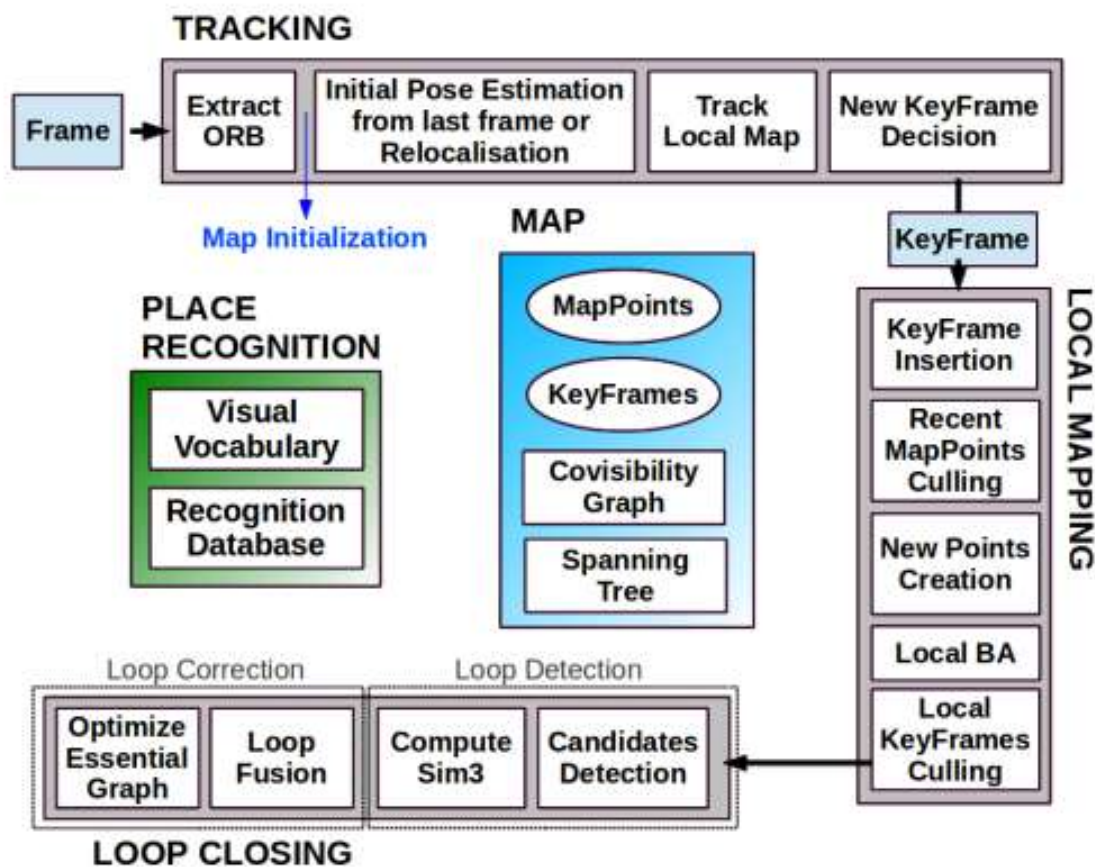


# ORB-SLAM代码详细解读

吴博 @泡泡机器人  
657390323@qq.com  
2016.8.29

# 代码主要结构

2



Tracking.cpp

LocalMapping.cpp

LoopClosing.cpp

Viewer.cpp

变量命名规则:

“p”表示指针数据类型      “n”表示int类型

“b”表示bool类型      “s”表示set类型

“v”表示vector数据类型      ‘l’表示list数据类型

“m”表示类成员变量

# System入口:



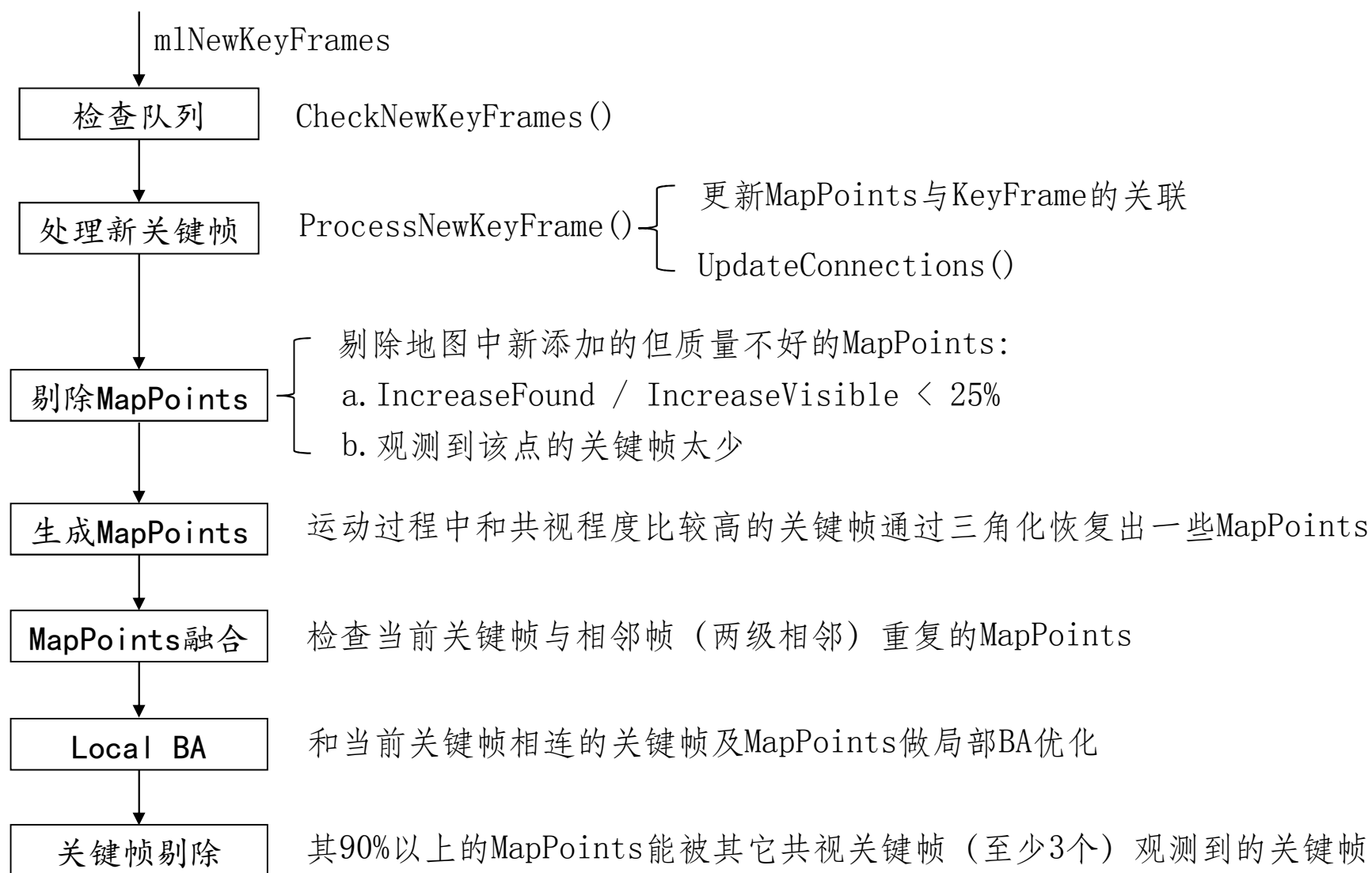
注: mpIniORBextractor相比mpORBextractorLeft提取的特征点多一倍

# Tracking线程:



注: `mbOnlyTracking`默认为false, 用户可通过运行界面选择仅跟踪定位模式

# LocalMapping线程:



# LocalClosing线程(闭环检测):

mlploopKeyFrameQueue

队列中取一帧

mpcurrentKF

判断距离上一次闭环检测是否超过10帧

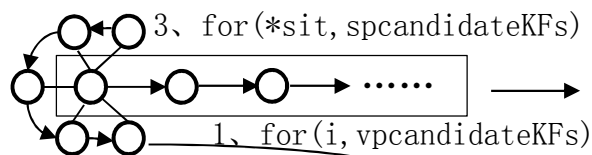
计算当前帧与相连关键帧的Bow最低得分

mpcurrentKF  
minscore

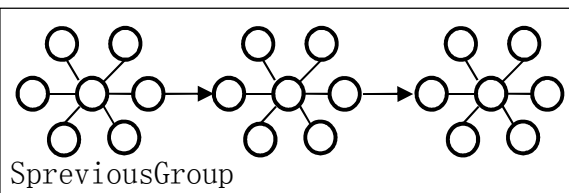
检测得到闭环候选帧

vpLoopCandidates

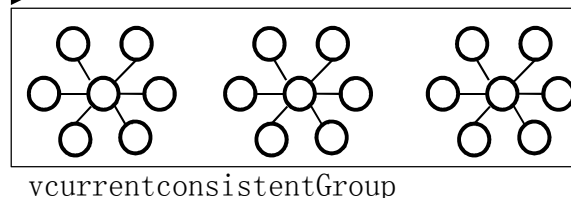
检测候选帧连续性



Result:  
mvpEnoughConsistentcandidates



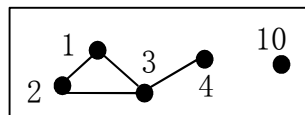
2, for(iG, mvConsistentGroup)



1、三个阈值都是计算获得，鲁邦性好  
minscore mincommons minscoreToRetain

2、通过分组可以将单独得分很高的无匹配关键帧剔除

分组示意图:



如图: 1、2、3、4、10都是闭环候选帧。

节点1: 与2、3相连, 1与2、3分为一组

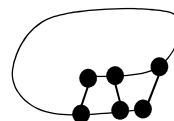
节点2: 与1、3相连, 2与1、3分为一组

节点3: 与1、2、4相连, 3与1、2、4分为一组

节点4: 与3相连, 4与3分为一组

节点10: 10自己单独一组

连续性检测示意图:



(pKF, minscore)

找出与当前帧有公共单词的关键帧, 但不包括与当前帧相连的关键帧

lKFsharingwords

统计候选帧中与pKF具有共同单词最多的单词数

maxcommonwords

得到阈值

$\text{mincommons} = 0.8 * \text{maxcommonwords}$

maxcommonwords  
mincommons  
minscore

筛选共有单词大于mincommons且Bow得分大于minscore的关键帧

lscoreAndMatch

将存在相连的分为一组, 计算组最高得分bestAccScore, 同时得到每组中得分最高的关键帧

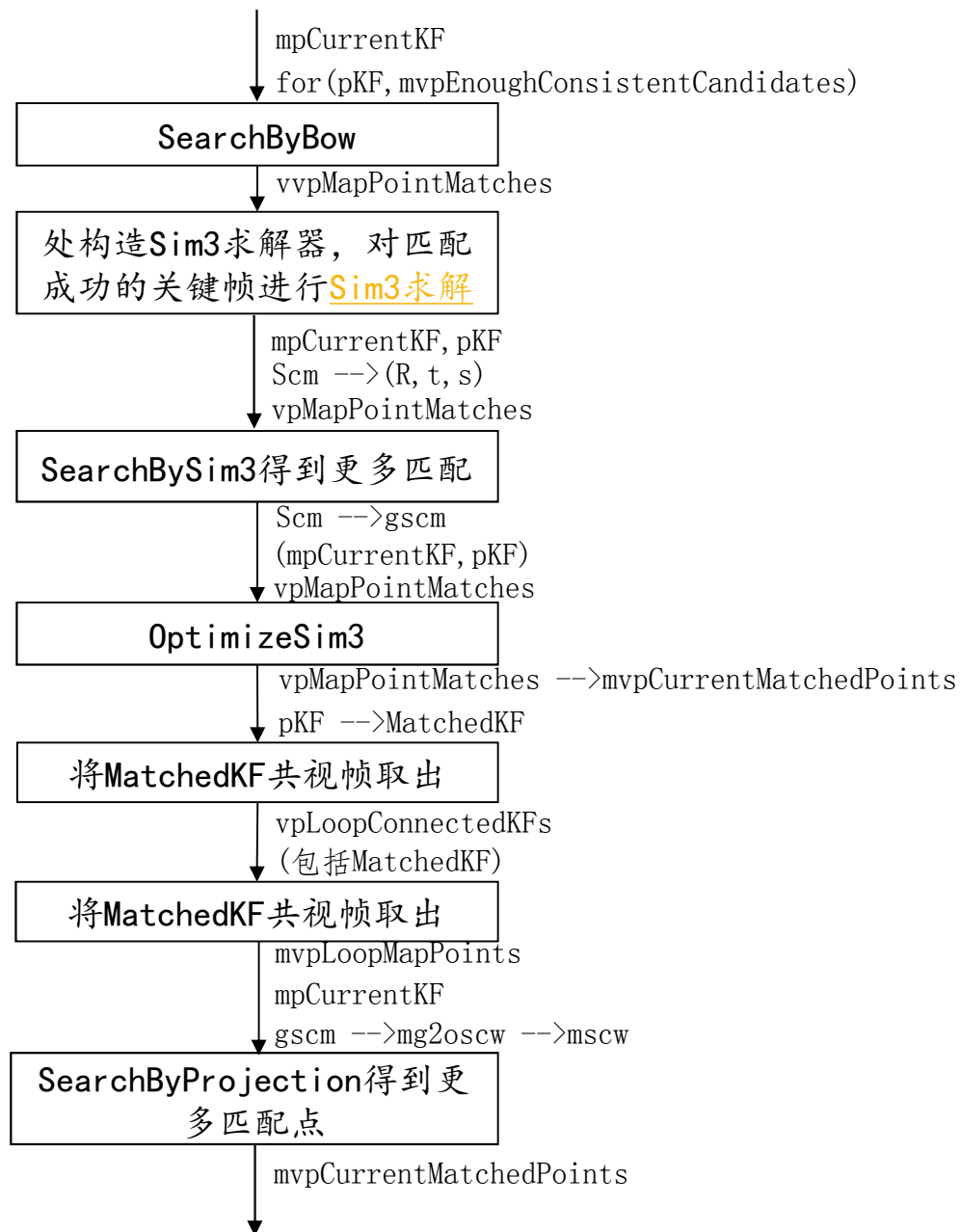
lsAccScoreAndMatch  
bestAccScore

得到阈值minScoreToRetain  
 $= 0.75 * \text{bestAccScore}$

lsAccScoreAndMatch  
minScoreToRetain

vpLoopCandidates

## LocalClosing线程 (Sim3计算):



# LocalClosing线程(Sim3计算):

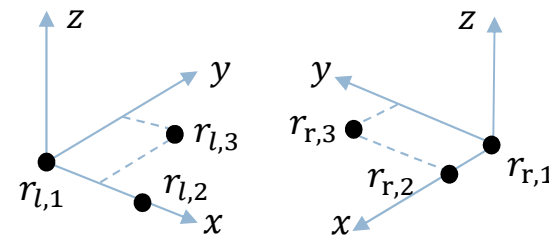
三对匹配3D，分别对左右三个3D点建立坐标系：

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

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

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

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



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：

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

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

$$\sum_{i=1}^n r'_{l,i} = 0 \quad \sum_{i=1}^n r'_{r,i} = 0$$

$$r'_{r,i} = sR(r'_{l,i}) - r'_0 \Rightarrow r'_0 = r_0 - \bar{r}_r + sR(\bar{r}_l) \quad (7)$$

$$\text{优化函数: } \sum_{i=1}^n \|e_i\|^2 = \sum_{i=1}^n \|r'_{r,i} - [sR(r'_{l,i}) + r'_0]\|^2 \quad (8)$$

$$= \sum_{i=1}^n \|r'_{r,i} - sR(r'_{l,i})\|^2 - 2r'_0 \cdot \sum_{i=1}^n [r'_{r,i} - sR(r'_{l,i})] + n\|r'_0\|^2$$

$$\sum_{i=1}^n \|r'_{r,i} - sR(r'_{l,i})\|^2 \quad \text{与优化量 } r'_0 \text{ 无关}$$

$$\min \sum_{i=0}^n \|e_i\|^2 \Rightarrow r'_0 = 0 \Rightarrow t = r_0 = \bar{r}_r - sR(\bar{r}_l) \quad (9)$$



# LocalClosing线程 (Sim3计算):

计算尺度:

$$\text{由于 } r'_0 = 0 \Rightarrow \sum_{i=1}^n \|e_i\|^2 = \sum_{i=1}^n \|r'_{r,i} - sR(r'_{l,i})\|^2 \quad (10)$$

$$\Rightarrow \sum_{i=1}^n \|e_i\|^2 = \sum_{i=1}^n \|r'_{r,i}\|^2 - 2s \sum_{i=1}^n r'_{r,i} \cdot R(r'_{l,i}) + s^2 \sum_{i=1}^n \|R(r'_{l,i})\|^2$$

$$\Rightarrow \sum_{i=1}^n \|e_i\|^2 = \sum_{i=1}^n \|r'_{r,i}\|^2 - 2s \sum_{i=1}^n r'_{r,i} \cdot R(r'_{l,i}) + s^2 \sum_{i=1}^n \|r'_{l,i}\|^2$$

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

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

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

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

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

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

则公式11变为:

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

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

## LocalClosing线程(Sim3计算):

如果对于大于三组匹配点:

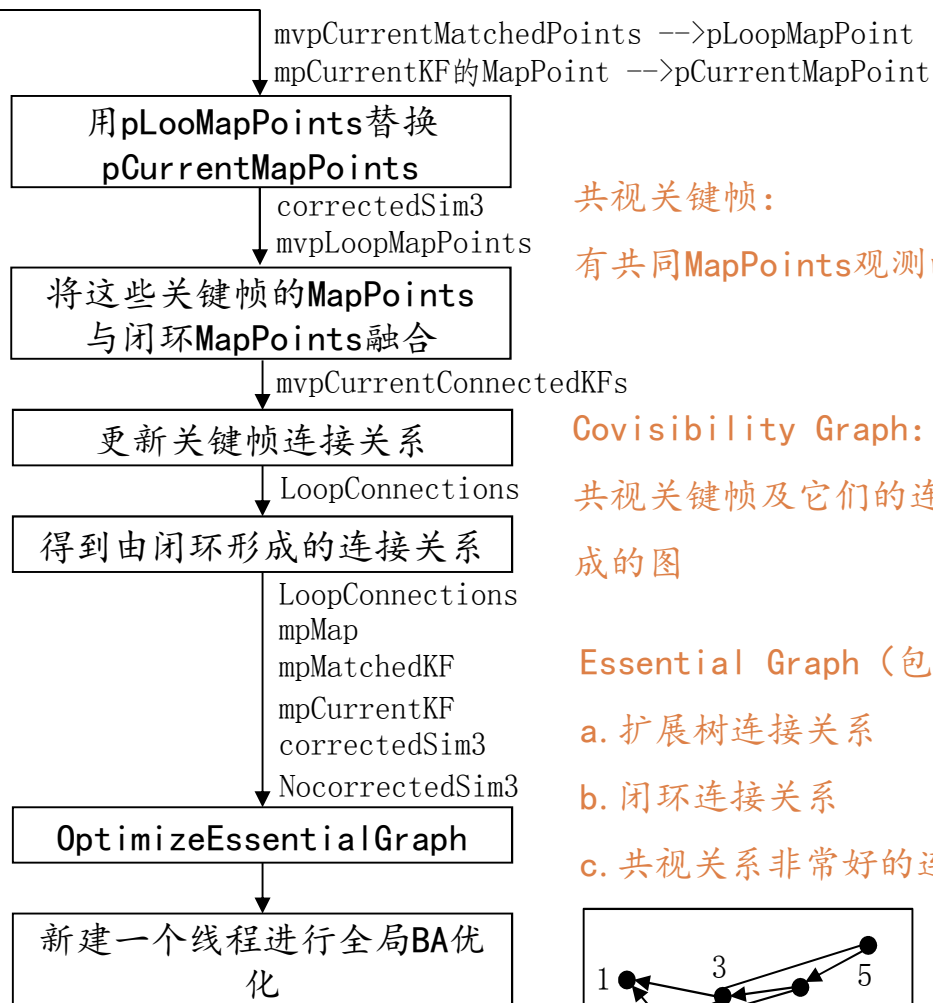
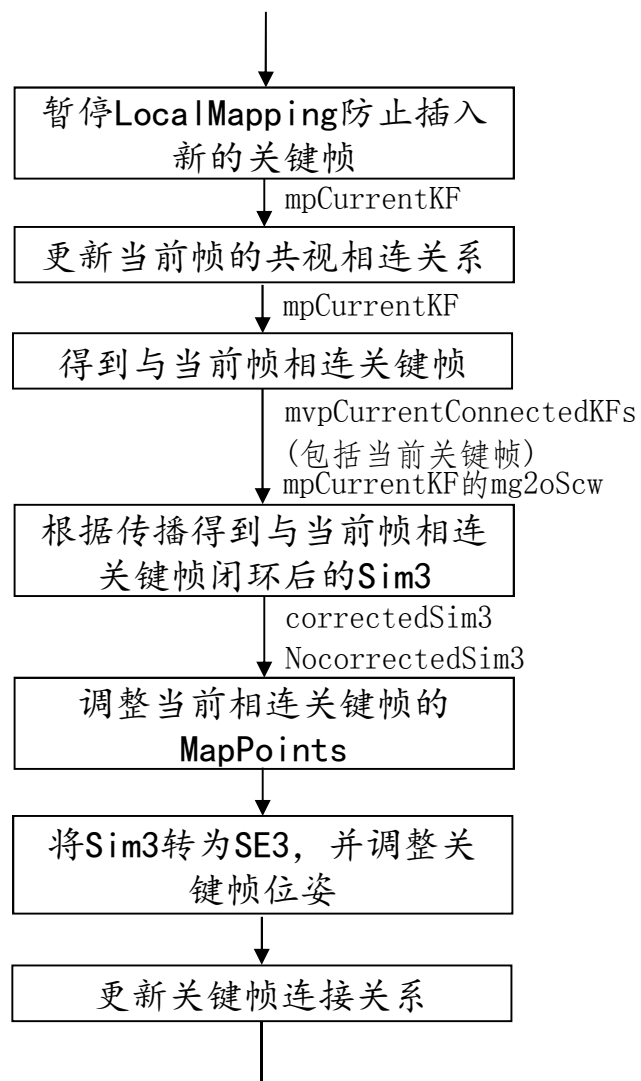
$$\text{令: } M = \begin{bmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{bmatrix} = \sum_{i=1}^n r'_{l,i} r'^T_{r,i}$$

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

特征值分解N矩阵: N最小特征值对应的特征向量就是待求四元数

四元数转欧拉角:  $q = \cos(\theta/2) + n\sin(\theta/2)$

# LocalClosing线程 (correctLoop):



共视关键帧:

有共同MapPoints观测的关键帧

Covisibility Graph:

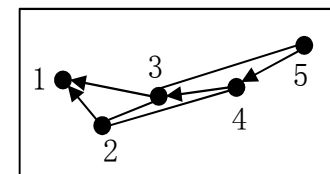
共视关键帧及它们的连接关系构成的图

Essential Graph (包含3部分):

a. 扩展树连接关系

b. 闭环连接关系

c. 共视关系非常好的连接关系

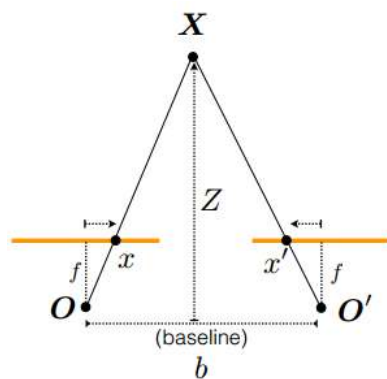


# Frame.cpp:

## ■ 双目立体匹配

- ✓ 为左目每个特征点建立带状区域搜索表，限定搜索区域。（已提前极线校正）
- ✓ 通过描述子进行特征点匹配，得到每个特征点最佳匹配点 `scaleduR0`
- ✓ 通过SAD滑窗得到匹配修正量 `bestincR`
- ✓  $(\text{bestincR}, \text{dist})$   $(\text{bestincR}-1, \text{dist})$   $(\text{bestincR}+1, \text{dist})$  三个点拟合出抛物线，得到亚像素修正量 `deltaR`
- ✓ 最终匹配点位置为: `scaleduR0 + bestincR + deltaR`

## ■ Disparity与Depth



$$\frac{X}{Z} = \frac{x}{f} \quad \frac{b - X}{Z} = \frac{x'}{f}$$

$$\Rightarrow \text{Disparity: } d = x - x' = \frac{bf}{Z}$$

# 单目Initializer.cpp:

- 特征点归一化，坐标均值为0，一阶绝对矩为1

$$mean\_x = (\sum_{i=0}^N u_i)/N \quad mean\_y = (\sum_{i=0}^N v_i)/N \quad (1)$$

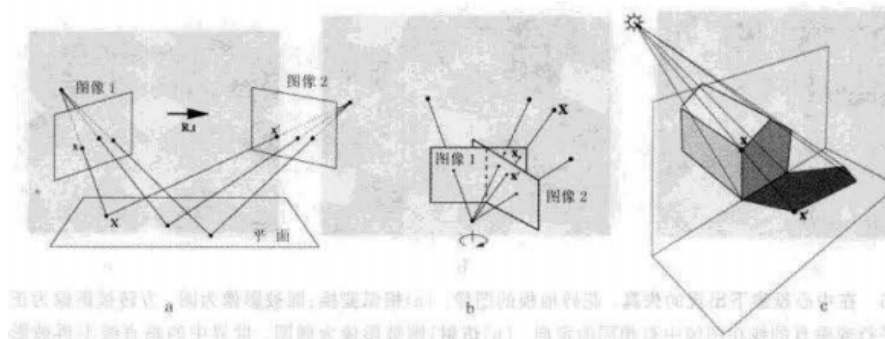
$$mean\_devx = (\sum_{i=0}^N |u_i - mean\_x|)/N \quad mean\_devy = (\sum_{i=0}^N |v_i - mean\_y|)/N \quad (2)$$

$$sX = 1/mean\_devx \quad sY = 1/mean\_devy \quad (3)$$

$$T = \begin{bmatrix} sX & 0 & -meanx * sX \\ 0 & sY & -meany * sY \\ 0 & 0 & 1 \end{bmatrix} \quad (4)$$

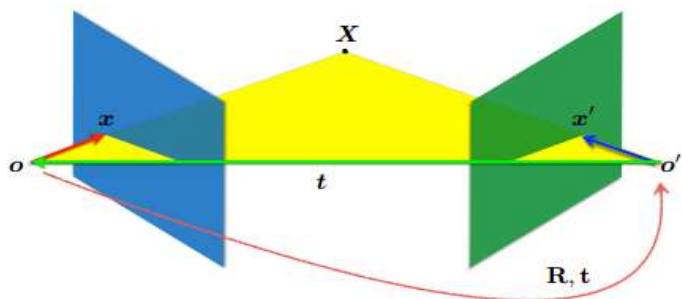
- 单应性矩阵模型 (Homograph Matrix)

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}_{right} = \lambda \begin{bmatrix} h1 & h2 & h3 \\ h4 & h5 & h6 \\ h7 & h8 & h9 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}_{left} \quad (5)$$



# 单目Initializer.cpp:

## 对极几何模型 (Fundamental Matrix)



刚体旋转  $x' = R(x - t)$  (1)

共平面  $(x - t)^T(t \times x) = 0$  (2)

$$\Rightarrow (x'^T R)(t \times x) = 0 \Rightarrow (x'^T R)([t_\times]x) = 0$$

$$\Rightarrow x'^T (R[t_\times])x = 0 \Rightarrow \mathbf{x'^T E x} = 0 \quad (3)$$

图像坐标系转相机坐标系:

$$\hat{x} = k^{-1}x \quad \hat{x}' = k'^{-1}x' \quad (4)$$

$$\Rightarrow \mathbf{x'^T F x} = 0 \quad \mathbf{F} = \mathbf{K'^{-T} E K^{-1}} \quad (5)$$

## Homograph矩阵求解

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}_{right} = \lambda \begin{bmatrix} h1 & h2 & h3 \\ h4 & h5 & h6 \\ h7 & h8 & h9 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}_{right} \Rightarrow x' = \lambda H x$$

DLT求解:  $x' \times Hx = 0 \Rightarrow Ah = 0$  (6)

其中:  $A = \begin{bmatrix} 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \\ -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \end{bmatrix} \Rightarrow$  特征值分解:  $A^T A$

最小特征值对应的特征向量

$$h = [h1 \ h2 \ h3 \ h4 \ h5 \ h6 \ h7 \ h8 \ h9]'$$

# 单目Initializer.cpp:

## ■ Fundamental 矩阵求解

$$x'Fx = 0 \quad F = \begin{bmatrix} f1 & f2 & f3 \\ f4 & f5 & f6 \\ f7 & f8 & f9 \end{bmatrix}$$

$$\Rightarrow Af = 0 \quad (1)$$

$$\text{其中: } A = \begin{bmatrix} x'x & x'y & x' & y'x & y'y & y' & x & y & 1 \end{bmatrix}$$
$$f = [f1 \ f2 \ f3 \ f4 \ f5 \ f6 \ f7 \ f8 \ f9]'$$

$\Rightarrow$  特征值分解分解:  $A^T A$   
最小特征值对应的特征向量

## ■ Fundamental 矩阵分解

$$E = k'^T F k$$

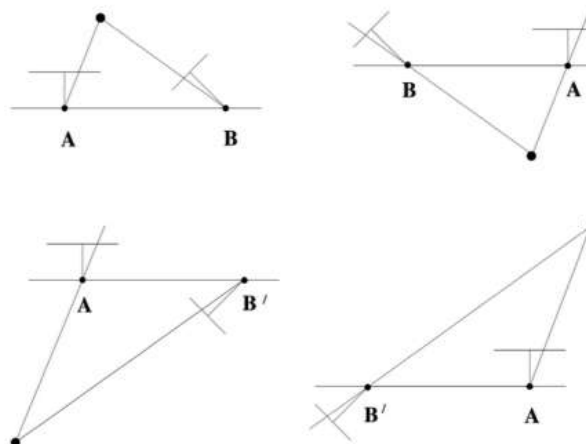
$$\text{SVD分解: } E = U \Sigma V^T \quad \text{令: } W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$E = [R|T] \begin{cases} R_1 = UWV^T & R_2 = UW^T V^T \\ T_1 = U_3 & T_2 = -U_3 \end{cases}$$

(R T) 选择:

统计四个模型中3D点在摄像头前方且投影误差小于阈值的3D点个数, 以及每个模型下较大的视差角。

如果其中一个模型的视差角大于阈值, 并且满足条件的3D点个数明显大于其它模型, 那么这个模型就是最优选择



# 单目Initializer.cpp:

## ■ Fundamental 模型评分

$$scoreF = \sum_{i=0}^N \rho(T_F - \|x'Fx\|^2/\sigma^2)$$

$$\rho(x) \begin{cases} 0 & x \leq 0 \\ x & else \end{cases} \quad T_H = 5.99$$

## ■ Homograph 模型评分

$$scoreH = \sum_{i=0}^N \rho(T_H - \|x' - Hx\|^2/\sigma^2) + \rho(T_H - \|x - H^{-1}x'\|^2/\sigma^2)$$

$$\rho(x) \begin{cases} 0 & x \leq 0 \\ x & else \end{cases} \quad T_F = 3.84$$

## ■ Homograph 模型与Fundamental模型选择

$$R_H = \frac{s_H}{s_H + s_F} \quad R_H > 0.45$$



# 单目Initializer.cpp:

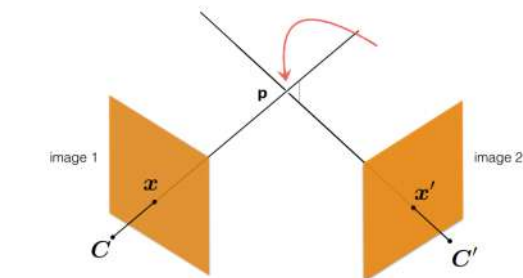
## 三角化恢复3D点

$(x, x')$  为匹配特征点对

$(P, P')$  分别为它们的投影矩阵

$$\Rightarrow x = PX \quad x' = P'X$$

$$\Rightarrow \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} p1 & p2 & p3 & p4 \\ p5 & p6 & p7 & p8 \\ p9 & p10 & p11 & p12 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$



简写: 
$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} - & P_0 & - \\ - & P_1 & - \\ - & P_2 & - \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

DLT求解: 
$$\begin{bmatrix} vP_2 - P_1 \\ P_0 - uP_2 \\ uP_1 - vP_0 \end{bmatrix} X = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

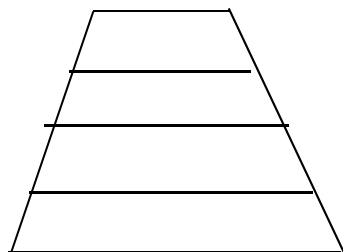
一组匹配点:

$$\begin{bmatrix} vP_2 - P_1 \\ P_0 - uP_2 \\ v'P'_2 - P'_1 \\ P_0 - u'P'_2 \end{bmatrix} X = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

## 尺度与距离

Nearer

Farther



Level:n-1  $\rightarrow$  dmin

Level:m  $\rightarrow$  d

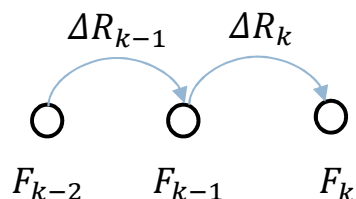
Level:0  $\rightarrow$  dmax

$$d/d_{\min} = 1.2^{(n-1-m)}$$

$$d_{\max}/d = 1.2^m$$

# Tracking.cpp:

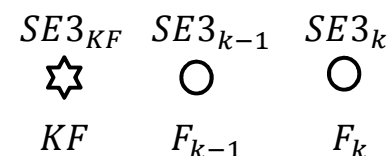
## TrackWithMotionModel



恒速模型:  $\Delta R \approx \Delta R_{k-1}$

这里是不是可以引入IMU来测量相对旋转呢

## TrackReferenceKeyFrame



跟踪参考帧模型:  $SE3_k \approx SE3_{KF}$

## Relocalization

EPnP求解:

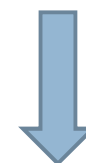
世界坐标系下有N个3D点:  $p_i^w \quad i = 1, \dots, n$

选择四个控制点:  $c_j^w \quad j = 1, \dots, 4$  质心, 三个主方向

对每个3D点, 可以找到4个  $\alpha_j$ , 使得:  $p_i^w = \sum_{j=1}^4 \alpha_{ij} c_j^w$ , 且:  $\sum_{j=1}^4 \alpha_{ij} = 1$  (1)

对于同样的  $\alpha_j$ , 可以使得:  $p_i^c = \sum_{j=1}^4 \alpha_{ij} c_j^c$  (2)

$p_i^c \quad c_j^c$  为待求量



# Tracking. cpp:

根据投影模型:

$$\lambda_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = P \cdot \sum_{j=1}^4 \alpha_{ij} c_j^c = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^4 \alpha_{ij} \begin{bmatrix} x_j^c \\ y_j^c \\ z_j^c \end{bmatrix} \quad (3)$$

展开可得:

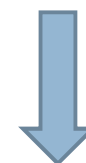
$$\sum_{j=1}^4 \alpha_{ij} f_u x_j^c + \alpha_{ij} (u_c - u_i) z_j^c = 0 \quad \sum_{j=1}^4 \alpha_{ij} f_v y_j^c + \alpha_{ij} (v_c - v_i) z_j^c = 0 \quad (4)$$

将 (4) 写成矩阵形式:  $Mx = 0$  其中待求量为四个控制点:  $x = [c_1^{cT}, c_2^{cT}, c_3^{cT}, c_4^{cT}]$  (5)

特征值分解M矩阵:  $x = \sum_{i=1}^N \beta_i v_i$  (6)

由于公式5中待求量是四个控制点，共有12个未知数，由公式4可知，每对3D-2D对应点可以形成两个约束，故理论上讲需要6组3D-2D对应点，可EPnP只需要至少4组即可，Why?

对于投影相机模型，公式 (6) 中N等于1，因为只有一个尺度变量；  
对于正交相机模型，公式 (6) 中N等于4，因为每个参考点的深度变化后仍满足约束；  
因此，当相机焦距比较小时，N为1。当相机焦距更大，相机接近于正交相机时， $M^T M$ 将有4个接近于0的特征值。



# Tracking. cpp:

四个参考点两两组合，可以得到6个距离，根据尺度不变性，可以得到如下约束：

$$N=1: \quad \|\beta v^{[i]} - \beta v^{[j]}\|^2 = \|c_i^w - c_j^w\|^2$$

$$\Rightarrow \quad \beta = \frac{\sum_{\{i,j\} \in [1;4]} \|v^{[i]} - v^{[j]}\| \cdot \|c_i^w - c_j^w\|}{\sum_{\{i,j\} \in [1;4]} \|v^{[i]} - v^{[j]}\|^2}$$

$$N=2: \quad \|(\beta_1 v_1^{[i]} + \beta_2 v_2^{[i]}) - (\beta_1 v_1^{[j]} + \beta_2 v_2^{[j]})\|^2 = \|c_i^w - c_j^w\|^2$$

.....

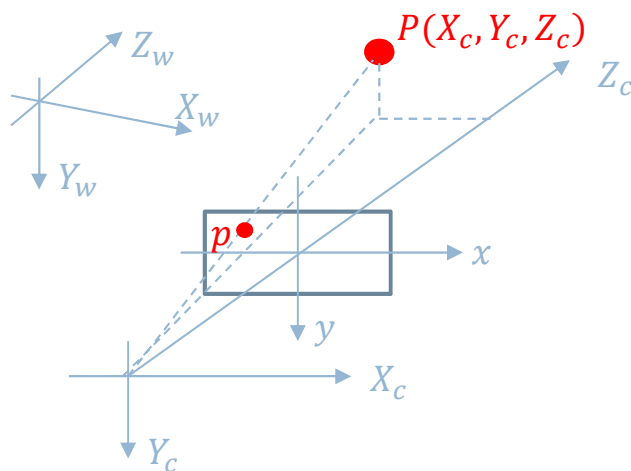
Gauss-Newton优化：

$$Error(\beta) = \sum_{(i,j) \text{ s.t. } i < j} (\|c_i^c - c_j^c\|^2 - \|c_i^w - c_j^w\|^2) \quad c_i^c = \sum_{j=1}^4 \beta_j v_j^{[i]}$$

# ORBmatcher.cpp:

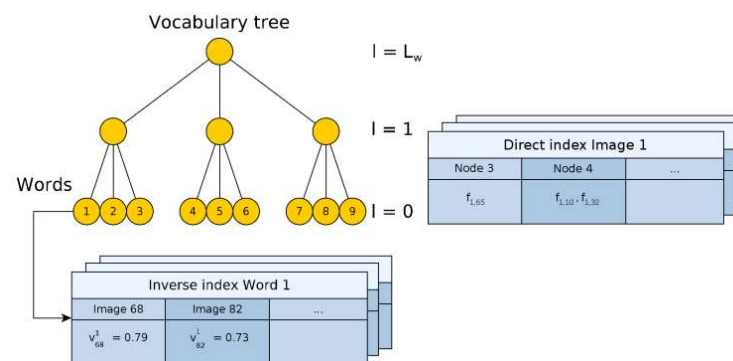
- 1、ORB-SLAM2中特征点匹配均采用了各种技巧减小特征点匹配范围。
- 2、ORB-SLAM2中特征点通过描述子匹配后会进行旋转一致性检测。并且最佳匹配特征点要明显优于次优匹配点。
- 3、ORB-SLAM2中特征点提取仍然是非常耗时的地方。

## ■ SearchByProjection与SearchBySim3



SearchByProjection函数和SearchBySim3函数利用将相机坐标系下的MapPoints投影到图像坐标系，在投影点附近搜索匹配

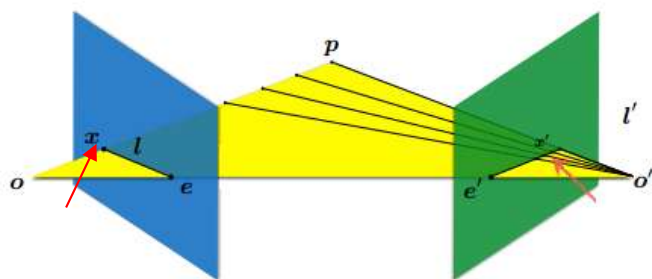
## ■ SearchByBoW



SearchByBoW函数通过判断特征点对应的word的node是否相同可以加速匹配过程

# ORBmatcher.cpp:

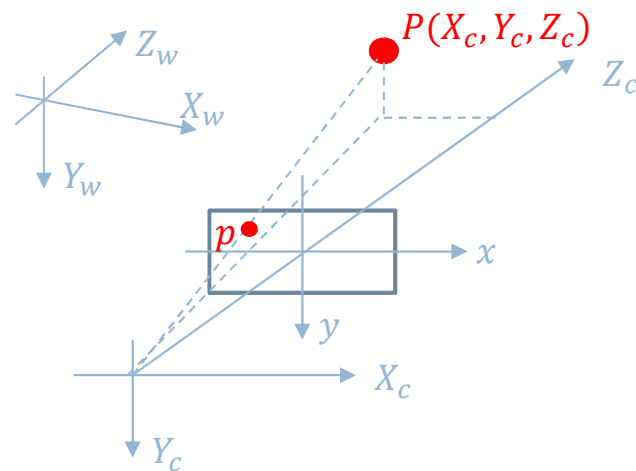
## ■ SearchForTriangulation



SearchByProjection函数利用对极几何约束：左目一个点对应右目一条线。

将左图像的每个特征点与右图像同一node节点的所有特征点依次检测，判断是否满足对极几何约束，满足约束就是匹配的特征点

## ■ Fuse



和SearchByProjection函数差不多，只不过是判断特征点p的MapPoint是否与MapPoint点P冲突

# Optimizer.cpp:

## ■ GlobalBundleAdjustment与LocalBundleAdjustment

3D-2D 最小化重投影误差  $e = (u, v) - \text{project}(T_{cw} * P_w)$

Vertex: `g2o::VertexSE3Expmap()`, 即当前帧的`Tcw`

`g2o::VertexSBAPointXYZ()`, `MapPoint`的`mWorldPos`

Edge: `g2o::EdgeSE3ProjectXYZ()`, `BaseBinaryEdge`

Vertex: 待优化当前帧的`Tcw`

Vertex: 待优化`MapPoint`的`mWorldPos`

measurement: `MapPoint`在当前帧中的二维位置  $(u, v)$

InfoMatrix: `invSigma2`(与特征点所在的尺度有关)

## ✓ Map中所有的MapPoints和关键帧做bundle adjustment优化

Global BA优化在ORB-SLAM2中有两个地方使用:

a. 单目初始化: `CreateInitialMapMonocular`函数

b. 闭环优化: `RunGlobalBundleAdjustment`函数

## ✓ LocalBundleAdjustment会在LocalMapping线程处理完队列中最后一个关键帧时进行

# Optimizer.cpp:

## ■ PoseOptimization

3D-2D 最小化重投影误差  $e = (u, v) - \text{project}(\text{Tcw} * \text{Pw})$

只优化Frame的Tcw, 不优化MapPoints的坐标

Vertex: `g2o::VertexSE3Expmap()`, 即当前帧的Tcw

Edge: `g2o::EdgeSE3ProjectXYZOnlyPose()`, `BaseUnaryEdge`

Vertex: 待优化当前帧的Tcw

measurement: MapPoint在当前帧中的二维位置  $(u, v)$

InfoMatrix: `invSigma2`(与特征点所在的尺度有关)

`g2o::EdgeStereoSE3ProjectXYZOnlyPose()`, `BaseUnaryEdge`

Vertex: 待优化当前帧的Tcw

measurement: MapPoint在当前帧中的二维位置  $(u_l, v, u_r)$

InfoMatrix: `invSigma2`(与特征点所在的尺度有关)



# Optimizer.cpp:

## ■ OptimizeEssentialGraph

Vertex: `g2o::VertexSim3Expmap`, Essential graph中关键帧的位姿

Edge: `g2o::EdgeSim3()`, `BaseBinaryEdge`

Vertex: 关键帧的Tcw, MapPoint的Pw

measurement: 经过CorrectLoop函数步骤2, Sim3传播校正后的位姿

InfoMatrix: 单位矩阵

✓ OptimizeEssentialGraph会在成功进行闭环检测后, 全局BA优化前进行

# Optimizer.cpp:

## ■ OptimizeSim3

Vertex: `g2o::VertexSim3Expmap()`, 两个关键帧的位姿  
          `g2o::VertexSBAPointXYZ()`, 两个关键帧共有的MapPoints  
Edge:    `g2o::EdgeSim3ProjectXYZ()`, BaseBinaryEdge  
          Vertex: 关键帧的Sim3, MapPoint的Pw  
          measurement: MapPoint在关键帧中的二维位置(u, v)  
          InfoMatrix: invSigma2(与特征点所在的尺度有关)  
          `g2o::EdgeInverseSim3ProjectXYZ()`, BaseBinaryEdge  
          Vertex: 关键帧的Sim3, MapPoint的Pw  
          measurement: MapPoint在关键帧中的二维位置(u, v)  
          InfoMatrix: invSigma2(与特征点所在的尺度有关)

✓ OptimizeSim3会在筛选闭环候选帧时用于位姿Sim3优化