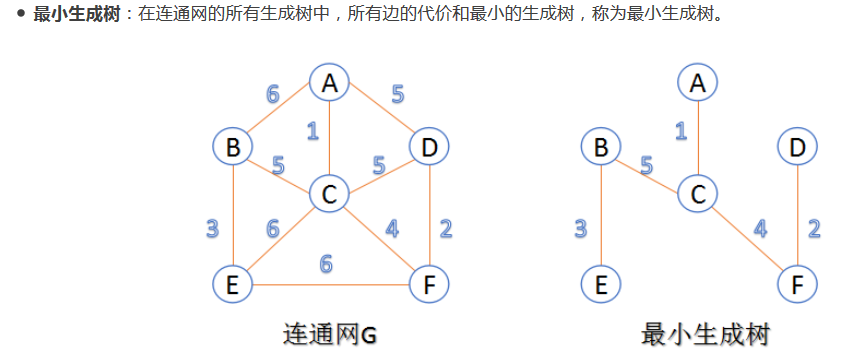
与slam14讲中的相机坐标系相同

往前是z轴正方向 右侧是x轴正方向 下面是y轴正方向

## 最小生成树(minimum spanning tree)

对于无向图G=(V, E),V表示图中的结点，E表示图中的边，所谓最小生成树就是联通图G中所有结点所需要的边长度总和最小，这些边加上V就构成图G的一颗最小生成树。这样还不清楚我们可以举个例子：有n个城市分布在不同的地方，为了保证每个城市间都能用火车通行，我们需要在城市之间修铁路，但是为了节约成本，我们在保证每两个城市之间都能联通的情况下要使得所修的铁路长度最短。这就是一个最小生成树的问题。



## covisibility graph 和 essential graph

covisibility graph(无向有权图):一直就有的概念，其顶点是相机的Pose，而边是Pose-Pose的变换关系——所以也算是Pose Graph 一种吧。当两个相机看到相似的空间点时，它们对应的Pose就会产生联系（我们就可以根据这些空间点在照片上的投影计算两个相机间的运动）。根据观测到的空间点的数量，给这个边加上一个权值，度量这个边的可信程度。

essential graph: 是orb-slam作者自己提出的概念，Essential Graph 比Covisibility Graph更为简单，ORB-SLAM主要用它来进行全局优化。为了限制优化的规模，ORB-SLAM试图尽量减少优化边的数量，使得essential graph能够连接所有的node但是edge会减少很多，可以认为是covisibity graph的最小生成树。



## 4、ComputeStereoMatches

涉及到双目匹配的工作：



对于建立带状搜索区域的说明：

我们都知道双目相机有一个特性就是相同的特征点在图像上有相同的v坐标，如下图所示：



那么可能在这个相同的v高度上不只有一个特征点，那么我们就需要轮询这三个候选的特征点和左图像的进行描述子比较，将描述子差值最小的那个认为是匹配的特征点。

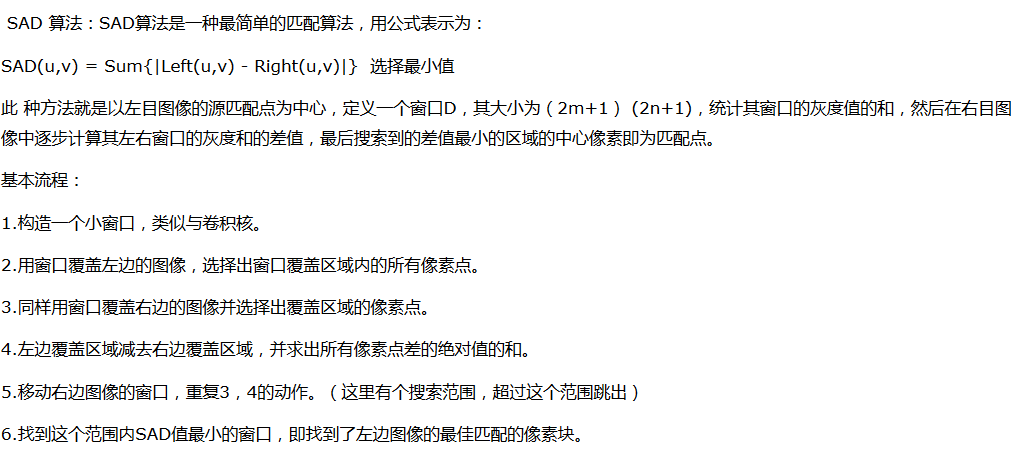


但是为了增加程序的鲁棒性我们将与v1坐标相差为r的特征点也考虑作为候选特征点：



作者在写这段程序时使用了一个变量vRowIndices，这个变量存储的内容为[右图像的v坐标，坐标v上的候选右图像特征点在右图像上的序号]，以上图为例V1-r到V2的坐标只存储了1，V2到V1+r的坐标存储了1和2。也就是说当我们得到左图像的像素坐标V1后，右图像的候选特征点是1和2。

这里解释一下SAD滑窗：



## 已知将世界坐标下的点变换到相机坐标下点的位姿变换求





则 

## vcell解释



这是一幅图像，将其分成4个cell，每个cell中都有三角形的特征点。所有的特征点由vector存储，在vector中的各自的编号是大写的数字。而在每个cell中的编号是小写的数字。vCell[j]，j就是cell中的小写数字而vCell[j]就是大写数字。

## 旋转一致性检测

其中角度直方图是用来剔除不满足两帧之间角度旋转的外点的，也就是所谓的旋转一致性检测

1、将关键帧与当前帧匹配特征点的angle相减，得到rot（0<=rot<360），放入一个直方图中，对于每一对匹配点的角度差，均可以放入一个bin的范围内（360/HISTO\_LENGTH）。

2、统计直方图最高的三个bin保留，其他范围内的匹配点剔除。另外，若最高的比第二高的高10倍以上，则只保留最高的bin中的匹配点。



如上图所示，左右两帧图像，红色的点表示对应的特征点，其中我们舍弃特征点7因为它不在前三个直方图中。

## 9、BowVector，FeatureVector

1、BowVector即是分类树中leaf的数值与权重   
2. FeatureVector: 是分类树中leaf的id值与对应输入ORB特征列表的特征序号.



红色的1 2 3表示词典中单词的序号；

蓝色的2 1 3 表示对应这个单词在这帧图像中出现的次数；

黑色的字表示在这帧图形中序号为多少的特征点属于这个单词，可以看到第1和第6的特征点属于单词1，一共出现了两次；

BowVector存储的是  

FeatureVector存储的是  

## 10、mObservations和keyframeCounter



mObservations中存储的是与当前帧共视的关键帧并记录其对应特征点在关键帧中的编号。

上图中相同颜色表示相同匹配的特征点。序号是在各自帧中的特征点编号。

当前帧中的特征点1的mObservations结构存储的是{(关键帧1,1)}

当前帧中的特征点2的mObservations结构存储的是{}

当前帧中的特征点3的mObservations结构存储的是{(关键帧1,2),(关键帧2,3)}

当前帧中的特征点4的mObservations结构存储的是{(关键帧2,1)}

当前帧中的特征点5的mObservations结构存储的是{(关键帧2,2)}

注：这里一定要注意！mObservations是地图点的变量

keyframeCounter中存储的是各个关键帧与当前帧共视个数：

关键帧1与当前帧共视红色、绿色和蓝色特征点，共三个。

关键帧2与当前帧共视黄色、蓝色和绿色特征点，共三个。

则keyframeCounter内容为{(关键帧1,3)，(关键帧2,3)}

## 25、ComputeDistinctiveDescriptors、

当我们通过不同的关键帧观测到了相同的地图点，这个数据结构保存在了mObservations变量中，那我们如何确定地图中这个地图点的描述子呢？

假设这个地图点在五个不同的关键帧中观测到了，则就有5个描述子可以描述这个地图点，首先我们求每两个描述子的差值:

1. 10 20 30 40
2. 12 21 31 41
3. 13 23 32 42
4. 14 24 34 43

这里的01表示第0个描述子和第1个描述子的差值。注意这里作者设定01=10.

然后按照列对差值进行排序，例如对第一列01 02 03 04 进行排序得到第一列的中值，依次类推分别得到第二列到第五列的中值，然后再取这五个列中值的最小值作为这个地图点的描述子。例如：假设第一列的排序后中值是12，第二列是89，第三列是90，第四列是5，第五列是23,则这个地图点描述子就是第四个描述子。

## 11、特征点尺度

在特征点被识别的过程中需要将图像拆分成不同分辨率等级，在不同分辨率等级下然后检测图像的特征点。

当金字塔的层数越高时则图像的分辨率越高则远处的特征点才能被识别出来。当我们知道地图中的点距离成像平面的距离后，我们可以大致的估计出这个特征点在哪个层级的金字塔中能够被检测出来。这个应该就是PredictScale函数的作用。

**特征点的金字塔层数和深度的统一：详见createnewmappoints函数**

## 12、ComputeKeyPointsOctTree

<https://www.cnblogs.com/JingeTU/p/6438968.html>

## 13、LocalMapping::CreateNewMapPoints

我们已经知道了当前关键帧和其紧密帧(当前关键帧相连的权重最大的10个关键帧)的位姿，现在需要做的是将两个关键帧进行地图点匹配然后三角化特征点变成一个地图点，并更新这个地图点的信息和与其相关的紧密帧和当前关键帧的信息。

遍历紧密关键帧：

1、if 紧密关键帧距离当前关键帧太近，continue

2、计算当前关键帧和紧密帧的基础矩阵F

3、SearchForTriangulation : 搜索紧密关键帧和当前关键帧相匹配的特征点

4、遍历匹配的特征点：

确定使用帧内/帧间三角化匹配的特征点

将三角化的点反投影回紧密关键帧和当前关键帧上计算重投影误差

if 重投影误差太大， continue

if 距离比值和两幅图像下的金字塔层级比值应该相差很大，continue

三角化成功

SearchForTriangulation：

1. 首先得到两帧图像的词典
2. 遍历当前关键帧和共视关键帧的相同单词：

遍历当前关键帧属于这个单词的特征点：

if 这个特征点已经有对应的地图点，continue；

遍历共视关键帧属于这个单词的特征点:

if 这个特征点已经有对应的地图点，continue；

寻找与当前关键帧特征点描述子最相近的描述子并且这两个描述子之差必须小于阈值并且这两个特征点还需要满足极线定理

## 14、KeyFrame::SetBadFlag 二叉树的更新



上图用实线表示的是二叉树，用虚线表示的共视帧，序号表示关键帧的标号。蓝色的表示候选父节点。

父节点是关键帧1，关键帧1有一个子节点为关键帧2；关键帧2有三个子节点分别为关键帧3、关键帧4和关键帧5；关键帧3有三个共视关键帧分别为1、6和7，关键帧4有三个共视关键帧分别为1 3 9，关键帧5有两个共视关键帧分别为11和10

关键帧2要被删除后，我们需要找到关键帧3、关键帧4和关键帧5的父节点。

算法如下：

首先关键帧1一定是候选的父节点。

首先遍历关键帧3的共视关键帧1 6 7，发现有候选父节点关键帧1，则记录关键帧3和1之间的权重w1。

再遍历关键帧4的共视关键帧1 3 9，发现有候选父节点关键帧1，则记录关键帧4和1之间的权重w2。

再遍历关键帧5的共视关键帧11 10，发现没有有候选父节点关键帧1。

比较w1和w2的权重，这里我们假设w1>w2，则更改关键帧3的父节点为1，将关键帧3插入到候选父节点中。此时需要求父节点的子节点只剩下关键帧4和5了，父节点候选关键帧为1和3。



遍历关键帧4的共视关键帧1 3，发现有候选父节点关键帧1和3，则记录关键帧4和1之间的权重g1,记录关键帧4和3之间的权重g2。比较g1和g2，这里我们假设g1<g2，则关键帧4的父节点为3。此时需要求的关键帧就只剩下5了，父节点候选关键帧为1 3 4。



遍历关键帧5的共视关键帧11和10，发现没有候选父节点。则退出寻找

至此所多有的子节点已经找到了父节点，但是明明子节点5没有找到父节点么，对于这种情况，关键帧5的父节点是其爷爷节点也就是关键帧1。



## 15、DetectLoopCandidates

输入的参数是当前关键帧和最小分数。

作用是在所有当前关键帧的共视帧中搜索当前关键帧的候选关键帧。

下面讲解最小分数是如何得到：



黄色圆形的是当前关键帧，蓝色的是当前关键帧的共视帧。蓝色圆圈的分数表示共视帧与当前帧的Bow分数，分数越高表示越相似。我们可以看到这里最低分数就是50分，即我们输入的最小分数。



之后在所有关键帧中寻找与当前帧有相同单词的关键帧，但是这些关键帧不能与当前关键帧相连。如图中所示我们这里找到了三个关键帧分数分别为50 60 80 90。我们称为隔离单词帧。再对隔离单词帧进行筛选，删除得分低于50的关键帧得到优秀隔离单词帧。



然后得到优秀隔离单词帧的前10个共视帧，这里为了方便我们只画出三个共视帧。



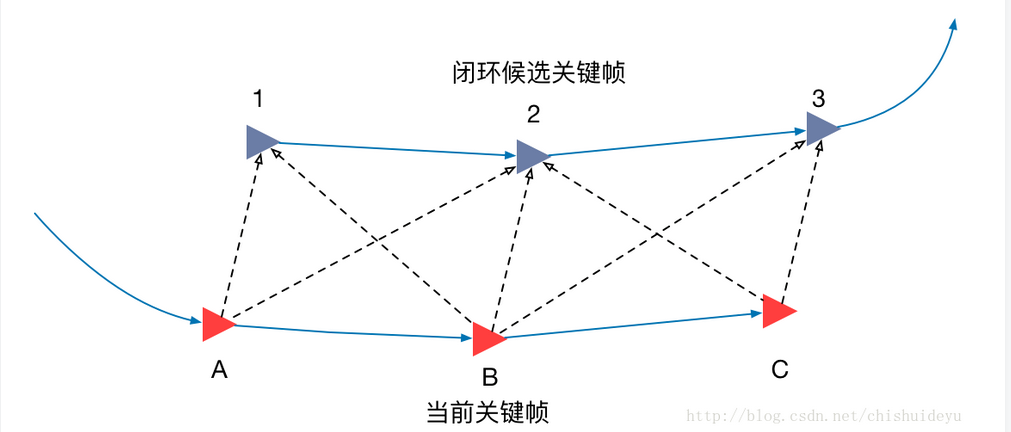
然后计算这些共视帧的分数和，删除一些比较低的分数和的共视帧。

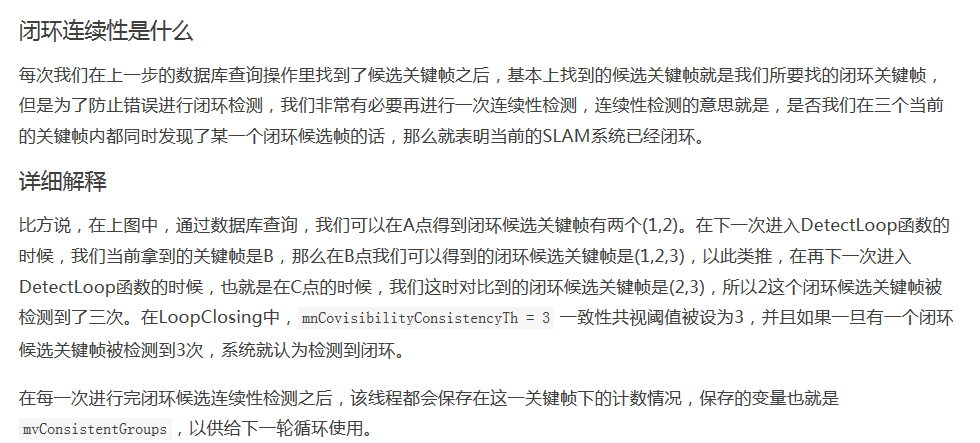


然后剩下的两个优秀隔离关键就是候选回环帧。

**然后再经历闭环连续性检测：**

**概念介绍：**





**举例：**

假设连续的四个关键帧都通过上面的流程检测到了闭环候选关键帧则我们假设如下的情形：



上图的意思是关键帧70的闭环候选帧是5 10 20。

可知70 71 72 73是四个连续的关键帧，闭环一致性检测需要维护一个变量mvConsistentGroups，当在70帧时mvConsistentGroups=，（5）表示关键帧5的子候选组(=候选回环关键帧+其共视帧)，0表示被连续观测到的次数。

当在71 帧时mvConsistentGroups=

当在72帧时mvConsistentGroups=

当在73帧时mvConsistentGroups=

则关键帧10满足一致性检测的条件，在连续的三个关键帧中都是候选关键帧。

注：这里一定要注意了，假设72关键帧没有找到闭环候选帧则mvConsistentGroups变量清空。

**算法流程：**

遍历回环候选帧：

1得到子候选组=候选回环关键帧+其共视帧；

2遍历子连续组(=mvConsistentGroups)：

2.1提取出子连续组中的第一个连续成员；

2.2遍历这个子候选组的关键帧：如果连续成员中有这个子候选组的关键帧，则break；

2.3如果上面找到了存在的关键帧，则

更新mvConsistentGroups。

## 16、Sim3Solver::CheckInliers

当我们使用三对匹配的地图点得到两帧的相对位姿变换(ICP)，如何判断其余样本中的点是否为inliner呢？



蓝色圆圈表示当前帧，红色圆圈表示闭环候选帧。我们已经得到红色帧和蓝色帧的相对位姿变换，判断任意一个样本即地图点(蓝色三角形)是否为inliner？

首先我们知道地图点在红色帧坐标系下的坐标，又知道红色帧和蓝色帧的相对位姿，则我们可以得到地图点在蓝色帧中图像的坐标，记为。我们也已知了地图点在蓝色帧的中的像素坐标记为。使用-=t，得到误差。然后将t与我们设定的阈值进行比较判断是否超过。



当然还没有完，我们再使用上面的原理，将地图点投影到红色帧图像中，并判断是否为inliner。

只有上述的两个条件同时满足我们才能说这个样本地图点是inliner。

## 17、PredictScale

已经知道地图点到当前帧的距离currentDist，想要估计这个地图点会在当前帧中图像的金字塔哪一层被检测出来。

同时我们也已经知道了mfMaxDistance(=dist\*levelScaleFactor)

dist：当前地图点到相机的距离

levelScaleFactor：地图点对应的特征点在所在的金塔层数所对应的图像缩放。

我们又知道如下的公式：

=levelScaleFactor

n是我们要求的特征点在哪一层金字塔检测出来

1.2是每层金字塔图像之间的缩放比例

ratio=mfMaxDistance/currentDist

则我们计算得到n的公式

## 18、CheckDistEpipolarLine

输入参数：kp1是当前帧图像特征点，kp2是共视帧图像特征点，F12是两个帧之间的基础矩阵,pKF2是共视帧。

kp1的坐标记为,kp2的坐标记为。。

如果两个点极线定理则：



但是orb算法中去了一个阈值，计：

=t，







如果小于此阈值则满足极线定理。此阈值的设置为3.84\*n\*n(n的定义是：在共视帧中特征点被检测出的那一层金字塔所对应图像缩放倍数的平方)。

这里a b c是共视帧图像中的直线方程系数：ax+by+c=0，如果理想情况下将带入x,y,z结果是0，但是实际情况下，点距离直线ax+by+c=0有一定的距离，而这个距离的平方就是

## 19、DetectRelocalizationCandidates

输入的参数是当前帧，目的是在所有的关键帧中找到与当前帧相似程度比较高的候选关键帧。作者在这里使用了bow单词数据结构来进行关键帧和当前帧的相似度的计算。

1. 首先得到当前帧的单词，然后找出拥有这些单词的所有关键帧。然后记录这些关键帧与当前帧拥有相同单词的个数，将那些拥有较少相同单词数的关键帧剔除，然后得到候选关键帧。剔除策略是得到相同单词最多个数，然后乘以0.8，以此为阈值。
2. 得到候选关键帧的前10名的共视帧，这里为了方便我们将这些共视帧成为候选关键帧的紧密帧，将这些紧密帧的分数相加然后将从这些紧密帧中选出与当前帧相似度最高的最优紧密帧。现在我们构成了一个数据结构[最优紧密帧，紧密帧的分数和]，如果紧密帧的分数和低于阈值则将这个最优紧密帧删除。这个阈值的设定是得到最优紧密帧的最高分数乘以0.75。这样我们对第一步得到的候选关键帧进行了筛选得到了新的候选关键帧。
3. 新的候选关键帧就是与当前帧相似度较高的关键帧。

## 20、PnPsolver::iterate

已经知道了当前帧的特征点和关键帧地图点的匹配，并且关键帧中地图点在世界坐标系下的位置已知，求当前帧的位姿。

这里我们会得到两个迭代次数mRansacMaxIts(ransac的最大迭代次数)和nIterations(人为设定的迭代次数)。当我们通过公式计算得到ransac最优迭代次数大于我们人为设定的迭代次数则mRansacMaxIts等于人为设定的迭代次数。只有人为设定的迭代次数大于公式计算得到的迭代次数时，mRansacMaxIts才是公式计算得到的。这里作者为什么要这么做?是因为如果通过公式计算得到的迭代次数太大会浪费很多的时间。

先随机找到四个点对使用epnp计算得到当前相机的位姿，然后使用这个位姿计算inliner，如果inliner的数量满足最小值：则使用这个相机的位姿更新相机位姿的最优解，然后再将 这些inliner应用到epnp中计算得到相机的位姿，再用这个位姿计算inliner，如果这次inliner还是满足最小值则我们结束ransac。

如果inliner的数量不满足最小值：则继续下一次ransac过程。直到达到最大迭代次数。

## 21、视差角的计算(CreateNewMapPoints函数中出现)

需要用到帧内视差角和帧间视差角：

帧内视差角计算方法(只针对双目情况)



tan(θ/2)=(b/2)/d，则θ=2\*artan(b/2d)

注：我认为帧内视差角度的测量公式不应该是这个，作者做了两个假设：第一个假设是空间点一定位于左右相机的中间位置，想想这怎么可能呢？；第二个假设是，作者使用z轴上的角度近似代替了真实的角度，如下图所示：通过作者的公式计算得到的是，而实际需要计算的是。我认为实际的角度应该和帧间的角度计算类似，这里直接给出结果，真实的帧内角度其实就是计算K.inverse\*像素其次坐标得到的向量的之间的夹角，也就是归一化平面向量的夹角。

那我们需要思考为什么orb作者要使用这个公式来计算帧内角度，我们通过下面两个图可以看出来，当地图点与OL和OR所构成的平面下角度最大，当地图点在OL和OR中间时角度最大，则作者直接计算了一个最大的角度，而这个角度只和深度相关。根据实际计算发现帧内角度的真实值和orb作者的这个值相差很小。

、



帧间视差角计算公式：得到两帧匹配的特征点，并根据相机的内参计算得到这两个特征点的归一化坐标分别为x1和x2，然后构造两个向量R1\*r1与R2\*r2，其中R1和R2分别是这两帧将相机坐标系的点变换到世界坐标系点的旋转矩阵，然后计算这里两个向量的夹角就是帧间的视差角。



推导如下：已知O1和O2帧在世界坐标系下的位姿为[R1,t1]和[R2,t2]，我们又得到了两帧图像匹配的特征点像素坐标x1和x2，需要求O1P和O2P之间的夹角。

首先求O1和O2在世界坐标系下的位置，可知分别为t1和t2。我们将两像素点结合相机内参计算得到归一化坐标X1和X2，注意归一化坐标是在相机坐标系下的坐标，我们将相机下的归一化坐标转换成世界坐标系坐标R1\*X1+t1，R2\*X2+t2，则我们分别得到了O1P向量在世界坐标下的表示=R1\*X1+t1-t1=R1\*X1，O2P向量在世界坐标系下的表示R2\*X2则我们能够通过这两个向量计算得到帧间的夹角。

则进行三角化的原则是：

如果帧间的视差角大于帧内视差角并且帧间视差角不是很小的情况下，使用三角法(使用两帧的左图像中匹配的像素点进行三角化，可以理解为移动距离比较大)；

对于双目,如果帧间的视差角比较小或者帧间视差角小于帧内视差角，谁的帧内视差角大用谁恢复出三维点坐标；

## 22、三角化方法

<http://www.cnblogs.com/liufuqiang/p/6792763.html>

三角化方法有三种，orb算法中使用的是mvg中的方法P312页

## 23、orb算法世界坐标系

与slam14讲中的90页的相机坐标系相同。

## 24、orb输出的相机位姿

在orb结束时会输出一个txt文档输出每个相机的位姿，那么每个相机的位姿是如何得到的呢？每个帧都有一个参考关键帧，我们其实只记录这个帧相对于参考关键帧的相对位姿，而在优化位姿时我们只优化关键帧的位姿。

那么每个帧的参考关键帧是如何得到的呢？其实每个帧的参考关键帧只在track线程中就指定好了。在track线程就是使用track结构中的mpReferenceKF来更新当前帧的参考关键帧。

那么track结构中的mpReferenceKF变量是在哪里被更新的呢？这里我们分两种情况：

初始化情况：第一个帧就是关键帧，且其参考帧就是自己，track结构中的mpReferenceKF也是当前关键帧。

非初始化情况：与当前帧共视最多的关键帧设置为当前帧的参考帧和track线程的参考帧。之后再判断当前帧是否为关键帧，如果当前帧是关键帧则参考帧再设置为自己，并设置track结构中的mpReferenceKF是当前关键帧。

## 26-1 SearchByBoW(用于TrackReferenceKeyFrame函数)

这个函数的主要作用是关键帧中的地图点已知，想要和普通帧中的特征点进行匹配，输出的结果是匹配上的关键帧中的地图点。

对关键帧的不同单词进行遍历：

遍历隶属于关键帧这个单词的特征点：

if 如果关键帧的这个特征点没有对应的地图点，continue

遍历隶属于相同单词的当前帧中的特征点进行遍历：

计算这两个特征点的描述子的距离，选出当前帧距离关键帧这个特征点的特征点；

if 最小的描述子距离是在阈值内，则更新输出的匹配点。

进行旋转一致性检测，剔除不好的点。

## 26-2 SearchByBoW(用于ComputeSim3函数)

## 27、SearchByProjection（用于TrackWithMotionModel函数）

我们已经知道了上一帧的位姿和当前帧的一个预估位姿，并且知道上一帧中地图点，这个函数的作用就是将上一帧的地图点根据当前帧的预估位姿投影到当前帧的图像中。

轮询上一帧中的所有地图点：

if 这个地图点不是非线性优化的异值点：

将这个地图点投影到当前帧中，得到这个像素点周围的候选特征点；

遍历这些候选特征点：

if 这个当前帧的这个特征点有右图像相匹配的点，用视差公式计算右图像的像素坐标与双目匹配的到的像素坐标比较，如果相差太大则continue；

更新最小的描述子距离；

如果最小的描述子距离<阈值，将上一帧的地图点添加到当前帧；

对得到的结果进行旋转一致性筛选。

## 28、SearchByProjection（用于SearchLocalPoints函数）

已知局部地图点的位置和当前帧的位姿，将局部地图点投影到当前帧中并与当前帧的特征点进行匹配,如果匹配成功则将这个地图点添加到当前帧中

整体流程大致与上面的相同，只是此函数没有旋转一致性筛选。

## 29、ComputeSim3

此函数的目的是确定闭环候选帧和当前帧的位姿变化[R,t]，我们已经有了初步的闭环候选关键帧。

1遍历闭环候选关键帧：

1.1将当前关键帧与闭环候选关键帧的地图点进行匹配；

1.2如果匹配的点数小于20，continue；

1.2如果匹配的点数大于20，设置这个闭环候选帧的ransac参数

2遍历上面匹配点数大于20的闭环关键帧：

2.1首先使用上面匹配的地图点+ransac算法计算得到当前关键帧相对于闭环候选帧的位姿变化；

2.2如果计算得到了有效位姿：

通过上面计算得到的相对位姿初值，将闭环候选关键帧的地图点投影到当前帧中获得更多的地图点匹配；

再使用非线性优化计算得到更精确的相对位姿；

如果非线性优化的有效边数大于20，则找到了最优闭环候选帧停止遍历；

3如果通过第2步的遍历没有候选关键帧满足条件，则返回false

4如果找到了最优闭环候选关键帧：

4.1将最优候选闭环帧的共视帧看到的地图点与当前帧特征点进行匹配，此时会产生更多的匹配点；

4.2 如果匹配点数>=40，则认为这个最优候选闭环帧有效，返回true

如果匹配的点数<40,则认为这个最优候选闭环帧无效，返回false