

h_立青人韦

博客园

首页

新随笔

联系

订阅

管理

随笔 - 7 文章 - 0 评论 - 5

orb_slam代码解析(4)LocalClosing线程

现在开始学习下闭环检测线程。

LocalMapping线程把关键帧送到了mlploopKeyFrameQueue队列中，我们检查该队列是否为空，如果有先的关键帧进来，那么就开始进行回环检测。LoopClosing中的关键帧都是LocalMapping中送过来的：送过来一帧，就检查一帧。

Funtion1:DetectLoop()

从队列中取出一个关键帧，作为当前关键帧，并设为不被擦除的特性，接下来判断距离上一次闭环检测是否超过10帧，如果是的话那么把当前关键帧加入mpKeyFrameDB中，并擦除当前关键帧。

遍历所有共视关键帧，计算当前关键帧与每个共视关键的bow相似度得分，并得到最低得分minScore，这是一个阈值，如果得分比这个最低得分还要低的话，那么就是说肯定不会成为闭环关键帧。

Funtion1.1:pKeyFrameDB->DetectLoopCandidates(mpCurrentKF, minScore)

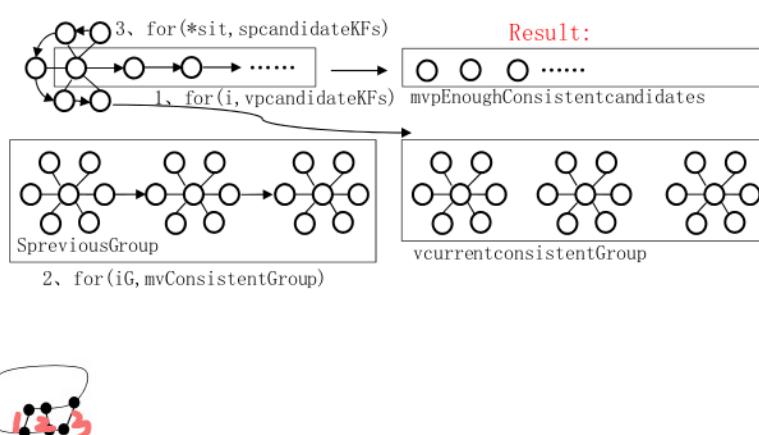
在所有关键帧(pKeyFrameDB)中找出闭环备选帧。

- * 1. 找出和当前帧具有公共单词的所有关键帧（不包括与当前帧相连的关键帧），统计所有闭环候选帧中与pKF具有共同单词最多的单词数，并以这个最多单词数的0.8倍作为一个阈值。
- * 2. 只和共同单词大于最多单词数的0.8倍这个阈值的关键帧进行相似度计算，选出相似度大于 minScore 的关键帧。
- * 3. 单算当前帧和某一关键帧的相似性是不够的，这里将与关键帧相连（权值最高，共视程度最高）的前十个关键帧归为一组，计算累计得分，选出最高的组得分，然后以此的0.75 倍作为阈值，得分大于这个阈值的组，作为得分较高的组。
- * 4. 只返回累计这些得分较高的组中分数最高的关键帧。

ok，这样我们就得到了闭环候选关键帧组。

如果检测结束后，得不到这个候选关键帧组，那么在关键帧数据库里添加此帧，返回false。

接下来检查关键帧的连续性。计算组得分的目的是剔除单独一帧得分较高，但是没有共视关键帧，作为闭环来说不够鲁棒。对于通过了闭环检测的关键帧，还需要通过连续性检测（连续三帧都通过上面的筛选），才能作为闭环候选帧。



最终要达到的目的就是如上图所示，接连3帧检测到的闭环关键帧是联系的。

在程序上的实现：先是进来一组候选关键帧，每一帧将自己以及与自己相连的关键帧构成一个“子候选组”。

公告

昵称：h_立青人韦

园龄：2年2个月

粉丝：4

关注：9

+加关注

2019年1月							>	
<	日	一	二	三	四	五	六	>
	30	31	1	2	3	4	5	
	6	7	8	9	10	11	12	
	13	14	15	16	17	18	19	
	20	21	22	23	24	25	26	
	27	28	29	30	31	1	2	
	3	4	5	6	7	8	9	

搜索





常用链接

我的随笔

我的评论

我的参与

最新评论

我的标签

随笔档案

2018年10月 (1)

2018年4月 (4)

2018年1月 (1)

2017年5月 (1)

最新评论

1. Re:orb_slam代码解析(2)Tracking线程

@h_立青人韦 代码是orb_slam2中文件

LocalMapping.cc,函数void

LocalMapping::CreateNewMapPoints()中的这句话x3D = x3D.rowRa.....

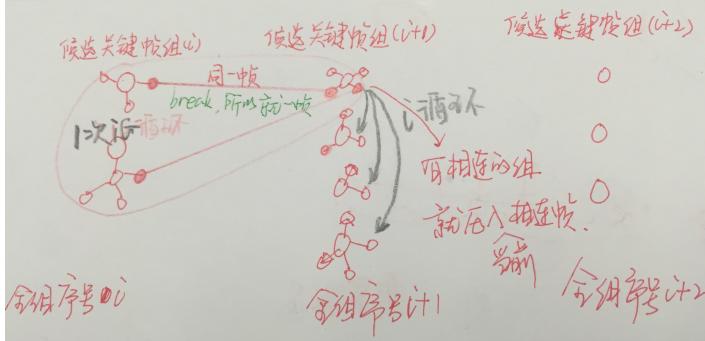
--Max_诸葛小亮

2. Re:orb_slam代码解析(2)Tracking线程

@Max_诸葛小亮我不太清楚你说的是代码中的哪块内容。但如果是4维向量代表空间点，则是齐次坐标的形式，在齐次坐标中坐标的每个分量同乘一个非零常数，仍表示同

一个点，所以除以向量的第四个元素只是强制性最后.....

当系统刚开始，或者未检测到有连续时，“子连续组”里的每一个“子候选组”的序号都为0。之后的关键帧的关键帧组来的时候，每个组里的候选关键帧组成自己的“子候选组”，与之前的“子连续组”里的“子候选组”依次比较。如果相连的话，把相连的“子候选组”和序号压入当前连续组，这个序号比前一组“子连续组”的序号+1。如果nCurrentConsistency大于等于3，那么该“子候选组”代表的候选帧过关，进入mvpEnoughConsistentCandidates。最后，循环结束，当前连续组送给系统连续组用于下一次检测。并把当前帧压入关键帧数据库。



Funtion2:ComputeSim3()

依次从筛选的闭环候选帧中取出一帧关键帧pKF。

Funtion2.1:nmatches = matcher.SearchByBoW(mpCurrentKF,pKF, vvpMapPointMatches[i])

将当前帧mpCurrentKF与闭环候选关键帧pKF匹配，通过bow加速得到mpCurrentKF与pKF之间的匹配特征点，vvpMapPointMatches和GetMapPointMatches在同一个索引下对应的是同一个mappoint，这部分需要和跟踪线程的这个函数做区别。

如果返回的匹配数量多于20个的话，那么就给当前帧和这个关键帧构造sim3求解器。

Funtion2.2:nmatches = new Sim3Solver(mpCurrentKF,pKF,vvpMapPointMatches[i],mbFixScale)

将上一步得到的匹配的mappoint对，表达成各自坐标系下的空间点，并变化为像平面下的坐标。

Funtion2.3:SetRansacParameters(0.99,20,300)

RANSAC的相关内容可以参考：[随机抽样一致](#)。

Funtion2.4:Scm = pSolver->iterate(5,bNoMore,vbInliers,nInliers)

对有较好的匹配的关键帧求取Sim3变换，最多迭代5次，Ransac求解mvX3Dc1和mvX3Dc2之间Sim3，函数返回mvX3Dc2到mvX3Dc1的Sim3变换，也就是候选帧pKF到当前帧mpCurrentKF的Sim3变换（T12）。当这个函数迭代的次数不超过5次或者总的迭代次数不超过300次，那么就任意取三组点根据两组匹配的3D点，计算之间的Sim3变换。

Funtion2.4.1: ComputeSim3(P3Dc1i,P3Dc2i)

这里记录下求解sim3的算法：[参考论文](#)

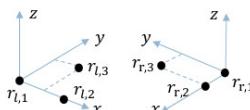
三对匹配3D，分别对左右三个3D点建立坐标系：

$$X\text{轴: } \hat{x}_l = x_l / \|x_l\| \quad \text{其中: } x_l = r_{l,2} - r_{l,1} \quad (1)$$

$$Y\text{轴: } \hat{y}_l = y_l / \|y_l\| \quad \text{其中: } y_l = (r_{l,3} - r_{l,1}) - [(r_{l,3} - r_{l,1}) \cdot \hat{x}_l] \hat{x}_l \quad (2)$$

$$Z\text{轴: } \hat{z}_l = \hat{x}_l \times \hat{y}_l \quad (3)$$

$$\text{右坐标系同理, 且令: } M_l = |\hat{x}_l \hat{y}_l \hat{z}_l| \quad M_r = |\hat{x}_r \hat{y}_r \hat{z}_r| \quad (4)$$



如果左边坐标系有一个向量 r_l ，那么： $M_l^T r_l$ 可以得到 r_l 向量沿着坐标轴的值

左乘 M_r 可以变换到右坐标系，故可推导出旋转：

$$r_r = M_r M_l^T r_l \Rightarrow R = M_r M_l^T \quad (5)$$

计算平移量：

$$\text{质心: } \bar{r}_l = \frac{1}{n} \sum_{i=1}^n r_{l,i} \quad \bar{r}_r = \frac{1}{n} \sum_{i=1}^n r_{r,i} \quad (5)$$

$$\text{原点移到质心: } r'_{l,i} = r_{l,i} - \bar{r}_l \quad r'_{r,i} = r_{r,i} - \bar{r}_r \quad (6)$$

$$\begin{cases} \bar{r}_{l,i} = \bar{r}_{l,i} + \bar{r}_l \\ \bar{r}_{r,i} = \bar{r}_{r,i} + \bar{r}_r \end{cases} \quad \sum_{i=1}^n r'_{l,i} = 0 \quad \sum_{i=1}^n r'_{r,i} = 0$$

$$\begin{aligned} R &= S\mathcal{R}(T_l) + T_o \Rightarrow E = \bar{r}_r - S\mathcal{R}(\bar{r}_l) - T_o \\ R &= \bar{r}'_{r,i} + \bar{r}_r - S\mathcal{R}(\bar{r}'_{l,i} + \bar{r}_l) - (\bar{r}_o + T_o) \\ &= \bar{r}'_{r,i} - S\mathcal{R}(\bar{r}'_{l,i}) - (-\bar{r}_r + S\mathcal{R}(\bar{r}_l) + T_o) \\ \text{优化函数: } &\sum_{i=1}^n \|e_i\|^2 = \sum_{i=1}^n \|r'_{r,i} - [sR(r'_{l,i}) + r'_0]\|^2 \quad (8) \\ &= \sum_{i=1}^n \|r'_{r,i} - sR(r'_{l,i})\|^2 - 2r'_0 \sum_{i=1}^n (r'_{r,i} - sR(r'_{l,i})) + n\|r'_0\|^2 \\ &\sum_{i=1}^n \|r'_{r,i} - sR(r'_{l,i})\|^2 \text{ 与优化量 } r'_0 \text{ 无关} \\ \min \sum_{i=0}^n \|e_i\|^2 &\Rightarrow r'_0 = 0 \Rightarrow t = r_0 = \bar{r}_r - sR(\bar{r}_l) \quad (9) \end{aligned}$$

-h_立青人韦

3. Re:orb_slam代码解析(2)Tracking线程

博主您好，非常感谢您刚才的回答。但我还有一个问题想请教一下。我在ORB_SLAM2的代码中发现，利用SVD分解求出的四维坐标并不是空间点的三维坐标而是需要用前三维除以第四维，归一化得到的结果作为空间点.....

--Max_诸葛小亮

4. Re:orb_slam代码解析(2)Tracking线程

@Max_诸葛小亮因为第三行可以由前两行线性表示，故而省略...

-h_立青人韦

5. Re:orb_slam代码解析(2)Tracking线程

您好，请问在

FUNCTION2.2.1.2.3.1.1:Triangulate中，为什么在三角化求解阶段把DLT分解右侧的向量第三行给省略了？这是什么原因？

--Max_诸葛小亮

阅读排行榜

1. orb_slam代码解析(2)Tracking线程(2977)

2. 针孔的相机成像模型(1149)

3. orb_slam代码解析(3)LocalMapping线程(415)

4. orb_slam代码解析(4)LocalClosing线程(351)

5. ubuntu14.04+ros_indigo+lsdslam(310)

评论排行榜

1. orb_slam代码解析(2)Tracking线程(5)

推荐排行榜

1. orb_slam代码解析(2)Tracking线程(1)

2. orb_slam代码解析(3)LocalMapping线程(1)

3. Count bits set in parallel (查找32位整形数中置1的个数) (1)

计算尺度：

$$\begin{aligned} \text{由于 } r'_0 = 0 &\implies \sum_{i=1}^n \|e_i\|^2 = \sum_{i=1}^n \|r'_{r,i} - sR(r'_{l,i})\|^2 \quad (10) \\ &\implies \sum_{i=1}^n \|e_i\|^2 = \sum_{i=1}^n \|r'_{r,i}\|^2 - 2s \sum_{i=1}^n r'_{r,i} \cdot R(r'_{l,i}) + s^2 \sum_{i=1}^n \|R(r'_{l,i})\|^2 \\ &\implies \sum_{i=1}^n \|e_i\|^2 = \sum_{i=1}^n \|r'_{r,i}\|^2 - 2s \sum_{i=1}^n r'_{r,i} \cdot R(r'_{l,i}) + s^2 \sum_{i=1}^n \|r'_{l,i}\|^2 \end{aligned}$$

$$\text{令: } \sum_{i=1}^n \|e_i\|^2 = S_r - 2sD + s^2S_l = (s\sqrt{S_l} - D/\sqrt{S_l})^2 + (S_lS_r - D^2)/S_l \quad (11)$$

$$\implies s = \left(\sum_{i=1}^n r'_{r,i} \cdot R(r'_{l,i}) \right) / \sum_{i=1}^n \|r'_{l,i}\|^2 \quad (12)$$

$$\text{根据对称性: } r_r = sR(r_l) + r_0 \quad r_l = \bar{s}R(r_r) + \bar{r}_0$$

$$\implies \bar{s} = 1/s \quad \bar{r}_0 = -\frac{1}{s}R^{-1}(r_0) \quad \bar{R} = R^{-1}$$

$$\text{可是: } \bar{s} = 1/s \neq \left(\sum_{i=1}^n r'_{l,i} \cdot \bar{R}(r'_{r,i}) \right) / \sum_{i=1}^n \|r'_{r,i}\|^2$$

$$\text{如果公式10变为: } e_i = \frac{1}{\sqrt{s}}r'_{r,i} - \sqrt{s}R(r'_{l,i})$$

则公式11变为:

$$\frac{1}{s}S_r - 2D + sS_r = \left(\sqrt{s}S_l - \frac{1}{\sqrt{s}}S_r \right)^2 + 2(S_lS_r - D)$$

$$\implies s = \left(\sum_{i=1}^n \|r'_{r,i}\|^2 / \sum_{i=1}^n \|r'_{l,i}\|^2 \right)^{1/2} \quad (13)$$

论文中提到：公式11变换后的式子中的D越大，剩余误差会越小，所以我们就必须找到旋转矩阵让式子

$$\sum_{i=1}^n \mathbf{r}'_{r,i} \cdot R(\mathbf{r}'_{l,i})$$

尽量越大越好。

$$\dot{\mathbf{r}}' = \dot{q}\dot{\mathbf{r}}\dot{q}^* = (\mathbf{Q}\dot{\mathbf{r}})\dot{q}^* = \bar{\mathbf{Q}}^T(\mathbf{Q}\dot{\mathbf{r}}) = (\bar{\mathbf{Q}}^T\mathbf{Q})\dot{\mathbf{r}} \text{ 也就是 } \mathbf{r}' = R\mathbf{r}^*, \text{ 所以D就可以表示成}$$

$$\sum_{i=1}^n (\dot{q}\dot{\mathbf{r}}'_{l,i}\dot{q}^*) \cdot \dot{\mathbf{r}}'_{r,i} = \sum_{i=1}^n (\dot{q}\dot{\mathbf{r}}'_{l,i}\dot{q}^*) \cdot (\dot{\mathbf{r}}'_{r,i}\dot{q}), \text{ 然后:}$$

$$\dot{q}\dot{\mathbf{r}}'_{l,i} = \begin{bmatrix} 0 & -x'_{l,i} & -y'_{l,i} & -z'_{l,i} \\ x'_{l,i} & 0 & z'_{l,i} & -y'_{l,i} \\ y'_{l,i} & -z'_{l,i} & 0 & x'_{l,i} \\ z'_{l,i} & y'_{l,i} & -x'_{l,i} & 0 \end{bmatrix} \dot{q} = \bar{\mathbb{R}}_{l,i} \dot{q}$$

$$\dot{\mathbf{r}}'_{r,i}\dot{q} = \begin{bmatrix} 0 & -x'_{r,i} & -y'_{r,i} & -z'_{r,i} \\ x'_{r,i} & 0 & -z'_{r,i} & y'_{r,i} \\ y'_{r,i} & z'_{r,i} & 0 & -x'_{r,i} \\ z'_{r,i} & -y'_{r,i} & x'_{r,i} & 0 \end{bmatrix} \dot{q} = \mathbb{R}_{r,i} \dot{q}.$$

$$\sum_{i=1}^n (\bar{\mathbb{R}}_{l,i} \dot{q}) \cdot (\mathbb{R}_{r,i} \dot{q})$$

$$\sum_{i=1}^n \dot{q}^T \bar{\mathbb{R}}_{l,i}^T \mathbb{R}_{r,i} \dot{q}$$

$$\dot{q}^T \left(\sum_{i=1}^n N_i \right) \dot{q} = \dot{q}^T N \dot{q},$$

$$M = \sum_{i=1}^n \mathbf{r}'_{l,i} \mathbf{r}'_{r,i}^T \quad M = \begin{bmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{bmatrix}$$

$$N = \begin{bmatrix} (S_{xx} + S_{yy} + S_{zz}) & S_{yz} - S_{zy} & S_{zx} - S_{xz} & S_{xy} - S_{yx} \\ S_{yz} - S_{zy} & (S_{xx} - S_{yy} - S_{zz}) & S_{xy} + S_{yx} & S_{zx} + S_{xz} \\ S_{zx} - S_{xz} & S_{xy} + S_{yx} & (-S_{xx} + S_{yy} - S_{zz}) & S_{yz} + S_{zy} \\ S_{xy} - S_{yx} & S_{zx} + S_{xz} & S_{yz} + S_{zy} & (-S_{xx} - S_{yy} + S_{zz}) \end{bmatrix}.$$

最后指出四元数的解就是对应的N的最大特征值的列向量。证明见论文中附录A3。

在程序的设计上也是先根据去心的匹配点对构造出了矩阵M，然后构造矩阵N，N矩阵最大特征值

(第一个特征值) 对应特征向量就是要求的四元数 (q0 q1 q2 q3)，将(q1 q2 q3)放入vec行向量，vec就是四元数旋转轴乘以sin(ang/2)，计算旋转的角度和旋转向量，再计算出旋转矩阵。再求出尺度（用的非对称模式）和平移量，最后给出位姿变换矩阵。

每次得到计算的sim3变换后，都能前几次变换成功后的mapoint集合起来得到更多的maoppoint。

Funtion2.5:matcher.SearchBySim3(mpCurrentKF,pKF, vpMapPointMatches,s,R,t,7.5)

查找更多的匹配（成功的闭环匹配需要满足足够多的匹配特征点数，之前使用SearchByBoW进行特征点匹配时会有漏匹配），通过Sim3变换，确定pKF1的特征点在pKF2中的大致区域，同理，确定pKF2的特征点在pKF1中的大致区域。在该区域内通过描述子进行匹配捕获pKF1和pKF2之前漏匹配的特征点，如果成功对上了，则认为是互相匹配上了。更新匹配vpMapPointMatches。

Funtion2.7:nInliers = Optimizer::OptimizeSim3(mpCurrentKF, pKF, vpMapPointMatches, gScm, 10, mbFixScale)

如果mbFixScale为true，则是6DoF优化（双目RGBD），如果是false，则是7DoF优化（单目），优化mpCurrentKF与pKF对应的MapPoints间的Sim3，得到优化后的量gScm。

如果优化后，通过卡方检验的内点数超过20的话，那么mpMatchedKF就是最终闭环检测出来与当前帧形成闭环的关键帧，得到从世界坐标系到该候选帧的Sim3变换，Scale=1，得到g2o优化后从世界坐标系到当前帧的Sim3变换。

只要有一个候选帧通过Sim3的求解与优化，就跳出停止对其它候选帧的判断。

如果没有一个闭环匹配候选帧通过Sim3的求解与优化，清空mvpEnoughConsistentCandidates。

取出闭环匹配上关键帧的相连关键帧，得到它们的MapPoints放入mvpLoopMapPoints，在执行上：将mpMatchedKF相连的关键帧全部取出来放入vpLoopConnectedKFs，将vpLoopConnectedKFs的MapPoints拿出来放入mvpLoopMapPoints。

Funtion2.8:matcher.SearchByProjection(mpCurrentKF, mScw, mvpLoopMapPoints, mvpCurrentMatchedPoints, 10)

将闭环匹配上关键帧以及相连关键帧的MapPoints投影到当前关键帧进行投影匹配。投影用的是mscw。根据投影查找更多的匹配（成功的闭环匹配需要满足足够多的匹配特征点数），根据Sim3变换，将每个mvpLoopMapPoints投影到mpCurrentKF上，并根据尺度确定一个搜索区域，根据该MapPoint的描述子与该区域内的特征点进行匹配，如果匹配误差小于TH_LOW即匹配成功，更新mvpCurrentMatchedPoints。mvpCurrentMatchedPoints将用于SearchAndFuse中检测当前帧MapPoints与匹配的MapPoints是否存在冲突。

判断当前帧与检测出的所有闭环关键帧是否有足够多的MapPoints匹配（40个），最后清空mvpEnoughConsistentCandidates。

Funtion3:CorrectLoop

- * 1. 通过求解的Sim3以及相对姿态关系，调整与当前帧相连的关键帧位姿以及这些关键帧观测到的MapPoints的位置（相连关键帧---当前帧）
- * 2. 将闭环帧以及闭环帧相连的关键帧的MapPoints和与当前帧相连的关键帧的点进行匹配（相连关键帧+当前帧---闭环帧+相连关键帧）
- * 3. 通过MapPoints的匹配关系更新这些帧之间的连接关系，即更新covisibility graph
- * 4. 对Essential Graph（Pose Graph）进行优化，MapPoints的位置则根据优化后的位姿做相对应的调整
- * 5. 创建线程进行全局Bundle Adjustment

Funtion3.1:mpLocalMapper->RequestStop()

请求局部地图停止，防止局部地图线程中InsertKeyFrame函数插入新的关键帧，如果全局优化线程还在运行，那么就终止它。

Funtion3.2:mpCurrentKF->UpdateConnections()

根据共视关系更新当前帧与其它关键帧之间的连接

通过位姿传播，得到Sim3优化后，与当前帧相连的关键帧的位姿，以及它们的MapPoints。 当前帧与世界坐标系之间的Sim变换在ComputeSim3函数中已经确定并优化，通过相对位姿关系，可以确定这些相连的关键帧与

世界坐标系之间的Sim3变换。取出与当前帧相连的关键帧，包括当前关键帧，先将mpCurrentKF的Sim3变换存入，固定不动。为了得到闭环g2o优化后各个关键帧的位姿，相连帧的位姿是从世界坐标系到相机坐标系的，当前帧的位姿（从世界坐标系到相机坐标系）固定不动，传播是世界到当前帧在传播到相连帧。然后不经过优化后的当前帧的话就直接得到世界坐标系到相连坐标系的位姿。

得到调整相连帧位姿后，修正这些关键帧的MapPoints。将该未校正的eigP3Dw先从世界坐标系映射到未校正的pKF相机坐标系（ $\times g2oSwi$ ），然后再反射到校正后的世界坐标系下（ $\times g2oCorrectedSwi$ ）。

将Sim3转换为SE3，根据更新的Sim3，更新关键帧的位姿。把sim3的尺度信息去掉就是SE3位姿。

根据共视关系更新当前帧与其它关键帧之间的连接。这时候的连接已经包括当前帧和闭环帧直接的连接。

检查当前帧的MapPoints与闭环匹配帧的MapPoints是否存在冲突，对冲突的MapPoints进行替换或填补。如果有重复的MapPoint（当前帧和匹配帧各有一个），则用匹配帧的代替现有的，如果当前帧没有该MapPoint，则直接添加。

Funtion3.3: SearchAndFuse(CorrectedSim3)

通过将闭环时相连关键帧的mvpLoopMapPoints投影到这些关键帧中，进行MapPoints检查与替换。

更新当前关键帧之间的共视相连关系，得到因闭环时MapPoints融合而新得到的连接关系。首先，遍历当前帧相连关键帧（一级相连），得到与当前帧相连关键帧的相连关键帧（二级相连），更新一级相连关键帧的连接关系（这个是在修正MapPoint之后得到的连接关系）。取出该帧更新后的连接关系，从连接关系中去除闭环之前的二级连接关系，剩下的连接就是由闭环得到的连接关系，从连接关系中去除闭环之前的一级连接关系，剩下的连接就是由闭环得到的连接关系。

Funtion3.4:OptimizeEssentialGraph(mpMap, mpMatchedKF, mpCurrentKF, NonCorrectedSim3, CorrectedSim3, LoopConnections, mbFixScale)

进行EssentialGraph优化，LoopConnections是形成闭环后新生成的连接关系，不包括步骤7中当前帧与闭环匹配帧在MapPoints融合之后而新得到的连接关系。1.将地图中所有keyframe的pose作为顶点添加到优化器（尽可能使用经过Sim3调整的位姿）。2.1、LoopConnections是闭环时因为MapPoints调整而出现的新关键帧连接关系；2.2、添加跟踪时形成的边、闭环匹配成功形成的边（1.只添加扩展树的边（有父关键帧）2.添加在CorrectLoop函数中AddLoopEdge函数添加的闭环连接边）;2.3、最有很好共视关系的关键帧也作为边进行优化。最后一次取得优化后的各个位姿和空间点。

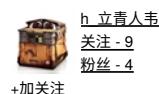
Funtion3.5:mpMatchedKF->AddLoopEdge(mpCurrentKF); mpCurrentKF->AddLoopEdge(mpMatchedKF)

添加当前帧与闭环匹配帧之间的边，这两句话应该放在OptimizeEssentialGraph之前，因为OptimizeEssentialGraph的步骤2.2.2(青色部分)中有优化。

Funtion3.6:new thread(&LoopClosing::RunGlobalBundleAdjustment,this,mpCurrentKF->mnId)

这是整个程序真正意义上在全局地图的后端优化。优化后要更新所有的地图点和关键帧的位姿。因为地图管理线程（localmapping）在全局BA优化的过程中仍然在工作，这就有可能导致新的关键帧没有被优化，所以我们需要通过spanning tree来进行调整（就是调整没有被优化的新进来的点和关键帧）。

mvpKeyFrameOrigins放的是原始关键帧（在初始化的时候插入的关键帧），然后往下找其子节点，如果字节点没有经过BA优化，根据子节点的位姿算出父节点的从世界坐标系到相机坐标系的位姿，如果优化后就直接压入列表。然后把子节点送入列表，删除父节点，再取出列表中的刚刚插入的子节点作为新的父节点，这样反复下去。对于地图点，如果经过BA优化就保持原来的，如果没有经过优化就利用生成它的关键帧的位姿修正为成世界坐标系里的点。



0 0

« 上一篇: [orb_slam代码解析\(3\)LocalMapping线程](#)

» 下一篇: [ubuntu14.04+ros_indigo+lsdslam](#)

posted @ 2018-04-05 19:14 h_立青人韦 阅读(352) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请[登录](#)或[注册](#)，访问网站首页。

【推荐】超50万VC++源码: 大型组态工控、电力仿真CAD与GIS源码库!

【推荐】基于HTML5的 WebGL 楼宇自控 3D 可视化监控

【推荐】专业便捷的企业级代码托管服务 - Gitee 码云

相关博文：

- 重读ORB_SLAM之LoopClosing线程难点
- 重读ORB_SLAM之LocalMapping线程难点
- orb_slam代码解析(2)Tracking线程
- ORB_SLAM2 源码阅读 ORB_SLAM2::ORBextractor
- 视觉SLAM算法框架解析(2) ORB-SLAM

最新新闻：

- 对话顾剑民博士：自动驾驶热度“滑向低谷”，寒冬未真正到来
 - 彭蕾：CEO如何面对“至暗时刻”和无可诉说的孤独感
 - BuzzFeed衰落启示：无限竞争+劣质产品=商业模式失败
 - 阿里没有“失速”
 - FBI抓捕第二位苹果中国工程师 指控窃取无人机机密面临10年监禁
- » [更多新闻...](#)

Copyright ©2019 h_立青人韦