征匹配算子继承, 甚至影响着使用深度学习完成匹配工作的网络框架。SIFT 算法可以很好的应对尺度、旋转情况下的

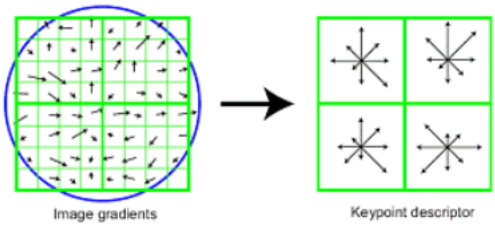


图 2: SIFT 特征提取的 128 维向量

匹配, 但是该算法建立在两幅图像满足相似变换这一假设。因此该算法适合处理相两幅图像邻角度小于 30°C 的图像, 对于较大倾角的倾斜图像,SIFT 只能提取很少量的特征, 同时伴随着较大比例的错误匹配, 因此 ASIFT 算子 [22] 被设计出来解决大倾角图像的匹配问题。

SURF 特征 SURF 算法同样是经历局部特征点的提取、描述和特征点的匹配, 但它一种稳健的具有尺度和旋转不变性的局部特征点检测和描述算法, 它降低了特征描述向量的维度, 同时这个算子更加容易实现并行处理。这种算子是对 SIFT 进行改进, 同样使用到图像空间金字塔等结构。不同于 SIFT 算法, 它使用黑塞矩阵 (Hessian Matrix) 和盒式滤波器去描述兴趣点 (Faster-Hessian Detector)。在特征的描述环节, 该方法首先是在兴趣点周围的圆形区域计算方向, 然后再根据选择的方向构建方形区域, 并从中提取特征点描述符。SURF 特征是一种具有较高准确度和速度的特征算子。

ORB 特征 由于 ORB-SLAM 系列和相关论文的影响，ORB 这种特征变得非常流行，它运行速度非常的快，我在比较好算力的台式机上从 $480*664$ 图片提取 ORB 特征并建立描述子需要 8ms 左右。那么 ORB 中有那些操作呢?

```
    ComputePyramid(image);
    ComputeKeyPointsOctTree(allKeypoints);
    _descriptors.create(nkeypoints, 32, CV_8U);
```

首先，我们将图片建立金字塔，在金字塔每层图像上进行 FAST 特征提取。对于感兴趣的点，如上图所示的 p 点，以该点为中心，3 像素为半径，获得圆周上的 16 个像素。如果圆周上有连续 n 个像素点的灰度值大于或小于该点的灰度值，则认为该点为 FAST 特征。

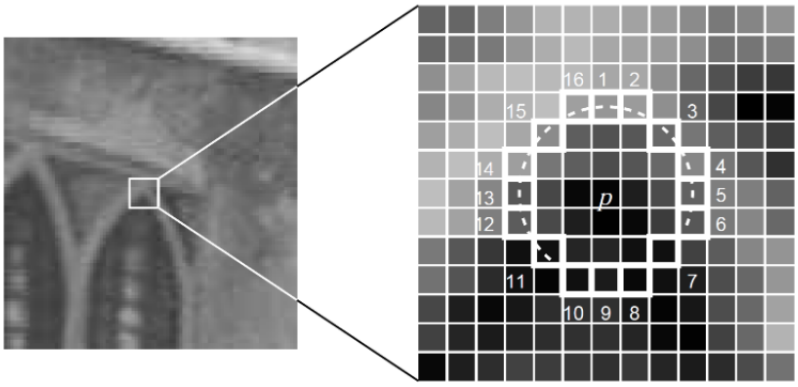


图 3: 感兴趣点 p 和圆周上的 16 个像素。 [?]

其次，我们要去除不好的 FAST 点。FAST 特征点和该点周围的 16 个像素输入决策树，筛选出来高质量的 FAST。对于临近的特征点，选择具有较大区分度的特征点，删除临近的其余点。

为了实现特征点的旋转不变性。对于特征点和 r 半径内的像素点，计算 r 半径范围内的质心，并建立从特征点位置到质心位置的向量，这个向量就是该特征点的方向。具体的计算过程如下所示：

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \tag{4.2}$$

这里的 x 和 y 表示特征点为中心，r 为半径的圆内点的坐标，他们范围是 $(-r, r)$。利用上面的动量公式：我们可以计算重心：

$$C = (\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}) \tag{4.3}$$

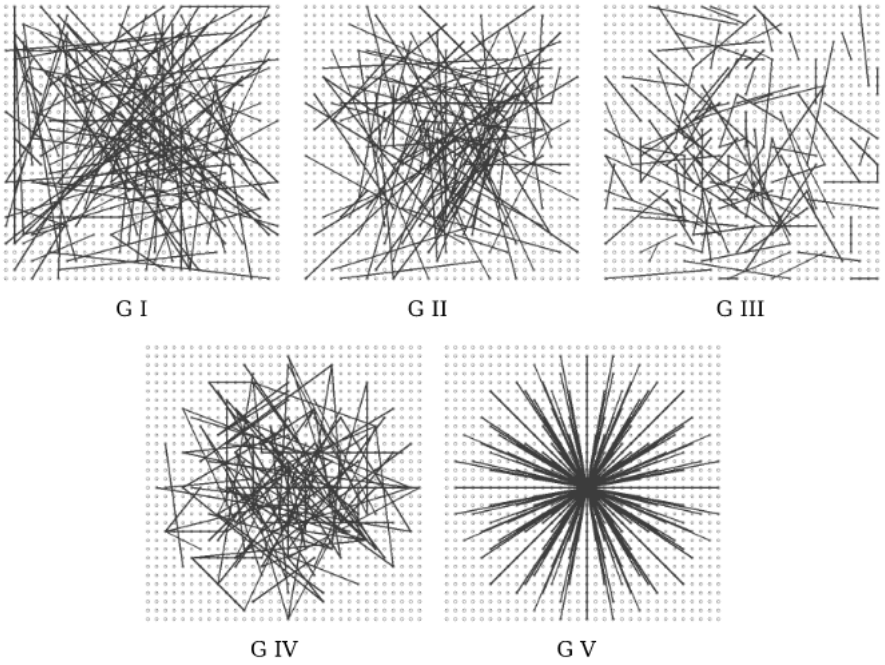


图 4: BRIEF [11] 不同选择点的方式。1. x,y 方向平均分布采样；2.x,y 均服从 Gauss(0,$S^2/25$) 各向同性采样；3.x 服从 Gauss(0,$S^2/25$)，y 服从 Gauss(0,$S^2/100$) 采样；4. x,y 从网格中随机获取；5.x 一直在 (0,0)，y 从网格中随机选取。

因此围绕这个特征点的图像块的方向可以表示出来了：

$$\theta = atan2(m_{01}, m_{10}) \tag{4.4}$$

上述公式中的 $atan2()$ 是象限敏感（quadrant-aware）的反正切函数。

在获得特征点的位置和方向之后，我们一起来看如何为这个特征点建立描述子。ORB 在 BRIEF 的基础上改进出来自己的描述方法，我们先来回顾原始的 BRIEF 的过程。对于 $n = 256$ 个点对 (p_i, p_j)，每一对之间的灰度值 I 高低就产生了一个二值化结果。

$$r(p_i, p_j) = \begin{cases} 1 & I(p_i) < I(p_i) \\ 0 & I(p_i) > I(p_i) \end{cases} \tag{4.5}$$

因此 256 个像素对就构成了一个 256 维的向量。那么 BRIEF 是怎么选择

这些点对呢？如图4所示，原论文给出了 5 种不同的思路，但是作者使用第二种：两个点服从高斯 (0,$S^2/25$) 各向同性采样。

与传统 BRIEF 相比，ORB 为描述子增加了方向信息，其被称为 Steered BRIEF。原先的 n 个像素对构成了矩阵 S，

$$S = \begin{pmatrix} p_i^0 \cdots & p_i^1 \\ p_j^0 \cdots & p_j^1 \end{pmatrix} \tag{4.6}$$

把方向信息考虑在整体的描述子中，我们可以获得 ORB 特征构建的具有旋转不变性的描述子：

$$S_\theta = R_\theta S \tag{4.7}$$

上述公式中的 $R_\theta = \begin{bmatrix} cos\theta sin\theta \\ -sin\theta cos\theta \end{bmatrix}$。

4.1.2 线面特征提取与匹配

线面特征被广泛的应用在在 SLAM 中，这些传统的特征提取与匹配方式，经过长时间的发展，已经逐渐稳定。LSD 是比较流行的一个直线提取方法，KNNMatch 做特征描述。同时，还有特征质量的简单筛选。

```
using namespace cv;
using namespace std;
using namespace cv::line_descriptor;
struct sort_descriptor_by_queryIdx
{
    inline bool operator()(const vector<DMatch>&
       a, const vector<DMatch>& b){
        return ( a[0].queryIdx < b[0].queryIdx );
    }
};
struct sort_lines_by_response
{
    inline bool operator()(const KeyLine& a,
       const KeyLine& b){
        return ( a.response > b.response );
```

```
158         }
159     };
160
161     void ExtractLineSegment(const Mat &img, const Mat
162         &image2, vector<KeyLine> &keylines,vector<
163         KeyLine> &keylines2)
164     {
165         Mat mLdesc,mLdesc2;
166
167         vector<vector<DMatch>> lmatches;
168
169         Ptr<BinaryDescriptor> lbd = BinaryDescriptor
170             ::createBinaryDescriptor();
171         Ptr<line_descriptor::LSDDetector> lsd =
172             line_descriptor::LSDDetector::
173             createLSDDetector();
174
175         cout<<"extract lsd line segments"<<endl;
176         lsd->detect(img, keylines, 1.2,1);
177         lsd->detect(image2,keylines2,1.2,1);
178         int lsdNFeatures = 50;
179         cout<<"filter lines"<<endl;
180         if(keylines.size()>lsdNFeatures)
181         {
182             sort(keylines.begin(), keylines.end(),
183                 sort_lines_by_response());
184             keylines.resize(lsdNFeatures);
185             for( int i=0; i<lsdNFeatures; i++)
186                 keylines[i].class_id = i;
187         }
188         if(keylines2.size()>lsdNFeatures)
189         {
190             sort(keylines2.begin(), keylines2.end(),
```

```
191                 sort_lines_by_response());
192             keylines2.resize(lsdNFeatures);
193             for(int i=0; i<lsdNFeatures; i++)
194                 keylines2[i].class_id = i;
195         }
196         cout<<"lbd describle"<<endl;
197         lbd->compute(img, keylines, mLdesc);
198         lbd->compute(image2,keylines2,mLdesc2);//计算
199             特征线段的描述子
200         BFMatcher* bfm = new BFMatcher(NORM_HAMMING,
201             false);
202         bfm->knnMatch(mLdesc, mLdesc2, lmatches, 2);
203         vector<DMatch> matches;
204         for(size_t i=0;i<lmatches.size();i++)
205         {
206             const DMatch& bestMatch = lmatches[i][0];
207             const DMatch& betterMatch = lmatches[i
208                 ][1];
209             float  distanceRatio = bestMatch.distance
210                 / betterMatch.distance;
211             if (distanceRatio < 0.75)
212                 matches.push_back(bestMatch);
213         }
214
215         cv::Mat outImg;
216         std::vector<char> mask( lmatches.size(), 1 );
217         drawLineMatches( img, keylines, image2,
218             keylines2, matches, outImg, Scalar::all(
219             -1 ), Scalar::all( -1 ), mask,
220                             DrawLinesMatchesFlags::
221                                 DEFAULT );
222
223         imshow( "Matches", outImg );
```

```
        waitKey();

    }
```

最后提取的结果是下图展示的。最后大家觉得这种线提取方式，有什么优缺点，可以在下方评论或私信我，一起交流讨论。

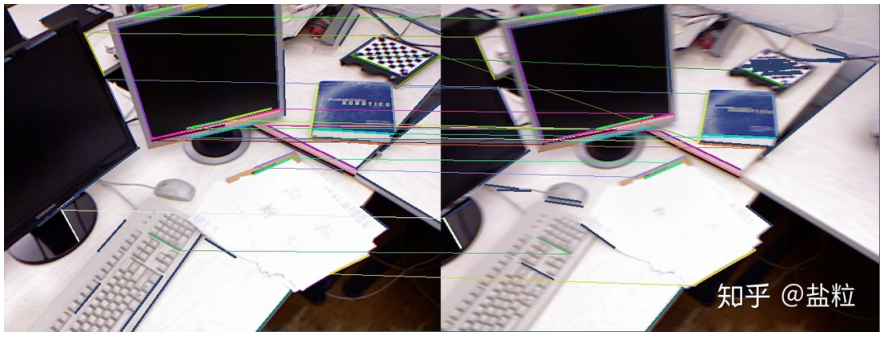


图 5: LSD 直线提取与匹配结果

平面提取 平面提取一般是针对点云而言的，文章 [?] 总结了一个从点云到平面分割的基本方式。

- 首先是从 3D 点云计算 normal map.
- 第一步聚类，具有相似的 normal 方向时，分在一类 S。
- 空间点 × normal 产成距离，根据距离，再把这类等开成不同的 instance。

和上述方法相比，AHCP [4] 提取平面的效果更好一些，当然计算也更复杂一些。每三个连续的 2D 点组成一个节点，就凑成了 a,b,c,d,e,...。通过 AHC 方法（块状聚类），找到 g 具有最小的 MSE，在把 g 与左右进行融合。通过将融合信息与阈值相比，删除误差大的组合。这种每三个点，step=1，就很容易通过 MSE 找出来边界。

AHC 是一个比较好的开源平面提取方案。这个方法分为四个阶段，首先是建立 Initial graph；AHC 分为两个 A，B 两个部分；粗糙的提取面，提取精细的面；如果不需要精细平面的时候，对于 640*480 的图片，可以达到 50Hz。

图 6: 平面提取结果。左图是深度图对应的 RGB 图像，中图是 AHC 方法提取的平面联通区域，右图是平面分割结果。

平面匹配 不像是点线特征，可以用一种特征来描述。平面的匹配大多是使用平面的法向量与原点到平面的距离来判断两平面的参数是不是一样。

4.2 结构特征构建

在这个小节中，我们将特征 SLAM 系统中的约束罗列出来，主要的目标是介绍如何来形成这样的约束，至于如何在 SLAM 系统中利用这些约束，我们会在接下来的章节中逐一介绍。

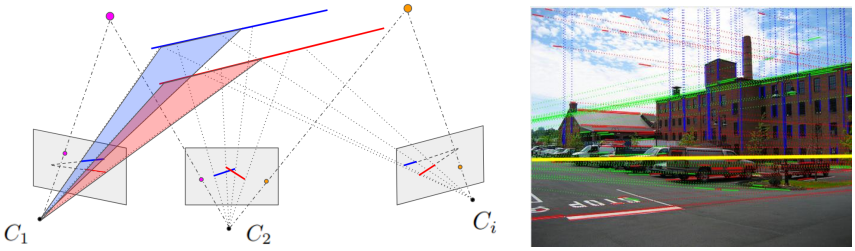
我们从一帧图像上提取了点线面信息，通过探索单帧图像中特征之间的关系，我们可以获得更多的几何约束。

4.2.1 灭点-平行直线集

如图 7所示，3D 场景中的直线被相机观测到，通过投影变换，我们可以在图像上获得 3D 直线的投影，对于空间中的平行直线，图像上的投影直线之间交于一点，图像上的这个 2D 点就称为灭点（vanishing point）。

灭点在 SLAM 中有一些特别的用处，早期一些工作使用他们来做相机参数的计算，现在有些工作利用不同方向上的灭点的正交关系来进行结构化场景中的相机位姿追踪。

我们已经知道，2D 图像上的灭点是平行直线的交点。对于 2D 直线的方程 $ax + by + c = 0$, 我们可以通过联合两条直线的方程求解两条直线的交点 $p = (x, y)$。如果另外直线和上述两条直线平行, 那么 p 点距离该直线的距离应该是比较小的。由于像素误差的存在，平行直线不会是完全交于一点，因此存在很多方法（RANSAC,J-Linkage 等）[?] 去求出最合理的一个点。



(a) 平行直线示例

(b) 灭点与平行直线。红绿蓝颜色代表三个方向上的平行直线，黄色直线是灭线。

图 7: 灭点

第一种方法利用霍夫变换进行灭点检测。对于每一对可能的直线，可以计算其交点的方向，并将这些角度累积到一个参数中。通过选择包含最多投票的参数，可以获得主导的灭点。然而，这种方法对于水平角度参数的量化非常敏感，可能导致多次检测。由于灭点是独立检测的，这种方法不能直接施加正交性约束。高斯球灭点检测算法的思想类似，将图像中的直线映射到高斯球上，在球上划分累加单元，找到交点最多的累加单元对应的球坐标，然后映射回原图像即为灭点的位置。第二种方法使用 RANSAC 对灭点进行检测。具体过程是随机选择两条直线创建一个假设的灭点，然后计算通过该假设灭点的直线的数量。经过多次迭代后，返回能最大化相交直线数量的灭点（同时返回相关联的直线簇）。J-Linkage 是对 RANSAC 多模型的扩展。RANSAC 及其延伸算法（如 multi-RANSAC、J-linkage 等）的理论限制是不能保证解的最优性（对于最大化直线数量而言）。此外，结果不唯一，同一组数据的两次运行可能得到不同的结果。一些方法提出克服这些限制的途径，但受限于线性问题的线性距离函数。

灭点检测 考虑到灭点有可能位于像平面的无穷远处，Barnard 等人 [50] 提出在所谓的“高斯球（Gaussian sphere）”上进行灭点检测。这个球的球心位于相机光学中心，半径为 1 的单位球。他们采用类似 Hough 变换的方式，将每条检测出的直线对应到高斯球上的一个大圆，将灭点对应到众多大圆的交点。通过对高斯球表面进行量化，并记录落入其中的交点的个数，最终得到峰值点处即为灭点。然而，这种方法与一般的 Hough 变换存在相同的缺点，即检测精度受高斯球表面量化程度的限制，且占用较大的内存空间，

同时受噪声的影响较大。

在计算机视觉领域，一种名为 RANSAC（Random Sample Consensus，随机采样一致性）的稳健估计方法被广泛用于对包含大量噪声的数据进行模型参数估计，例如基础矩阵的估计。与通常使用的最小二乘估计不同，RANSAC 的基本思想是不是使用尽可能多的数据点获得模型的估计，而是使用尽可能少的可行数据扩大一致性数据集，最后再使用一致性数据集内的数据点进行最优估计。这种思想使得 RANSAC 方法能够有效剔除数据中的异常值（outliers 或称粗差），从而获得稳健的估计。

然而，由于 RANSAC 方法是为了处理数据集中只存在单个模型实例的情况而设计的，对于多实例模型的稳健估计问题，如从数据集中估计 N 条直线，RANSAC 方法失效。这是因为在这种问题中，同时存在外点和伪外点（pseudo-outliers），后者定义为 “是当前实例的外点，却是其他实例内点的点”。

与 RANSAC 不同，残差直方图分析提供了一个新的思路：分析某个单独数据点在不同模型假设上的残差的分布，并指出该残差的分布模式反映了模型实例的情况。每个数据点的残差分布的峰值对应着真实的模型。残差直方图分析方法可以像随机 Hough 变换一样自动发现模型实例的数量，而不像 multiRANSAC 那样需要预先指定模型数量。然而实验表明，残差直方图分析方法在寻找残差分布模式时存在很多问题，导致实验结果并不理想。

受到 RANSAC 和随机直方图分析的启发，Toldo 和 Fusiello 最近提出了一种新的多实例模型稳健估计方法，即 J-linkage[55]。这种方法在许多实验中被证明优于 multiRANSAC，随机 Hough 变换（RHT）[56]，残差直方图分析等多实例模型估计方法。

J-linkage J-linkage 算法的核心思想是将数据点放入相似概念空间（conceptual space）进行分析，使得同一模型实例的数据点能够在相似概念空间中聚集。在 J-linkage 中，相似概念空间是最关键的概念，它不同于随机 Hough 变换中使用的参数空间和残差直方图分析中的残差空间。参数空间由于需要对连续的参数进行量化，导致了随机 Hough 变换的一些缺点；而残差空间则因残差模式估计的困难而影响了残差直方图分析的实际效果。相似概念空间的概念源自模式识别中的相似概念表示方式，即给定一个对象 x 和 n 个类别，则对象的相似概念表示为其对每个类别的后验概率的向量 $[P(x|C_1),\ P(x|C_2),\ P(x|C_n)]$。在 J-linkage 中，每个数据点的相似概念表示为数据点对于每个模型假设的倾向情况的向量，与残差直方图分析中的

单个数据点对于各模型假设的残差分布相似。

J-linkage 的算法框架包含两个主要步骤：首先是生成数据点的相似概念空间，然后是在该空间中进行聚类。以下分别介绍这两个步骤。

1. 生成相似概念空间：首先，确定相似概念空间的维度 M。该维度对应模型假设的个数，即根据从数据点集合中随机抽取的 n 个数据点计算的模型，其中 n 是确定一个模型所需的最小数据点个数，构成最小采样集（MSS，minimal sample set）。M 的大小与数据点集合中外点的百分比相关，必须足够大以确保算法以某个确定的概率获得包含 K 个不包含外点的最小采样集，其中 K 是用户定义的值。

$$\rho = 1 - \sum_{k=0}^{K-1} \left(\begin{array}{c} M \\ k \end{array} \right) p^k (1-p)^{M-k} \tag{4.8}$$

其中 p 表示从数据点集合中抽样得到一个完全由内点（inliers）组成的最小采样集的概率，这个概率与抽样方法、MSS 的大小有关。

接着，生成 M 个模型假设。J-linkage 采用一种优先选择邻近空间中数据点的方式来生成 MSS，即当一个点已被选择后，接下来一个点被选中的概率取决于其邻近点的数量。

$$P(x_j|x_i) = \begin{cases} \frac{1}{z} \exp^{-\frac{||x_j - x_i||^2}{\sigma^2}} & if x_j x_i \\ 0 & if x_j = x_i \end{cases} \tag{4.9}$$

每个 MSS 生成之后，都可以利用其内的数据点计算出一个模型实例的假设.

最后，定位数据点在相似概念空间中的位置，即用其相似概念表示数据点。通过计算每个 MSS 内的数据点，可以得到一个模型实例的假设。数据点在相似概念空间中的定位需满足一致性阈值，所有满足一致性阈值的数据点构成模型实例的一致集（CS，consensus set）。

$$|F(p_i, \theta_j)| \leq \varepsilon \tag{4.10}$$

这里 εRANSAC 算法中的一致性阈值。那么所有满足公式 4.10的数据点被称为模型实例 j 的一致集（CS，consensus set）。通过一致集构建倾向集合（PS，preference set），描述数据点对于每个模型倾向情况的相似概念。每个数据点的倾向集合用一个二值向量表示。

$$\begin{aligned} PS &= [ps_{i,1},\ ps_{i,2},\ \cdots ps_{i,M}]_{1\times M} \\ ps_{i,j} &= \begin{cases} 1 & |F(p_i, \theta_j)| \leq \varepsilon \\ 0 & otherwise \end{cases} \end{aligned} \tag{4.11}$$

我们称这个向量 PS_i 为数据点 i 的倾向向量。若数据点集合大小为 N，则所有数据点的倾向向量构成 $1 \times M$ 倾向矩阵，表示相似概念空间的表现形式:

$$PM = \begin{bmatrix} PS_1 \\ PS_2 \\ \cdot \\ \cdot \\ \cdot \\ PS_N \end{bmatrix}_{N \times M} \tag{4.12}$$

这个倾向矩阵就是数据点集合的相似概念空间的表现形式。

2. 相似概念的聚类: 相似概念空间生成后，我们可以对其中的 N 个 M 维倾向向量进行聚类。J-linkage 采用一种自底向上的凝聚聚类方式，即从包含单个实例的类别开始，每次扫描都将具有最小距离的两个类别合并。这种聚类方式被称为凝聚聚类算法，而不同的凝聚聚类算法使用不同的距离测度。J-linkage 算法采用的是 Jaccard 距离，这也是该算法名字的由来。Jaccard 距离是一种用于度量两个集合相似度的指标，它定义为两个集合交集大小与并集大小的差异比值。在 J-linkage 中，对于相似概念空间的倾向向量，Jaccard 距离被用作类别合并的标准。合并具有最小 Jaccard 距离的两个类别，直到满足停止条件。总体而言，J-linkage 算法通过这种自底向上的聚类方式在相似概念空间中寻找并合并相似的模型实例。这种凝聚聚类方法使得算法能够自适应地识别模型实例的数量，并以稳健的方式完成多模型估计。

4.2.2 曼哈顿世界约束-平面与平面垂直

与前面特征构成的约束不同，曼哈顿约束可以整体作为一种约束，这种约束在图像与图像之间可以发挥一作用。

除了图像与图像之间的曼哈顿系统的约束之外，我们还可以将构成曼哈顿系统中的每个平面元素单独做出来平面级（非图像级）的约束。

4.2.3 点线在平面上-共面

点线在平面上带来的约束可以被成为共面约束，如图 9所示，空间点 P_i,P_j 和直线（n_l 与 n_π 的交线）都在平面（法向量 n_π 上）。

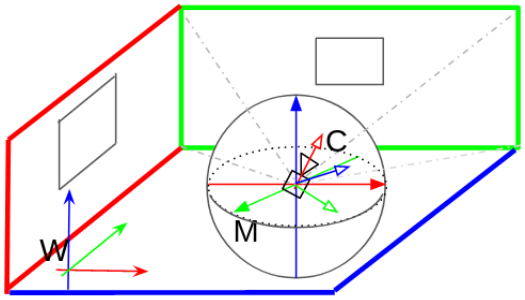


图 8: 曼哈顿世界。不同面之间具有垂直关系。

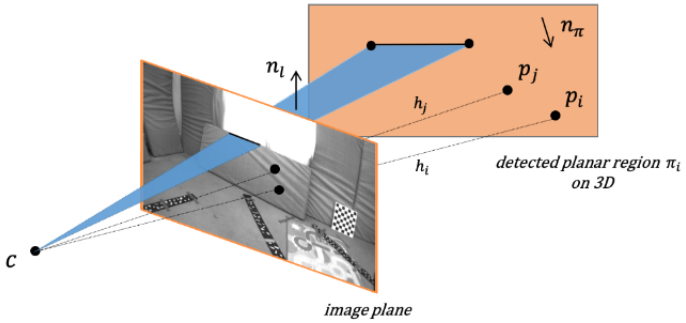


图 9: 点线共面几何 [6]。

共面约束产生的方程也是比较直接的，无非就是点和线的描述要满足平面的方程。

$$\begin{cases} n_x P_x + n_y P_y + n_z P_z + d = 0 \\ n_l \times n_\pi = dir(l) \end{cases} \tag{4.13}$$

上面的公式分别用简单直接的方式介绍点线构成的共面关系。这里点的表示是简单的 (P_x, P_y, P_z) 方式，直线的表示是用普朗克参数形式 (n_l, d) 的形式表示。值得说明的是，不同参数都是可以相互转化的，因此你可以将你喜欢的参数化形式用在共面约束中。

4.3 误差剔除策略

4.3.1 RANSAC

随机抽样一致算法（Random Sample Consensus，RANSAC）[1]采用迭代的方式从一组包含外点的数据中估算出需求模型的参数。

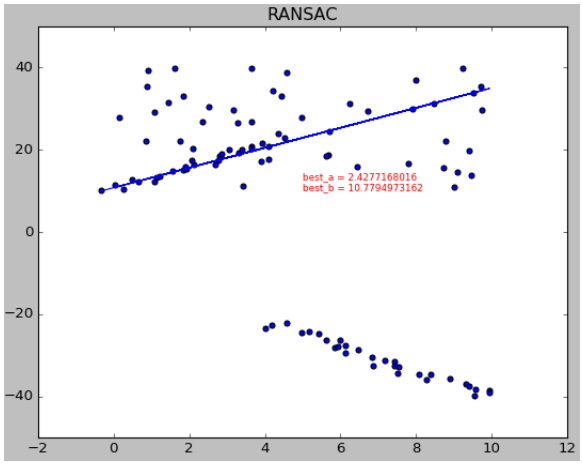


图 10: RANSAC 在 2D 数据中拟合直线。

- 随机从数据 D 中取最小抽样集 S，其中抽样点个数为 n，是模型初始化所需的最小样本数, 基于抽样集 S 估计出模型 M。
- 计算最小抽样集 S 以外的数据与模型 M 之间的距离，距离小于阈值 t_d 的数据构成该模型的内点集 I_M。
- 若 I_M 中的个数大于阈值 t_n, 则认为是找到模型 M 是该数据包含的正确模型。

我们重复以上步骤多次，比较哪次计算中内群数量最多，最多内群点对应的模型就是该数据所要求模型。值得指出的是，图10中输入为两条直线，但是上述介绍的方式找到一个直线后就结束了。

[1] https://github.com/yanyan-li/SLAM-BOOK/tree/master/code/chapter 2

参考文献

[1] Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[2] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International journal of computer vision*, 61:211–231, 2005.

[3] Michael Calonder, Vincent Lepetit, Mustafa Ozuysal, Tomasz Trzcinski, Christoph Strecha, and Pascal Fua. Brief: Computing a local binary descriptor very fast. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1281–1298, 2011.

[4] Chen Feng, Yuichi Taguchi, and Vineet R Kamat. Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6218–6225. IEEE, 2014.

[5] Luo Juan and Oubong Gwun. A comparison of sift, pca-sift and surf. *International Journal of Image Processing (IJIP)*, 3(4):143–152, 2009.

[6] Xin Li, Yanyan Li, Evin Pınar Örnek, Jinlong Lin, and Federico Tombari. Co-planar parametrization for stereo-slam and visual-inertial odometry. *IEEE Robotics and Automation Letters*, 5(4):6972–6979, 2020.

[7] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI'81: 7th international joint conference on Artificial intelligence*, volume 2, pages 674–679, 1981.

[8] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[9] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[10] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[11] Torbjörn Sjöstrand, Stephen Mrenna, and Peter Skands. A brief introduction to pythia 8.1. *Computer Physics Communications*, 178(11):852–867, 2008.