第16周汇报

——张溢炉

1. 《Fast Relocalisation and Loop Closing in Keyframe-Based SLAM》总结
2. 内容摘要

本文介绍了ORB-SLAM算法中重定位和闭环检测的方法

1. 总结及理解

* 提取500个角点时使用non-maximum suppression会改善角点分布，但是会降低匹配表现（特征点少了），故提取1000个角点
* 自动初始化时是2D-2D问题，重定位时3D-2D问题，闭环检测是3D-3D问题
* 在做一致性检测时，之所以选择3个方向，是因为在不同深度旋转的角度不一样
* 闭环检测：认为loop帧数据是比较准确的
* 选取候选帧
* 计算当前帧和闭环帧之间的相似变换Sk,l，算的过程中利用RANSAC作了几何检验。首先计算匹配点；利用无三维坐标匹配点同一张图周围最近3个有3D维坐标（两帧图都是用loop帧的路标点）的点计算该匹配点3D坐标（相机坐标系下），丢弃距离超过10个像素的点；对两帧进行上述操作后得到3D-3D点，然后利用文献[26]方法计算相似变换矩阵；然后验证，匹配点重投影小于6个像素表示是内点，如果在RANSAC前70次迭代中找到支持40%匹配点为内点的相似变换，则表示成功；最后利用所有内点在计算相似变换Sk,l
* 优化pose，因为BA有huber函数，漂移太大很难收敛。具体方法见文献[25]。detaSij是通过相似变换算出，闭环内两两帧之间变换，SjwSiw是由pose计算得出，r为residual error，理想状态等于1，这时两种方法恢复的值相同。目的是不断迭代优化pose，然后重复计算相似变换，使目标函数接近于0。协方差矩阵/\是指定的，不为单位矩阵时，表示不同自由度之间相互影响，但其常是单位矩阵。优化方法：g2o+LM+CHOLMOD



优化完pose后，利用如下公式校准每个路标点xj，使用Ti作为参考，接近于loop帧的位姿

* 路标点融合，如在loop看到的点，在当前帧又看到了，就会出现同一个物理点，在两帧中计算出了两个不同的3D坐标。将同一物理点（匹配点）在loop帧上世界坐标和当前帧该点的世界坐标一起投影到当前帧，若相距低于20像素，擦除当前帧的世界坐标。然后按同一的方法，融合loop周围关键帧与当前帧共视路标点，直到没有点可以擦除停止。在循环上诉操作，融合loop帧及其相邻帧与当前帧之前帧，直到没有点可以擦除。在点被擦除后就需要更新graph的边，包括帧之间以及帧与路标点的边
* 校正未参与闭环检测过程的新帧，在闭环检测当前帧之后前端又处理了一些新的帧。只需校正最新帧与闭环检测当前帧即可，其他帧会很快被局部优化。校正公式如下，这里只需利用尺寸校正的尺寸sk，l，注意校正前后标志

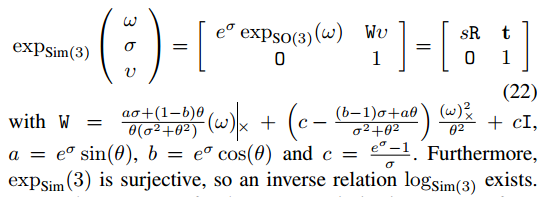


* 重定位：
* 为了更快的选出候选帧，不像闭环检测那样设置最小阈值alpha和一致性检测。在后面的几何检测以及在重定位之后跟踪超过20帧才会插入关键帧的设置，可以增强其鲁棒性。
* 使用闭环检测相似的方法恢复一些3D点，然后利用3D-2D+RANSAC计算位姿，如果在RANSAC前178次迭代中找到支持40%匹配点(至少20个点)为内点，则将该帧作为参考帧，若接下来超过20帧跟踪成功，重定位才算真正成功

1. 《Scale Drift-Aware Large Scale Monocular SLAM》总结
2. 内容摘要

本文主要讲述使用7dof优化的方法，没有说到两帧之间相似变换的求解方法，具体方法见下一篇。

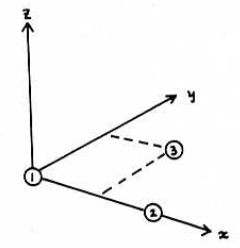
1. 总结及理解

* 单目相机有7个自由度会漂移，因为它是通过累加的方法计算尺寸，而双目、深度相机只有6个自由度，与单目相似，当前变换是不断利用上一次位姿获得的。这也是靠前的位姿和尺寸会更准确一些的原因
* sim（3）的李代数表示形式如下

1. 《Closed-form solution of absolute orientation using unit quaternions》总结
2. 内容摘要

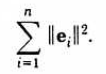
本文讲述一种计算已知一些点在自己坐标系下三维坐标的两帧图像之间的相似变换方法，该方法不用迭代，只需计算一次，使用于3个点及以上的情况。其中，将旋转用四元数表示。

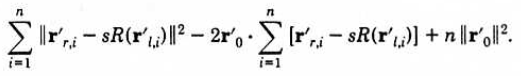
1. 总结及理解

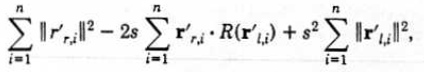
* 求解两坐标系之间相似变换方法分类：
* 单目具有7个自由度，那么取三个点就能找到9个约束，去掉其中两个约束便能求出解
* 迭代的方法：最小二乘方法
* close-form solution：如本文方法，给出匹配点在两个坐标系的坐标，其不需要迭代，只需一次计算，另外，其并不需要很好的初始值。
* 计算平移量：首先得到很多左右两个坐标系的匹配点。如下公式理解：假如r为向量，想象相机的带着一个坐标系在运动，先旋转，将两个坐标系方向变成一样，再变换尺寸，因为远近不一样，物理不变的长度，在近的坐标系变长，当然因为视角不同，物体形状还会畸变，这里先不考虑，故需要放缩一下；最后平移，r0是在右坐标系下的平移量。
* 具体方法
* 有选择的丢弃一些约束，使用已知在左右两个坐标系坐标的不在一条直线三个匹配点计算旋转。以其中2点作为x轴，2个点中一个作为原点，以3个点平面作为xy平面，将各轴化为单位向量。得到如下图坐标系，左右坐标系进行相同操作，这样就得到三个点在左右坐标系下的坐标系。因为原点相同，所以丢弃了平移，将各轴化为单位向量解决了尺寸问题，这样两坐标系只存在旋转，得到如下旋转公式，先转换为世界坐标系，再转换到右坐标系



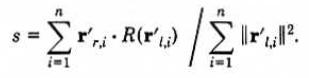
* 本文方法：给出n个匹配点在左右两个坐标系的3D坐标，最小化重投影误差，如下公式：



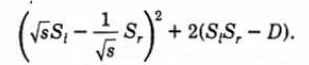
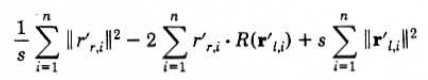
* 通过中心化可化为如下公式，第二项为0，第一项与s、R相关，第三项与r0相关
* 令第三项为0，有
* 第一项可化为不对称和对称两种情况：
* 不对称情况：两个坐标系测量精度差别很大时，将第一项进行恒等变换有：

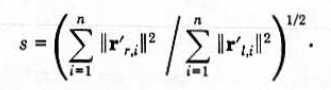


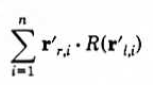
令第一项为0有：

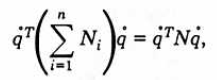
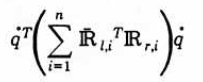


* 对称情况：两个坐标系测量精度差别不大时，将第一项进行恒等变换有



令第一项为0，这里s与R无关，有

* 对于以上两种情况，D越大越好，无论D多大约束的存在最后结果肯定大于0，D越大值越小，D公式如下
* 然后使用单位四元数可将其化为



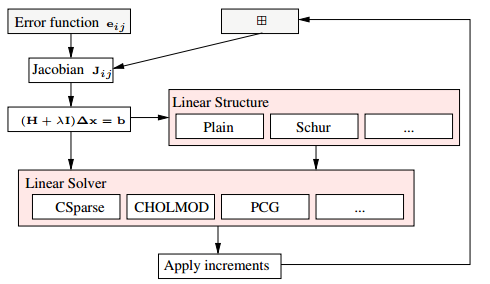
* A3证明欲求q使qTNq最大，只需求得N的最大特征值对应的特征向量作为q即可，从而求得旋转
* 然后使用上面s公式求出s
* 最后使用上面r0公式求出r0
* 当匹配点共面时存在两种特殊的旋转：旋转前后两坐标系交于一条直线和旋转轴为面的法向量。具体计算方法见论文

1. 《g2o: A General Framework for Graph Optimization》总结
2. 内容摘要

本文讲述g2o的原理

1. 总结及理解

* 整个框架如下图，g2o采用迭代方法操作，图中灰色部分error function和田操作（累加方式）是需外部指定的。另外，为了更高效，可以指定J；为了设计合适的求解方法，可以自己添加linear solver。在求解前可以先使用plain、schur方法消元。



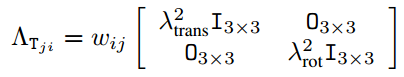
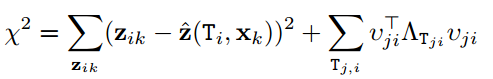
* 本文实验使用了CHOLMOD（cholesky）、PCG来求解。前者计算速度快，但不能利用参数的块结构。后者迭代次数多，计算速度慢，但能利用参数块结构。

1. 《Double Window Optimisation for Constant Time Visual SLAM》总结
2. 内容摘要

本文采用inner和outer窗口用于用于局部地图优化，利用了pose-point（Ti）和pose-pose（Ti,j）两种约束来优化。该方法能自动建立合适的graph，动态选择并优化inner和和outer窗口

1. 总结及理解

* Full BA太花时间，使用局部地图来优化能保持较好的实时性，包括以下方法
* Active windows：选取全部关键帧的一部分，通常窗口边界的关键帧是固定的，那些与它有共视点的帧会被包含。这种方法比较适合大范围、少闭环的场景，但不适合小范围、闭环比较多的情况，可能会因为包括的关键帧太多而很难收敛
* Relative representation：比较经典的是relative bundle adjustment（RBA），它优化的是两帧之间的相对位置。此方法适合少闭环情况，在闭环比较多的情况精度会下降。
* Pose-pose reduction：相对BA，其只优化pose，不优化路标点，路标点不变。
* Double window：以关键帧Vref作为参考帧，与其及其邻居相连权重最大的前M1帧被取为inner窗口W1，接下来M2帧被取为outer窗口W2，一般M1<<M2。W1帧使用BA优化，W2中一些帧也使用BA优化，W2的所有帧使用pose optimization。两者不需迭代进行，共同优化以下函数即可，优化完后以Vref为参考恢复计算了相对位姿的帧
* 闭环检测：用另一坐标系求出的路标点（两者的匹配点）投影到自己坐标系求出相似变换或齐次坐标，方便后面求残差



* 小闭环检测，首先文中定义了当前帧Vi的两级邻居N1、N2，小闭环候选帧是在N2中找到的，将Vi求出的的路标点投影到Vref求出位姿Tloop，然后删除一些重复的点。最后最小化下面目标函数优化



* 大闭环优化：和之前方法相似