SLAM(simutaneous localization and mapping)算法总结

综述链接：<https://mp.weixin.qq.com/s/TgDeeTqlnrWxBqr4qjj0Xw>

问题

* 后端优化时，优化的是每个位置的姿态？假设所有位置的噪声都是一样的？
* 闭塞是什么意思
* Mahanalobis or Euclidean distance
* 重定位：epnp
* 初始化过程
* Sim3算法
* 理清Harris、hessian、laplacian、DOG、LOG在关键点检测中发挥的作用
* 通过图片标定观察各种角度、旋转拍摄的角点的变形，计算拍摄远近对图像尺寸的影响
* 图像金字塔之间需要两两比较，有一个配对就可以了吗
* 数学求解方法
* 将稀疏矩阵转变成某块为0的方法称为消元：schur、plain
* 求解线性方程的方法：csparse、cholesky、Preconditioned Conjugate

Gradient (PCG)

* 从E、H矩阵分解出位姿：SVD
* SVD求超定方程：对于齐次线性方程 A\*X =0;当A的秩大于列数时，就需要求解最小二乘解，在||X||=1的约束下，其最小二乘解为矩阵A'A最小特征值所对应的特征向量。其原因？

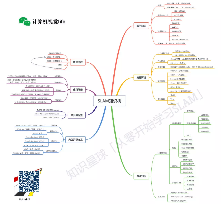
想法

* 跟踪丢失后能不能直接从路标点重定位，路标点可能来自不同的帧，接近的帧共视点比较多

1. 自己总结

* 先通过自主探索SLAM建立地图（尽量遍历整个房间、，建图完成后导航就更方便了，当然一开始也可以使用遥控器来帮助机器人建立完整的地图
* 典型SLAM系统前后端如下图，传统的方法包括运动方程和观测方程，运动方程利用两帧图像计算相机位姿，观测方程用于后端优化。单目摄像头较难计算像素点的深度，需要通过两帧图像计算；双目可以通过一帧图像计算深度；RGB-D摄像头直接得到深度
* 帧间估计：也称视觉里程计，通过前后两帧图像计算摄像头位姿
* 后端优化：优化相机运动轨迹，减小噪声的干扰和累积误差。累积误差会造成全局不一致
* 闭环检测：判断自身是否进入历史同一地点．闭环检测发生时可触发 SLAM 后端全局一致性算法进行地图优化，可以显著减小或消除累积轨迹误差和地图误差．闭环检测问题本质上是场景识别问题。相当于找到原点，那么有两种方法估计其位姿，一种从连续变化来估计当前位姿，这会有累积误差，一种是直接由原点原来帧和当前帧估计。那么比较这两帧就能知道累积误差有多大，可以通过全局BA来减小累积误差
* 地图构建：利用传感器得到的数据建立三维地图
* ORB-SLAM关键帧理解，关键帧可能保存多个，用于下一帧的参考帧

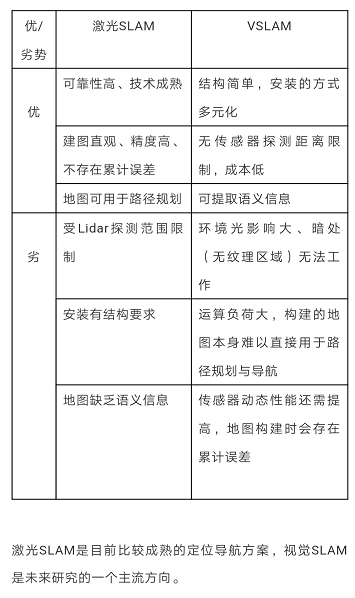
<https://www.cnblogs.com/yepeichu/p/10784265.html>

* 视觉
* 因为观测值有噪声，可通过机器学习的方法进行拟合
* slam知识图
* 视觉slam分类
* 关键帧（feature-based method）：视觉里程计、后端、回环检测、地图构建
* 视觉里程计：首先看到图片，进行图片预处理（图像金字塔怎么做），选取合适的关键帧？如何保存关键帧？保存哪些信息（关键点（角点（位置、方向等）、描述子（周围点））、位姿）？怎么去除冗余？怎么提取和匹配关键点？三角计算方法计算深度？相机坐标、世界坐标、像素坐标之间的转换？重定位方法？如何选择可比较的关键帧（位姿？）匹配的帧越多计算量越大？
* 后端：图优化（BA，）？局部优化什么时候执行？
* 回环检测：如何检测自己到达原来位置？怎么更正轨迹？
* 地图构建：怎么拼接多幅图像的相同点?投影到所有图像平面使（x，y）最小
* 与slam相关的一些任务：object or scene recognition（物体或场景识别）, solving for 3D structure from multiple images, stereo correspondence（双目摄像头找出匹配点）,motion tracking（跟着物体的运动，计算位姿）Camera calibration, 3D reconstruction, image registration，image stitching（图像缝合）
* 英文理解
* viewpoint：摄像头的位姿？viewpoint depth rotation：摄像头与特征点之间连线（深度）旋转的角度？
* affine distortion：The degree of affine distortion is expressed in terms of the equivalent viewpoint rotation in depth for a planar surface。与viewpoint改变有关
* scale change：随着拍摄距离远近，特征点大小的变化？

<https://blog.csdn.net/RobotLife/article/details/87194017>

* 视差：当相机移动时，物体会在图像上形成视差
* stereo baseline：双目之间的基线，双目基线越大，能测量到的距离越远
* trajectory（轨迹）：由各帧各个时刻的位置构成，不包括姿态？包括
* landmark（路标）：环境中的标记，指建图时具有3D世界坐标的点
* motion（运动）：两个时刻间位置x变化，不包括姿态？包括
* 观测（observation？）：在位置x观测到y，然后投影到像素平面
* covisibility：co-visibility共视
* rotation invariant：沿与图像垂直轴的选择
* in-plane rotation：沿垂直轴旋转？
* real time：意味着按帧率处理
* relative pose constraint：两帧之间的变换Tij，T就是constraint，不同位姿下的相机有不同的相机坐标系，位姿得变换可以看成坐标系的变换
* absolute pose：从世界坐标系到相机坐标系变换Ti，
* visual overlap：各不同视角下拍到的图像存在重叠的部分
* tracked feature：是指获得了三维坐标的特征
* observation：在成像平面的像素点z，相机是路标点的observation
* 影响因素总结：image rotation（旋转）、scale change（尺寸改变，远近的影响）、affine distortion（仿射畸变，旋转矩阵变为可逆矩阵，由正方体变成平行四边体）, addition of noise（噪声的增加）, change in illumination（光照变化）、clutter（background混乱）、blur（模糊，可能有速度快造成）、occlusion（特征点只在一副图中被检测出来，不能repeated）、Skew（倾斜？）,anisotropic scaling（各向异性拓展？圆形更具各向同性，每个方向径向距离相等）, perspective effects（感知的影响），texture（纹理，低纹理不好提取特征点）
* 特征的要求：repeatability（关键点，角点，在不同视角图片中可找出来，必须在不同帧中找出来方便匹配）, distinctiveness（描述子，不同关键点可区分开来）, and robustness（描述子，抗干扰等性能），计算速度
* 按编程的思想去解决问题，先整体后局部
* 自动驾驶需要像素200百万左右，80km每小时需要帧率40帧
* 导致跟踪失败的原因：occlusion or aggressive motion, or at system reinitialization。此时需要重定位
* GPU运行数据存于显存，有点像计算机的运存
* 用3D-2D的方法计算出位姿后，
* 计算线性方程方法和从解中分解出位姿
* 双目不具有尺度不确定性是因为基线距离的绝对长度已经知道了，且每次是单独测量的不会累积误差。但单目初始化时绝对长度不知道，只能使用归一化的方法，当然可以初始化设定绝对长度，但之后会随着运动增加尺度漂移，因为它是才初始化的结果不断积累运算的
* 如果每步都用2D-2D那么就会有尺度不确定性问题，第一步确定尺寸后，相当于确定了单位
* Hessian矩阵和高斯、JTJ联系起来
* 求解3D-3D变换矩阵的方法
* 取三对点得到9个约束，丢弃其中2个方程
* close-form方法，其实质也是最小化平方误差：只需一次计算
* 最小平方误差的迭代方法：找到好的初始值，然后迭代
* 正交变换分为两类：第一类是旋转变换,第二类是镜面反射.正交变换的矩阵的行列式等于1或-1,我们规定行列式等于1正交变换称为旋转变换,行列式等于-1正交变换称为镜面反射.
* 所以说“旋转变换的矩阵的行列式为+1”
* 1

1. 数据集

* TUM：包含室内外场景，用手持相机拍的，比较随意，范围较小，时间短，文件名就是拍摄帧的时间，帧率30
* KITTI：小区驾车拍的，范围大，时间长
* Euroc：无人机拍摄，搭载了IMU
* 激光slam和视觉slam对比
* 激光slam精度高、响应快、数据量小，但成本高
* 视觉slam：
* RGB-D摄像头：国外：微软Kinect系列、Intel realsense系列、苹果、英飞凌、TI等为代表的消费级RGB-D；orbbec、pico、human+、爱观、图漾、艾芯智能、知微传感等；英特尔的RealSense、华硕的Xtion

1. 角点的理解：

<https://blog.csdn.net/yizhang_ml/article/details/86994193>

1. 随机抽样一致 (RANSAC，Random Sample Consensus)算法：注意理解局外点（异常点）和局内点（正常点）

<https://blog.csdn.net/robinhjwy/article/details/79174914>

1. SVD分解

<https://www.cnblogs.com/endlesscoding/p/10033527.html>

1. SVD解超定方程：对于齐次线性方程 A\*X =0;当A的秩大于列数时，就需要求解最小二乘解，在||X||=1的约束下，其最小二乘解为矩阵A'A最小特征值所对应的特征向量。

<https://blog.csdn.net/potxxx/article/details/86761222>

1. 模拟退火

<https://www.cnblogs.com/peng-ym/p/9158909.html>

1. SIFT理解

<https://www.cnblogs.com/ronny/p/4028776.html>

<https://www.cnblogs.com/jsxyhelu/p/7591497.html>

1. Surf特征点

<https://www.cnblogs.com/zyly/p/9531907.html>

<https://blog.csdn.net/blateyang/article/details/76512398>

1. Fast、fast-er

<https://blog.csdn.net/ssw_1990/article/details/70569871>

<https://blog.csdn.net/tostq/article/details/49335135>

1. ORB：原文用来证明steered BRIEF相关性大的方法两种方法：

1.PCA分解矩阵得到特征值并从大到小排序，其中BRIEF 和rBRIEF 的特征值在前十几个特征值后迅速下降，steered BRIEF的特征值却比较均匀；

2.局外点和局内点各自的描述子距离（两描述子的1-范式）两种方法。

<https://blog.csdn.net/kevin_cc98/article/details/75123316>

1. 单应矩阵

<https://www.cnblogs.com/wangguchangqing/p/8287585.html>

1. 1

网页文章：

1、较好资料《SLAM for dummies》；

2、现在经典的方案是“图像前端，优化后端，闭环检测”的三部曲；

3、Kinect。众所周知这是一款深度相机，你或许还听说过别的牌子，但Kinect的价格便宜，测量范围在3m-12m之间，精度约3cm，较适合于小萝卜这样的室内机器人。

4、对姿态估计，经典的算法是ICP（Iterative Closest Point，迭代最近点）。这个算法要求知道这两个图像间的一组匹配点，说的通俗点，就是左边图像哪些点和右边是一样的。你当然看见那块黑白相间的板子同时出现在两张图像中。在小萝卜看来，这里牵涉到两个简单的问题：特征点的提取和匹配。

5、自己动手编程需要学习大量的先决知识。首先你要会C和C++，网上很多代码还用了11标准的C++。第二要会用Linux。第三要会cmake，vim/emacs及一些编程工具。第四要会用openCV, PCL, Eigen等第三方库。只有学会了这些东西之后，你才能真正上手编一个SLAM系统。如果你要跑实际机器人，还要会ROS。

6、最麻烦的问题，就是“噪声”。这种渐近式的匹配方式，和那些惯性测量设备一样，存在着累积噪声。因为我们在不断地更新关键帧，把新图像与最近的关键帧比较，从而获得机器人的位移信息。但是你要想到，如果有一个关键帧出现了偏移，那么剩下的位移估计都会多出一个误差。这个误差还会累积，因为后面的估计都基于前面的机器人位置。

7、滤波器发展：kalman滤波器、Graph-based SLAM。

8、事实上，小萝卜在探索房间时，经常会左转一下，右转一下。如果在某个时刻他回到了以前去过的地方，我们就直接与那时候采集的关键帧做比较，可以吗？我们说，可以，而且那是最好的方法。这个问题叫做闭环检测。

9、有两种思路：一是根据我们估计的机器人位置，看是否与以前某个位置邻近；二是根据图像的外观，看它是否和以前关键帧相似。目前主流方法是后一种，因为很多科学家认为前一种依靠有噪声的位置来减少位置的噪声，有点循环论证的意思。后一种方法呢，本质上是个模式识别问题（非监督聚类，分类），常用的是Bag-of-Words (BOW)。但是BOW需要事先对字典进行训练，因此SLAM研究者仍在探讨有没有更合适的方法。