**2019/1/25**

**Tracking类**

1.Tracking(System \*pSys, ORBVocabulary\* pVoc, FrameDrawer \*pFrameDrawer, MapDrawer \*pMapDrawer, Map \*pMap, KeyFrameDatabase\* pKFDB, const string &strSettingPath, const int sensor)

构造函数:从参数文件读取参数,构造特征提取器;

2. GrabImageRGBD(const cv::Mat &imRGB,const cv::Mat &imD, const double &timestamp) 得到当前帧的位姿

1)将RGB图转成灰度图,深度图转化imDepth

2)构造Frame

3)Track();

3. Track()

1. StereoInitialization()

若当前帧特征点>500 :

初始化当前帧位姿；

将**当前帧**设为**初始关键帧**；并添加到Map

构造MapPoint (当前帧)(a. 反投影得到3D坐标，并构造成 MapPoint; b.为 MapPoint添加属性；添加该MapPoint到地图；当前帧与 MapPoint 的 连接,初始关 键帧与MapPoint的连接；

往LocalMapping输送该初始关键帧(Tracking--->LocalMapping)

局部关键帧中加入初始帧信息；mvpLocalKeyFrames干嘛用,

局部地图点中加入地图点；mvpLocalMapPoints干嘛用?传入Map画图?

将初始帧设为Tracking\当前帧的参考帧mpReferenceKF；

1. MonocularInitailization()

若没有初始化，且第一帧特征点数>100,构建单目初始器；

已经初始化了，匹配当前帧与初始帧之间的特征点（SearchForInitailization()）；如果匹配成功的点数太少，则删除单目初始器，重新初始化；开启两个线程H和F，得到帧间相对运动和初始MapPoints；逐个删除那些无法进行三角化的匹配点；将初始化的第一帧作为世界坐标系；CreateInitialMapMonocular()将三角化得到的3D点包装成MapPoints存入KeyFrame和Map;

1. TrackReferenceKeyFrame() 将最近的参考关键帧作为当前位姿的初始值,进行优化,剔除外点(在优化过程中已确认的外点),最终匹配数>10, 跟踪有效!!!

将当前描述子转换为BoW向量；

词包匹配,通过当前帧与Tracking参考帧的词包匹配,更新得到当前帧的MapPoint；

匹配对>15，将上一帧的位姿作为当前位姿的初始值；

通过3D-2D的重投影误差来获得位姿；剔除优化后的outlier匹配点；

1. UpdateLastFrame() mlRelativeFramePoses??? 临时增加上一帧MapPoint

对于双目或rgbd相机，根据深度值会为上一关键帧生成新的MapPoints.在跟踪过程中，会不断剔除Outlier,如果不及时增加MapPoints会逐渐减少；这个函数就是补充增加RGBD和双目相机上一帧的MapPoints个数。

1. TrackWithMotionModel() 估计, PnP优化, 检测(>10,>20)

更新上一帧的MapPoints(临时增加)；根据恒速模型计算当前帧的位姿；投影,对上一帧的MapPoints进行跟踪，更新当前帧的MapPoint；优化位姿；剔除Outliers;

1. NeedNewKeyFrame()

重定位则不插入关键帧；局部地图被闭环检测使用，不插入关键帧；判断是否距离上一次插入关键帧时间太短；

得到参考关键帧跟踪到的MapPoints数量；查询局部地图管理器是否繁忙；（双目或RGBD）统计总的可以添加的MapPoints数量和跟踪到地图中的MapPoint数量；决策是否需要插入关键帧(条件:时间间隔,localMapping空闲状态,跟踪到点不够多,)跟踪太差了,而localmappingBA又在进行,则打断;

1. CreateNewKeyFrame()

将**当前帧**构造成关键帧；将当前帧设置为Tracking ,当前帧的参考关键帧；

(双目或RGBD)为当前帧生成新的MapPoint :（1）计算mRcw,mtcw和mRwc、mOw；（2）得到当前帧深度小于阈值的特征点；（3）按照深度从小到大排列，将距离比较近的点包装成MapPoints或者将不合格的点重新生成MapPoint； 将关键帧插入localmapping

1. SearchLocalPoints() /\*\*\*在局部地图中查找在当前帧视野范围内的点，将视野范围内的点和当前帧的特征点进行投影匹配\*\*\*/

遍历当前帧的mvpMapPoints，标记这些MapPoints不参与之后的搜索；将所有局部MapPoints投影到当前帧，判断LocalMapPoint中的点是否在视野内；对视野范围内的MapPoints通过投影进行特征点匹配。

8. TrackLocalMap()

\* @brief 对Local Map的MapPoints进行跟踪

\* 1. 更新局部地图，包括局部关键帧和关键点

\* 2. 对局部MapPoints进行投影匹配,增加当前帧与局部地图点的匹配

\* 3. 根据匹配对(3d-2d pnp)估计当前帧的姿态

\* 4. 根据姿态剔除误匹配

1. UpdateLocalPoints() 清空原先的局部关键点,将局部关键帧的MapPoint都加到局部关键点.

清空mvpLocalMapPoints;遍历局部关键帧mvpLocalKeyFrames,将局部关键帧的MapPoints添加到mvpLocalMapPoints中；

1. UpdateLocalKeyFrames() 根据当前帧的MapPoint更新局部关键帧,与2个参考关键帧

遍历当前帧的MapPoints,记录所有能观测到当前帧MoapPoints的关键帧；策略(1)将这些关键帧存到mvpLocalKeyFrames.策略(2)分别将这些关键帧的最佳共视帧，子关键帧，父关键帧存入；更新当前帧的参考关键帧mCurrentFrame.mpReferenceKF，与自己共视程度最高的关键帧作为参考关键帧

1. Relocalization()

计算当前帧特征点的Bow映射；通过词带,找到与当前帧相似的候选关键帧；

**对每一候选关键帧筛选:**

通过Bow进行匹配，得到匹配后的当前帧对应的MapPoint; 筛选(匹配对<15)

初始化 PnPsolver；

**对筛选后的每一候选关键帧:**

粗估计: EpnP RANSAC迭代求解位姿;

精估计: 通过之前的匹配获得当前帧的3D点,对当前帧姿态进行PnP优化；如果内点太少(30<n<50)，则通过投影的方式对之前未匹配的点进行匹配，再进行PnP优化,剔除外点；

只要有一帧成功了就则退出循环

**2019/1/26**

**Frame类**

1. Frame() 复制构造函数：将某一帧复制
2. Frame()初始化（构造函数） //图像分成48x64个格子

提取ORB特征点；矫正orb提取的特征点；获取深度d；矫正灰度图的边界并方格化;分配特征点到网格里mGrid；

1. AssignFeaturesToGrid()

将特征点分配到网格（64\*48），是为了加速特征匹配；mGrid[i][j]每一个元素都是一个向量vector.向量里面存的是特征点编号.

1. ExtractORB()
2. SetPose()

根据Tcw计算R，t;

1. PosInGrid()

将特征点像素坐标转换成方格坐标并判断是否在方格内；

1. isInFrustum(MapPoint \*pMP, float viewingCosLimit)

1)将MapPoint点投影到该帧，判断是否在方格子内

2)计算MapPoint点到相机中心的深度，判断是否在MapPoint点的距离范围内（这个距离范围如何得到）？？

判断准则：0. 其深度值是否>0;1. 取出MapPoint点投影到像素坐标，判断有无越界；2. MapPoint点到相机中心的距离，判断是否在合理的距离范围；3. 计算视角，是否<60度（MapPoint点的平均观测方向计算，viewingCosLimit视角和平均视角的方向阈值60是经验值吗？）；4. 根据深度预测尺度？？ 满足5个条件，则标记该点将来要被投影。

1. GetFeaturesInArea() 得到（x,y）附近区域内的方格子内的特帧点个数和索引

1)判断要求区域是否在图像内

2)判断该区域的方格子内的各个特征点是否满足要求。如果满足，则压入vIndices

8.PosInGrid(const cv::KeyPoint &kp, int &posX, int &posY)

9. ComputeBow()

计算词包mBowVec 和mFeatVec????

10.UndistortKeyPoints()

调用OpenCV的矫正函数矫正orb提取的特征点，存入mvKeysUn.

11.ComputeImageBounds(const cv::Mat &imLeft)

矫正图像畸变，确定新的图像边界

12.ComputeStereoMatches()!!! 双目匹配

1)建立每一行的特征点搜索带状区域vRowIndices

2)对左目的每一个特征点，通过描述子以及滑窗在右目带状搜索区域找到匹配点；

2.1)粗估计：遍历右目相应行的搜索带上的特征点vCandidates，若非近邻层点（删除）。比较描述子，描述子距离最小的右目特征点索引为bestIdxR;

2.2)精确估计：左右目特征点窗口的光度比较；在右目特征点附近区域进行滑窗SAD比较，并通过抛物线拟合出亚像素坐标；

2.3)求深度depth

3)vDistIdx中剔除光度偏差较大的匹配点对，删除对应的深度和右目上的匹配点索引；

13.ComputeStereoFromRGBD(const cv::Mat &imDepth)

14.UnprojectStereo(const int &i)

**2019/1/27**

**MapPoint类**

1. 构造函数：
2. 给定 点坐标和关键帧构造MapPoint. ID,Pos,mNormalVector(平均观测方向)；
3. 给定 点坐标和frame构造MapPoint.

Id，初始化平均观测方向（1,1,1）；点的距离dist，距离范围mfMaxDistance（用于计算ＭapPoint在frame中所在的层数），点在该帧的描述子，；

1. SetWorldPos(const cv::Mat &Pos)

赋予MapPoint点世界坐标；

1. GetWorldPos()

返回ＭapPoint点的世界坐标；

1. GetNormal()

返回MapPoint平均观测方向；

1. GetReferenceKeyFrame() 参考关键帧是什么概念？何用？

返回关键参考帧

1. AddObservation() map<KeyFrame\*,size\_t> mObservations

记录那些KeyFrame的哪个特征点对应到该MapPoint,并添加观测的相机数（帧数）(数据结构：mObservations) (观测到该点的相机数nObs++).

1. EraseObservation(KeyFrame\* pKF)

当该点不出现在该帧时，则删除他们之间的关系（nOb - -）;

删除之后，如果观测到该ＭapPoint点的相机数<2，则丢弃该点，则分别删除mObservation,关键帧对应的点和地图上的点；

1. GetObservation()

返回观测（mObservations）

1. Observations()

返回观测到该点的观测相机数（帧数）；

1. SetBadFlag()

分别删除mObservation,关键帧对应的点和地图上的点；

1. GetReplaced()

选择被替换？？？？

1. Replace() 闭环时候，满足什么条件替换？

将当前地图点（this）,替换成pMp.在形成闭环的时候，会更新KeyFrame与MapPoint之间的关系；

1. isBad() 什么情况下被认为是坏点

1)被替换掉了 2)。。。

没有经过MapPointCulling检测的MapPoints;？？？

1. IncreaseVisible() //视野范围内 Visible

mnVisible+=n;

1. IncreaseFound()

能找到该点的帧数+=n;

1. GetFoundRatio()

返回mnFound/mnVisible；

1. ComputeDistinctiveDescriptors()

更新当前点的最适合的描述子；

1. GetDescriptor()

返回该点的描述子；

1. GetIndexInKeyFrame(KeyFrame \*pKF)

得到该点在该关键帧的索引；

1. IsInKeyFrame(KeyFrame \*pKF)

判断该点是否有出现在该关键帧，即看是否该关键帧上有连接；

1. UpdateNormalAndDepth()

更新该点的平均观测方向以及观测距离范围，由于一个MapPoint会被许多相机观测到，因此在插入关键帧后，需要更新响应变量；

1. GetMinDistanceInvariance()为什么还要乘0.8或1.2 , GetMaxDistanceInvariance()
2. PredictScale(const float &currentDist, Frame\* pF)

根据dist计算ＭapPoint在pKF所在的层数

1. PredictScale(const float &currentDist, KeyFrame\* pKF)

**2019/1/31**

**Map类**

1. mnMaxKFid();

2. AddKeyFrame();

添加关键帧（往mspKeyFrames中insert）

3. AddMapPoint();

添加MapPoint.(往mspMapPoint中insert)

4. EraseMapPoint()

5. EraseKeyFrame()

6. SetReferenceMapPoints()

mvpReferenceMapPoints是什么？？？将我们提取出的mvpLocalMapPoints赋值给mvpReferenceMapPoints

7. GetAllKeyFrame()

8. GetAllMapPoints()

9. MapPointsInMap()返回地图中MapPoint的个数

10. KeyFramesInMap()返回地图中关键帧的个数

11. GetReferenceMapPoints()

12. GetMaxKFid()

13. clear()

//清空地图。包括MapPoint,KeyFrame,重置mnMaxKFid,参考点，关键帧origins

**FrameDrawer类**

将Tracking线程里的数据拷贝到绘图线程，并根据mstate进行绘图

1.FrameDrawer（Map\* pMap):mpMap(pMap)

构造函数，初始化画布

2. DrawFrame()

准备需要显示的信息，包括图像，特征点，地图，跟踪状态等

将成员变量赋值给局部变量（包括图像、状态、其它的提示）深拷贝im图像；1.未初始化状态，vIniKeys,vMatches; 跟踪正常状态，vbVO,vbMap；失踪状态，vCurrentKeys; 2.绘制初始化轨迹连线，绘制特征点边框。

初始化状态，则将当前帧的特征点与初始帧的特帧点连成线，形成轨迹；跟踪正常状态，绘制圆圈和方框；显示信息DrawTextInfo()；

3. DrawTextInfo()

4. Update()

将跟踪线程（Tracking）的数据拷贝到绘图线程（图像、特征点、地图、跟踪状态）

**2019/2/1**

**MapDrawer类**

1. MapDrawer()

读取各种参数fSetings

2. DrawMapPoints()

取出所有地图点，以及局部地图点；并画出地图点（黑色），局部地图点（红色）；

3. DrawKeyFrames（）

取出所有关键帧（蓝色）；显示关键帧共识关系（连接线 ：绿色），显示与其父关键帧的连接线，显示该帧与相关关键帧的闭环连接线；

4. DrawCurrentCamera（）

画出当前相机的模型；

5. SetCurrentCameraPose（）

返回相机位姿

6. GetCurrentOpenGLCameraMatrix（）

将相机位姿mCameraPose由Mat类型转化为OpenGlMatrix类型

**KeyFrameDatabase类**

1. KeyFrameDatabase（）

构造函数. MnInvertedFile！！！！std::vector<list<KeyFrame\*> >

2. add（KeyFrame \*pKF）

根据关加进来的关键帧的词包，更新数据库（mvInvertedFile）的倒排索引

3. erase(KeyFrame\* pKF)

键帧被删除后，更新数据库（mvIncertedFile）的倒排索引

4. clear()

删除mvINvertedFile

5. DetectLoopCandidates(KeyFrame\* pKF, float minScore)

在闭环检测中找到与该关键帧可能闭环的关键帧

\* 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）

\* 2. 只和具有共同单词较多的关键帧进行相似度计算

\* 3. 将与关键帧相连（权值最高）的前十个关键帧归为一组，计算累计得分

\* 4. 只返回累计得分较高的组中分数最高的关键帧

6. DetectRelocalizationCandidates(Frame \*F)

\* @brief 在重定位中找到与该帧相似的关键帧

\* 1. 找出和当前帧具有公共单词的所有关键帧

\* 2. 只和具有共同单词较多的关键帧进行相似度计算

\* 3. 将与关键帧相连（权值最高）的前十个关键帧归为一组，计算累计得分

\* 4. 只返回累计得分较高的组中分数最高的关键帧

**Initializer类**

1. 构造函数。内参矩阵K，测量误差sigma，迭代次数iterations等

2. Initialize()

**Optimizer类**

1. GlobalBundleAdjustemnt(Map\* pMap, int nIterations, bool\* pbStopFlag, const unsigned long nLoopKF, const bool bRobust)

得到地图pMap上所有关键帧和地图点；然后做BA

2. BundleAdjustment(const vector<KeyFrame \*> &vpKFs, const vector<MapPoint \*> &vpMP,int nIterations, bool\* pbStopFlag, const unsigned long nLoopKF, const bool bRobust)

**2019/2/19**

**LocalMapping类**

1.LocalMapping (Map \*pMap, const float bMonocular)

构造函数

2.SetLoopCloser(LoopClosing\* pLoopCloser)

3.SetTracker(Tracking \*pTracker)

4.run()

1.ProcessNewKeyFrame()

处理列表中的关键帧，得到该帧中生成的ＭapPoint，待检查。

1)从mlNewKeyFrames中取出新关键帧（取出即删除），并计算特征点的Bow映射关系；

2)得到该帧的对应的MapPoint；遍历MapPoint，（KF与pMP有连接，但pMP与ＫＦ没连接，故要连接（为什么会导致这样？？？）如果不是该帧生成的(说明是好点???不需要检测????)，则更新属性(添加观测，获得该点的平均观测方向和观测距离范围，更新3d点的最佳描述子)，否则，放入mlpRecentAddedMapPoints,接受MapPointCulling函数的检验；

3)更新关键帧间的连接关系（因为什么变了，所以要更新？？）mConnectedKeyFrameWeights，mvpOrderedConnectedKeyFrames，mvOrderedWeights

4)将该关键帧插入地图中pMap->AddKeyFrame(mpCurrentKeyFrame);

2. MapPointCulling()

剔除ProcessNewKeyFrame和CreateNewMapPoints函数中引入的质量不好的MapPoints,留下稳定的质量高的点；

遍历待检查的MapPoints，

1) 该点是坏点，删除（即放弃对该店的检测）

2)该点GetFoundRatio<0.25，令其坏点，并删除（放弃对该店的检测）；

3)该点超过2帧并且观测次数不超过2帧，令其坏点，并删除（放弃对该店的检测）；说明该点不稳定。

4) 该点超过3帧了，说明是好点，删除（即放弃对该店的检测，是好点没必要测）//要不要再加一个条件？且观测次数>=3；

5) 其他点继续留着，要测

3. CreateNewMapPoints()

相机运动过程中和共视程度比较高的关键帧通过三角化恢复出一些MapPoints

1)在当前关键帧的共识关键帧中找到共识程度最高的nn帧相邻帧vpNeighKFs；\* 权值是否要大于15？？？

2)遍历当前帧与相邻关键帧vpNeighKFs，若相机运动基线短，则取消；

3)计算两个关键帧的基本矩阵，通过极线约束限制匹配时的搜索范围，进行特征点匹配，得到vMatchedIndices；

4)取出匹配点计算视差角，并三角化恢复3D点坐标；

5)检测该点在两个关键帧下的重投影误差；若符合，则构造MapPoint，并添加属性，存入mlpRecentAddedMapPoints中接受MapPointCulling检验；

3. SearchInNeighbors() 检查并融合当前关键帧与相邻帧（两级相邻）重复的MapPoints

1)获得当前关键帧共识程度最高的前nn关键帧，遍历这些帧及各”相应“的5帧，都存入vpTargetKFs中；

2)将当前帧的MapPoint和vpTargetKFs中的点进行融合；

3)将vpTargetKFs中的点与当前帧的点融合；

4)更新当前帧MapPoints的描述子，深度，观测主方向等属性，并更新与其他帧的连接关系；

4. ComputeF12(KeyFrame \*&pKF1, KeyFrame \*&pKF2)

根据两关键帧的姿态计算两个关键帧之间的基础矩阵F

5. KeyFrameCulling() 删除当前帧的局部关键帧当中的冗余帧

1)提取当前帧的所有连接关键帧（共识程度>15?？？），作为局部关键帧；

2)遍历所有局部关键帧，提取每个帧的MapPoint并遍历;

3)MapPoint点是否在近处，判断局部关键帧中是否90%以上的点能被其他同尺度的关键帧观测到；同尺度是什么鬼？

4)删除冗余关键帧

6. ….......

**2019/2/19**

**LoopClosing类**

1. LoopClosing（）

构造函数，回环检测参数确定；

2. SetTracker(Tracking \*pTracker)

3. SetLocalMapper(LocalMapping \*pLocalMapper)

4. Run() 总的流程在这里

在LocalMapping中通过InsertKeyFrame将关键帧插入闭环检测队列mlpLoopKeyFrameQueue;查看列表中是否有等待被插入的关键帧；检测回环候选帧共识关系；计算当前帧与闭环帧的Sim3变换；闭环操作以及全局BA；设置闭环完成标志；

5. InsertKeyFrame(KeyFrame \*pKF)

将关键帧插入mlpLoopKeyFrameQueue

6. CheckNewKeyFrames()

检查是否有被等待的关键帧；

7. DetectLoop()

从mlpLoopKeyFrameQueue中取出关键帧，如果距离上次闭环没多久（小于0帧），不进行闭环操作；遍历该帧的所有共识关键帧，计算当前帧与共识关键帧的得分，得到最低得分minscore(最低得分作为阈值)；在所有关键帧（关键帧库）中找到与当前帧的一组闭环候选帧vpCandidateKFs；如果没有，则往mpKeyFrameDB添加当前帧；

遍历当前帧的一组闭环候选帧，得到每个闭环候选帧的相连的关键帧，与自己构成一个个”子候选组“spCandidateGruop；遍历之前的”子连续组“，看是否与子候选组有共同的关键帧，若有则计数器+1,并入“子连续组”，该候选帧成为合格的闭环候选帧。

8. ComputeSim3() 目的:得到最后的闭环帧及mvpCurrentMatchedPoints

从筛选合格的闭环候选帧中取出一帧关键帧，计算它与当前帧的匹配关系，如果匹配点对少，则剔除；否则匹配成功，构造sim3求解器；

遍历所有闭环候选帧：

粗估计：求解其与当前帧的sim3,得到当前帧到候选关键帧的位姿。

精确估计：利用位姿匹配两帧（弥补漏匹配），并更新两帧之间的MapPoint（为后边g2o准备）;进行g2o优化位姿,得到候选帧到当前帧的位姿gScm以及匹配点；若优化后匹配内点书大于20,则终于得到世界坐标系到当前帧的Sim3变换。以及优化后的MapPoint点。跳出整个循环（就认定该候选帧mpMatchedKF，mvpCurrentMatchedPoints）

注:仅当前帧----闭环帧间的MapPOint匹配 >20

检验：

取出闭环帧的相连关键帧，得到他们的MapPoint放入mvpLoopMapPoints，标记该MapPoint在闭环时被当前帧观测到。

寻找更多匹配点检验闭环：将mvpLoopMapPoints中的点全部投影到当前关键帧，查找更多匹配点，匹配成功的点存入mvpCurrentMatchedPoints。如果匹配成功的点>=40,则闭环成功；否则闭环失败，删除当前帧和所有闭环候选帧；

注:更大范围的检验, 当前帧及相连帧-------闭环帧及相连帧 的MapPoint匹配 >40

9. CorrectLoop() 真正的闭环操作

1）局部地图要求停止（防止新的关键帧插入）；

全局BA在运行的话，停止；

2)更新并修正当前帧与其他关键帧的连接关系；

2.1）当前帧在世界坐标系的Sim3变换已知(mg2oScw)，计算与当前帧相连的各个关键帧pKFi的位姿(只计算,没操作), 保留未调整前的位姿;

2.2) 修正各个pKFi关键帧的MapPoint位姿(操作)；

2.3) 修正各个pKFi 关键帧的位姿(操作)；

2.4) 更新pKFi 与其他关键帧的连接关系

3）闭环修正后，当前帧的MapPoint与闭环帧的MapPoint 要融合起来；

4）各个相连关键帧pKFi上的MapPoint点也要和闭环帧的MapPoint 融合

5）得到闭环关系（LoopConnections）用于优化!!!!!纽带作用?!!!!

5.1) 删除闭环之前当前帧的二级连接关系，剩下的就是由闭环得到的连接关系（LoopConnections）

5.2) 删除闭环之前当前帧的一级连接关系，剩下的就是由闭环得到的连接关系

6）Essential Graph优化，只优化关键帧，MapPoint根据优化后的位姿做相对应的调整。

7） 全局优化Full BA

**2019/2/14**

**Stereo\_kitti.cpp**

先将配对好的双目图像（string）和对应时间戳存在容器中(LoadImages)；构造SLAM系统；

遍历图片序列，开始slam track操作，计算用时；将每一帧的用时排序；存储CameraTrajectory.txt;

**System类**

1. System（） 构造函数

输出welcome message;将配置文件读入到内存中，存储到fsSettings变量中

;将字典voc加载到mpVocabulary中;

创建关键帧数据库，地图，帧绘画，地图绘画；初始化Track,LocalMapping,LoopClosing,Viewer线程；线程嵌套？？？

2. TrackStereo()

check mode change;check reset;

GrabImageStereo()!!!!!!入口！重要！

2. TrackMonocular（）

2.TrackRGBD（）

3. TrackRGBD（）

4.TrackMonocular（）

5.ActivateLocalizationMode()

6.DeactivateLocalizationMode()

7.Reset()

8.Shutdown()

9.SaveTrajectoryTUM()

从地图Map中获取所有关键帧，并按Id排序；一般的帧位姿是相对于参考关键帧，父关键帧等推算得出；

10.SaveKeyFrameTrajectoryTUM()

从地图Map中获取所有关键帧，并按Id排序；直接输出；

11.SaveTrajectoryKITTI()与saveTrajjectoryTUM()类似

从地图Map中获取所有关键帧，并按Id排序；一般的帧位姿是相对于参考关键帧，父关键帧等推算得出；

**2019/2/16**

**KeyFrame类**

1.KeyFrame();

构造函数：Id,mGrid大小,mGrid每个格子的特征点数，设置位姿；

2.computeBow()

计算单词；

3.SetPose()

设置位姿??????????

左目相机与立体相机中心坐标。

4.GetPose()

5.GetPoseInverse()

6.GetCameraCenter()

7.GetStereoCenter()

8.GetRotation()

9.GetTranslation()

10.AddConnection(KeyFrame \*pKF, const int &weight)

一个关键帧靠mConnectedKeyFrameWeights记录与其他关键帧的权重；

并通过mvpOrderedConnectedKeyFrames，mvOrderedWeights排序。

11.UpdateBestCovisibles()

1)相连关键帧根据权重排序（大-->小）

2)将排序结果存储在mvpOrderedConnectedKeyFrames和mvOrderedWeights中

**注：在UpdateConnections()中mvpOrderedConnectedKeyFrames存的是共识点数>15的关键帧。在这里为什么不需要>15。**

12.GetConnectedKeyFrames()

得到与该关键帧连接的所有关键帧（未按权值排序）

13.GetVectorCovisibleKeyFrames() 返回与该关键帧连接的关键帧（已按权值排序）

返回mvpOrderedConnectedKeyFrames。

14.GetBestCovisibilityKeyFrames(const int &N)

返回前N个权值大的关键帧(已按权值排序)

15.GetCovisiblesByWeight(const int &w)

返回与该关键帧连接的权重大于等于w的关键帧（已按权值排序）

16.GetWeight(KeyFrame \*pKF)

得到该关键帧与pKF的权重

17.AddMapPoint(MapPoint \*pMP, const size\_t &idx)

根据索引添加，为该关键帧添加地图点（索引即是关键帧中的关键点序号）

18.EraseMapPointMatch(const size\_t &idx)

根据索引删除关键帧里对应的MapPoint

19.ReplaceMapPointMatch(const size\_t &idx, MapPoint\* pMP)

什么时候替代？为何森马替代？

20.GetMapPoints()

返回该关键帧所有的地图点MapPoint

21.TrackedMapPoints(const int &minObs)

返回该关键帧中高质量MapPoint点的个数(被至少minObs个关键帧观测到)

22.GetMapPointMatches() 返回该关键帧的MapPoints

23.GetMapPoint(const size\_t &idx) 根据索引返回该关键帧的MapPoint点

24.UpdateConnections() 更新图的连接(即更新3个数据结构mConnectedKeyFrameWeights，mvpOrderedConnectedKeyFrames，mvOrderedWeights)

1)获得该关键帧的所有MapPoint点

2)通过MapPoint点获取与当前关键帧的有共识点的关键帧（权重=共识点数）

3)与当前关键帧共识点数>15的关键帧存入vPairs(共识程度高的关键帧),并更新该关键帧的mConnectedKeyFrameWeights；若没有超过15,则取权重最大的关键帧存入vPairs

4)将vPairs按权值排序。更新共识图连接（即更新3个数据结构）；

5)若没有初始化，则初始化共识程度最高的关键帧为父关键帧；

25.AddChild(KeyFrame \*pKF)

往子关键帧集合里添加pKF;

26.EraseChild(KeyFrame \*pKF)

27.ChangeParent(KeyFrame \*pKF)

28.GetChilds()

29.GetParent()

30.hasChild(KeyFrame \*pKF)

31.AddLoopEdge(KeyFrame \*pKF)

往mspLoopEdges中加入pKF;

32.GetLoopEdges() 怎样才算回环关键帧；

返回mspLoopEdges；

33.SetNotErase()

34.SetErase()

35.SetBadFlag()

1)让别人删除与自己的联系

2)让自己删除与别人的联系

3)子关键帧重新选择父关键帧；

4)计算与父关键帧之间的位姿；

5)从地图中删除，从关键帧库中删除

36.isBad()

37.EraseConnection(KeyFrame\* pKF)

38.GetFeaturesInArea(const float &x, const float &y, const float &r)

39.IsInImage(const float &x, const float &y) const

40.UnprojectStereo(int i)

41.ComputeSceneMedianDepth(const int q)