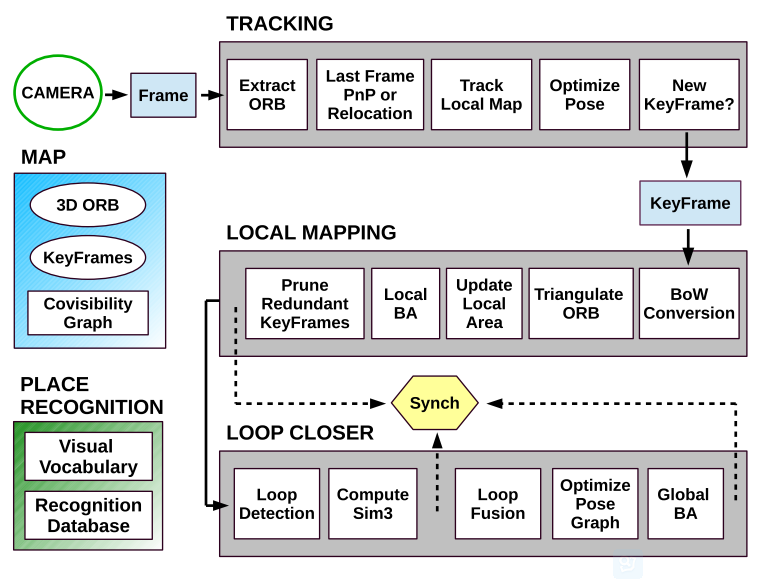
KeyFrame.cpp



功能描述：

创建关键帧；计算Bow，统计单词；更新关键帧共视图，生成树的连接；处理“坏”关键帧；估计当前关键帧场景深度（单目）；回环检测中寻找可能闭环的关键帧或重定位中寻找最相似的关键帧；

成员变量：

static long unsigned int nNextId; //上一关键帧的id

long unsigned int mnId; //当前关键帧id

const long unsigned int mnFrameId; //当前关键帧由哪个Frame初始化的

long unsigned int mnLoopQuery; //可能和当前关键帧形成闭环的关键帧id

int mnLoopWords; //和mnLoopQuery拥有相同words的数目

float mLoopScore; //和mnLoopQuery的相似度得分

long unsigned int mnRelocQuery; //重定位中和当前关键帧最为相似的关键帧id

int mnRelocWords; //和mnRelocQuery拥有相同words的数目

float mRelocScore; //和mnRelocQuery的相似度得分

DBoW2::BowVector mBowVec; //当前关键帧的单词

DBoW2::FeatureVector mFeatVec; //当前关键帧特征点在词典的分布

cv::Mat mTcp; //当前关键帧相对父帧（生成树中）的位资，父帧->当前帧

cv::Mat Tcw; //当前帧的位姿，世界->当前帧

cv::Mat Twc; //当前帧的位姿，当前帧->世界

cv::Mat Ow; //当前帧相机光心在世界坐标下的位置

cv::Mat Cw; //双目中心（左右两相机光心连线中点）在世界坐标系下的位置

std::map<KeyFrame\*,int> mConnectedKeyFrameWeights; //当前帧的共视关键帧及对应权重

std::vector<KeyFrame\*> mvpOrderedConnectedKeyFrames; //按权重由大到小排序的共视关键帧

std::vector<int> mvOrderedWeights; //由大到小排序的权重

KeyFrame\* mpParent; //生成树中当前帧的父帧

std::set<KeyFrame\*> mspChildrens; //生成树中当前帧的子帧

bool mbBad; //当前关键帧是否为坏帧

成员函数：

void SetPose(const cv::Mat &Tcw); //设置当前帧的位姿

cv::Mat GetPose(); //获得当前帧的位姿

cv::Mat GetPoseInverse(); //获得当前帧位姿的逆

cv::Mat GetCameraCenter(); //获得相机光心坐标

cv::Mat GetStereoCenter(); //获得双目中心

cv::Mat GetRotation(); //获得当前帧的姿态旋转矩阵

cv::Mat GetTranslation(); //获得当前帧的位移

void ComputeBoW(); //计算Bow，统计单词

void AddConnection(KeyFrame\* pKF, const int &weight); //将关键帧加入共视关键帧中，并重新排序（降）共视帧序列

void EraseConnection(KeyFrame\* pKF); //在共视关键帧序列中删除输入关键帧，并重新排序（降）共视帧序列

void UpdateConnections(); //更新共视图的连接；如果没有初始化，更新生成树的



void UpdateBestCovisibles(); //重新排序（降）共视帧序列

std::set<KeyFrame \*> GetConnectedKeyFrames(); //得到与该关键帧有共视关系的关键帧，set类型

std::vector<KeyFrame\* > GetVectorCovisibleKeyFrames(); //得到与该关键帧有共视关系的关键帧，vector类型

std::vector<KeyFrame\*> GetBestCovisibilityKeyFrames(const int &N); //得到共视程度最大的前N个关键帧

std::vector<KeyFrame\*> GetCovisiblesByWeight(const int &w); //得到与当前关键帧共视程度大于等于w的关键帧

int GetWeight(KeyFrame\* pKF); //得到输入关键帧和当前关键帧的共视权重

void AddChild(KeyFrame\* pKF); //在生成树中将输入关键帧标记为当前关键帧的子帧

void EraseChild(KeyFrame\* pKF); //从当前关键帧的子帧序列中删除输入关键帧

void ChangeParent(KeyFrame\* pKF); //将当前关键帧的父帧更改为输入关键帧

std::set<KeyFrame\*> GetChilds(); //得到当前关键帧的子帧序列

KeyFrame\* GetParent(); //得到当前关键帧的父帧

bool hasChild(KeyFrame\* pKF); //判断输入关键帧是否为当前关键帧的子帧

void AddMapPoint(MapPoint\* pMP, const size\_t &idx); //添加三维点

void EraseMapPointMatch(const size\_t &idx); //删除三维点，输入为对应特征点的序号

void EraseMapPointMatch(MapPoint\* pMP); //删除三维点，输入为三维点，故要先通过三维点的obervation得到对应特征点的序号，再删除之

void ReplaceMapPointMatch(const size\_t &idx, MapPoint\* pMP); //替换相应序号对应的三维点为输入的三维点

std::set<MapPoint\*> GetMapPoints(); //得到所有“好”的三维点集合

std::vector<MapPoint\*> GetMapPointMatches(); //得到所有三维点

MapPoint\* GetMapPoint(const size\_t &idx); //得到对应序号的三维点

int TrackedMapPoints(const int &minObs); //返回关键帧中自身可被观测的关键帧数目大于等于minObs的三维点数量

cv::Mat UnprojectStereo(int i); //反投影得到第i个特征点相对世界坐标系下的坐标

void SetNotErase(); //不允许将当前关键帧设为坏帧

void SetErase(); //允许将当前关键帧设为坏帧

void SetBadFlag(); //将当前关键帧设为坏帧，并更新删除后的共视图以及生成树



float ComputeSceneMedianDepth(const int q); //评估当前关键帧场景深度

类KeyFrameDatabase

功能描述：

提供了单词->关键帧的索引；回环检测中寻找可能闭环的关键帧或重定位中寻找最相似的关键帧；

成员变量：

const ORBVocabulary\* mpVoc; //预先训练好的词典

std::vector<list<KeyFrame\*> > mvInvertedFile; //单词->关键帧的索引，mvInvertedFile[i]表示包含了第i个word id的所有关键帧

成员函数：

void add(KeyFrame\* pKF); //将输入关键帧的单词添加到索引内；

void erase(KeyFrame\* pKF); //从索引内删除和输入关键帧相关的单词->关键帧；

void clear(); //删除整个索引

std::vector<KeyFrame \*> DetectLoopCandidates(KeyFrame\* pKF, float minScore); //回环检测中寻找可能闭环的关键帧



std::vector<KeyFrame\*> DetectRelocalizationCandidates(Frame\* F); //重定位中寻找最相似的关键帧；