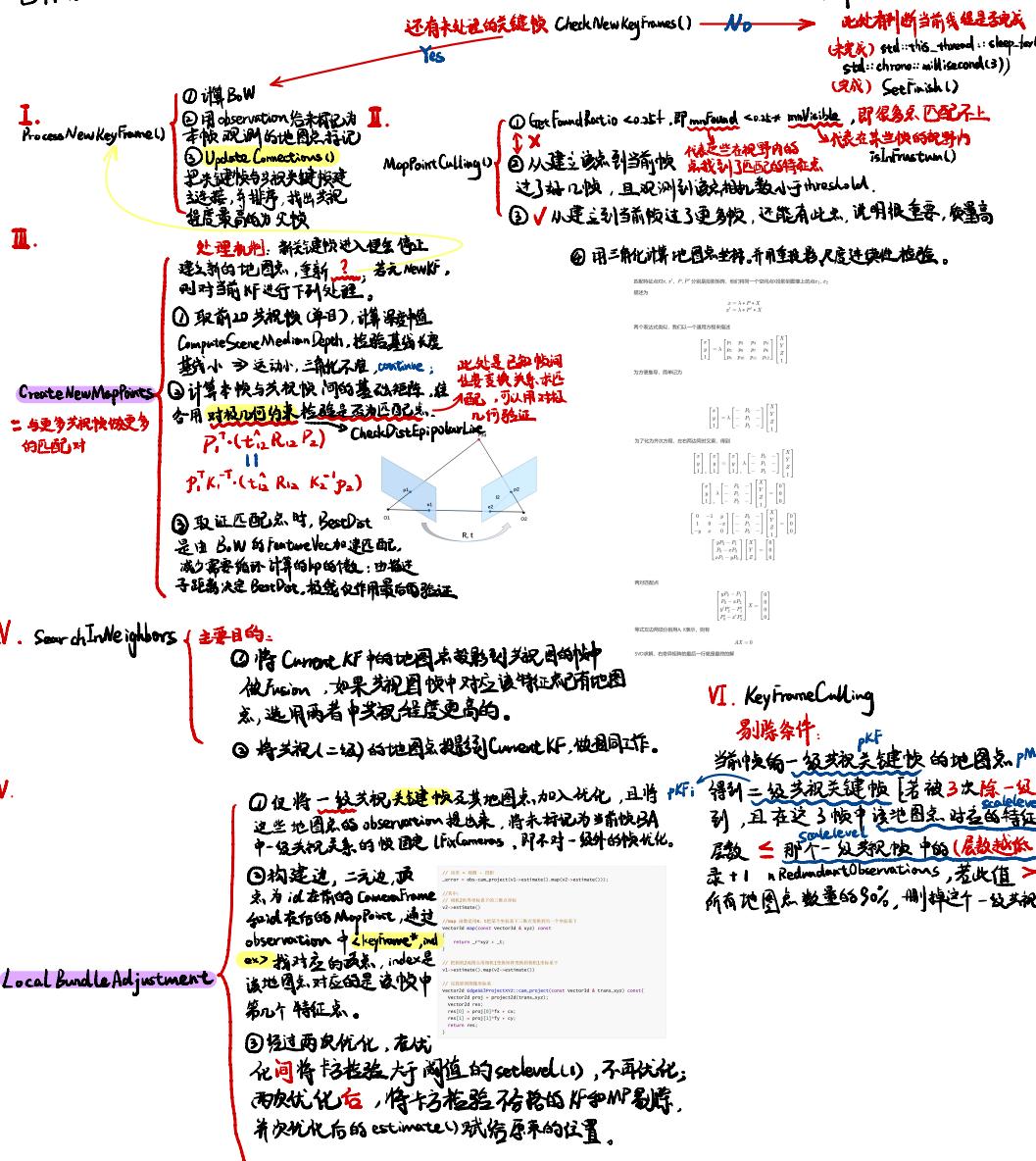


`Run` 是 Local Mapping 中的主函数，流程如下：

### [处理机制]

开始处理关键帧时，关闭接受新关键帧；下述处理结束，会打开接收新关键帧，并等待3毫秒。若未加入新关键帧，会继续处理已有的但还未处理的帧。



## 第6周：ORB\_SLAM2 课程课件

本周课程重点：

1. 局部建图 local mapping
  - CreateNewMapPoints()
  - ComputeF12
  - SearchForTriangulation
  - SearchInNeighbors()
  - 删除地图点和关键帧
  - 地图点法线朝向的计算
  - 共视图 (Covisibility Graph)
  - EdgeSE3ProjectXYZ 误差计算

# 第6周：ORB\_SLAM2 课程课件

本课件是公众号 计算机视觉life 旗下课程 [《全网最详细的ORB-SLAM2精讲：原理推导+逐行代码分析》](#) (点击可跳转课程详情) 的课程课件。谢谢各位学员的支持！

本课程对应的注释代码：[https://github.com/electech6/ORBSLAM2\\_detailed\\_comments](https://github.com/electech6/ORBSLAM2_detailed_comments)

由于源码注释和课件在持续更新，所以：

如视频课程中注释与上述GitHub中有不同，**以GitHub上最新源码为准。**

如视频课程中课件与本课件不同，**以本课件为准。**

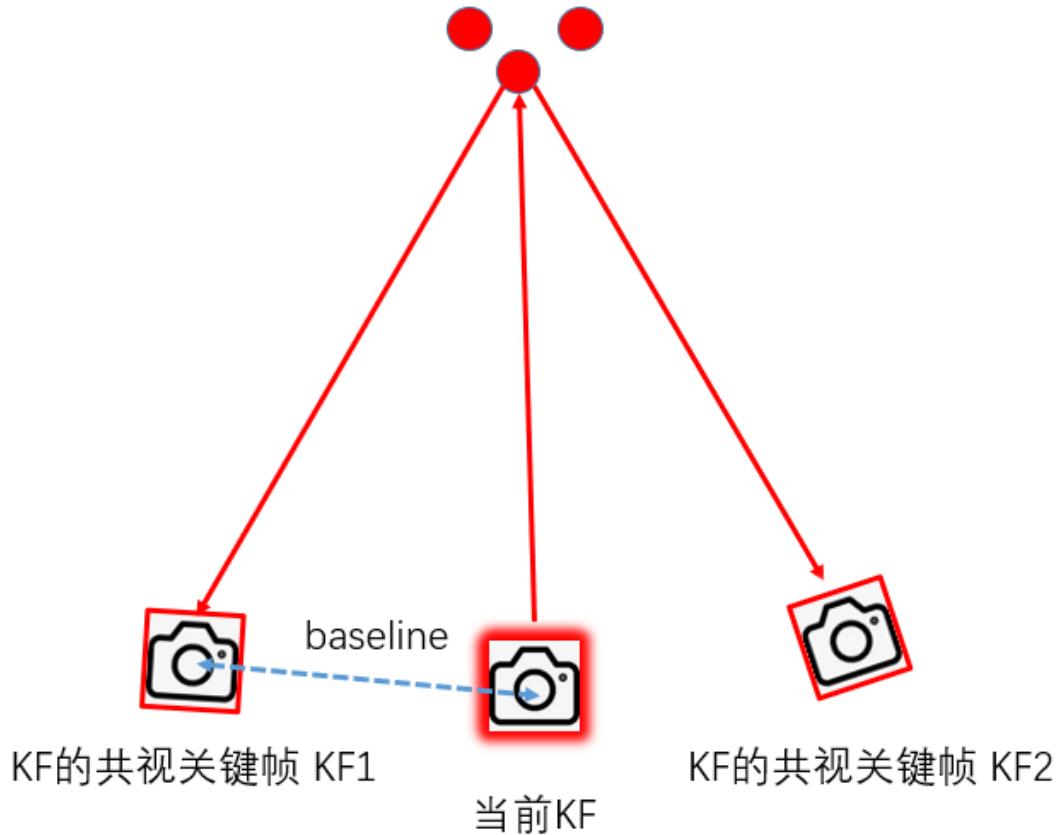
## 本周课程重点：

1. 掌握跟踪线程的整体流程（非常重要）。
2. 掌握局部建图线程如何生成新的地图点。
3. 理解删除冗余关键帧和地图点的原理。
4. 掌握local BA的原理及代码实现（重要）。

## 1. 局部建图 local mapping

### CreateNewMapPoints()

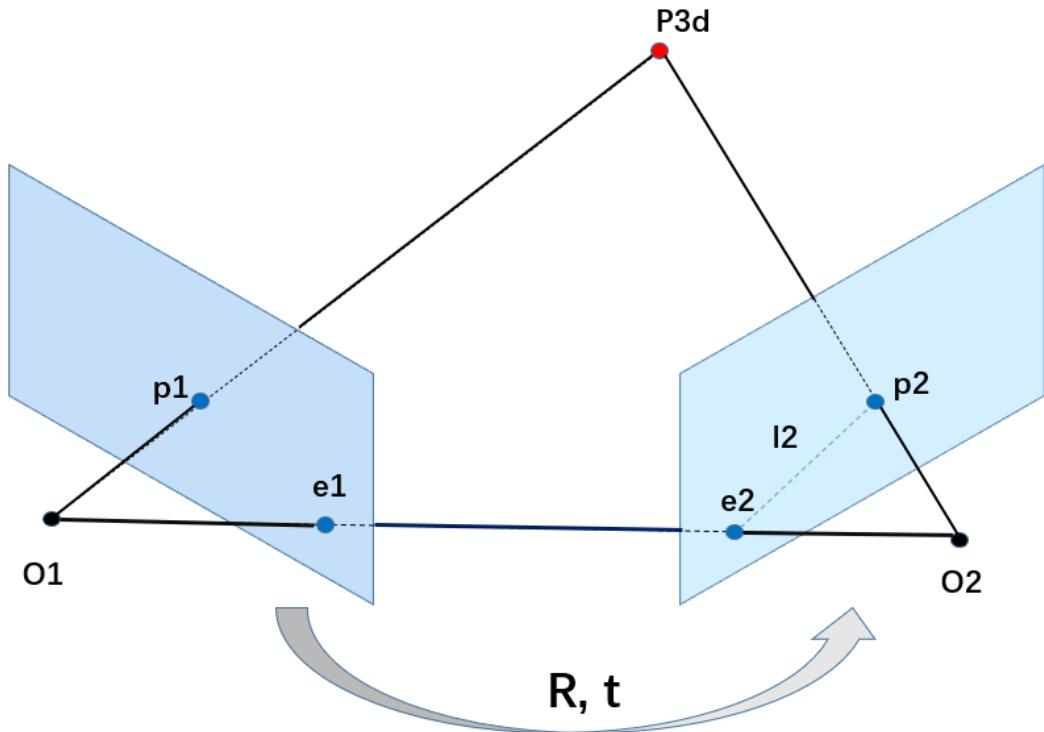
KF的地图点



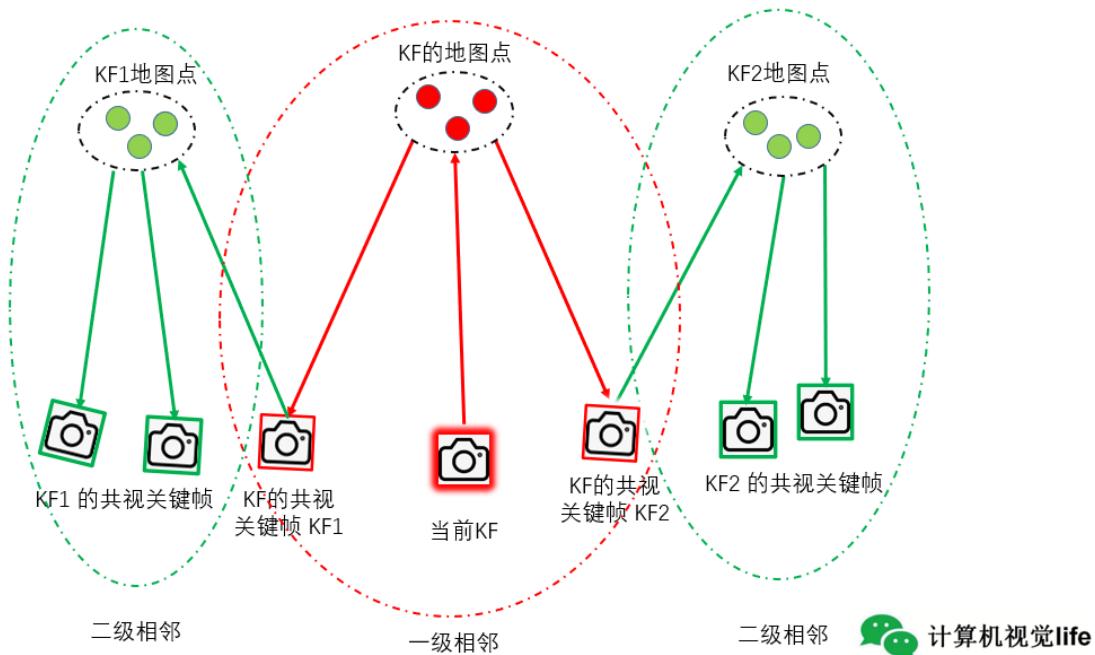
## ComputeF12

$$\begin{aligned}T_{1w} &= \begin{bmatrix} R_{1w} & t_{1w} \\ 0^T & 1 \end{bmatrix} \\T_{2w} &= \begin{bmatrix} R_{2w} & t_{2w} \\ 0^T & 1 \end{bmatrix} \\T_{12} = T_{1w}T_{2w}^{-1} &= \begin{bmatrix} R_{1w} & t_{1w} \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} R_{2w}^T & -R_{2w}^T t_{2w} \\ 0^T & 1 \end{bmatrix} \\&= \begin{bmatrix} R_{1w}R_{2w}^T & -R_{1w}R_{2w}^T t_{2w} + t_{1w} \\ 0^T & 1 \end{bmatrix}\end{aligned}$$

## SearchForTriangulation



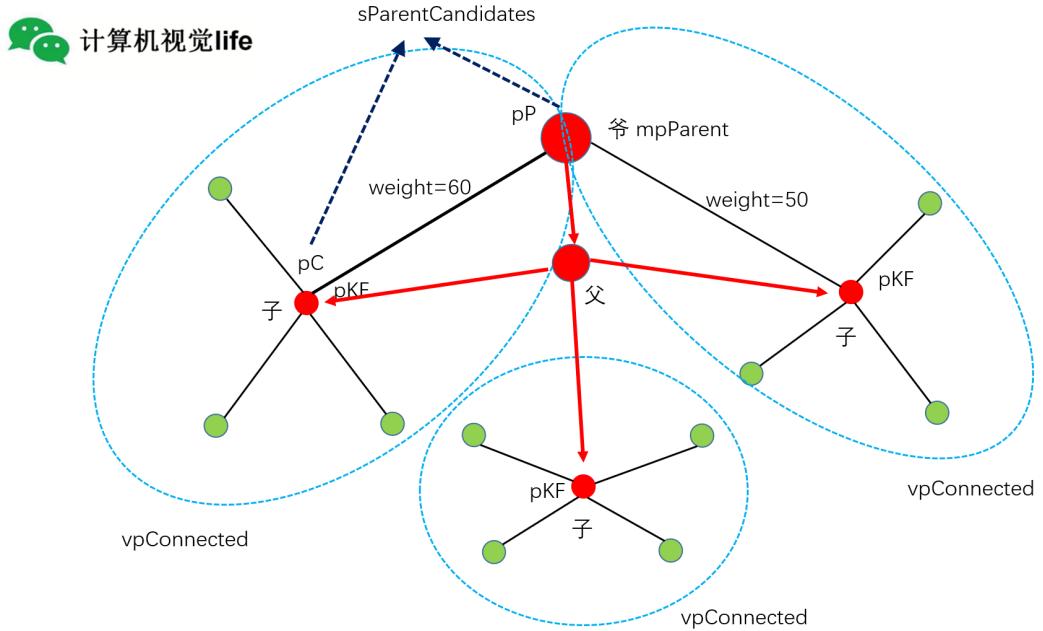
## SearchInNeighbors()



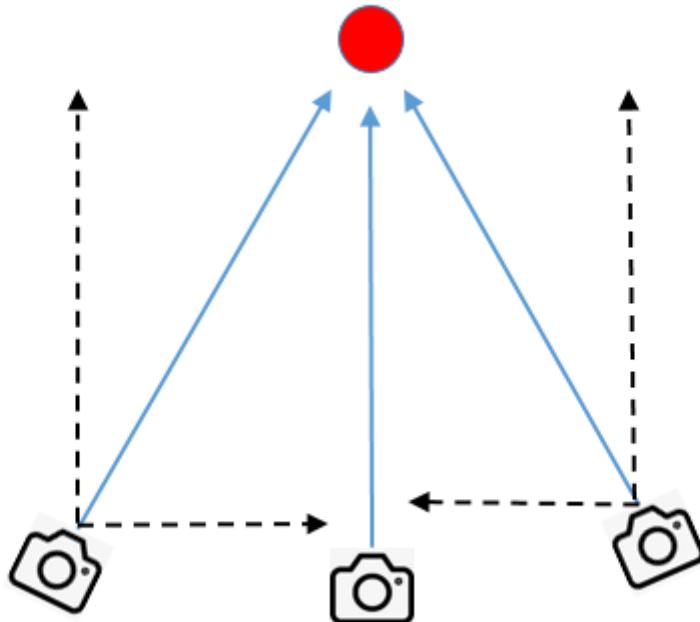
## 删除地图点和关键帧

删除地图点，在`MapPoint::SetBadFlag()` 被标记为需要删除

删除关键帧 `KeyFrame::SetBadFlag()`, 会更新spanning tree

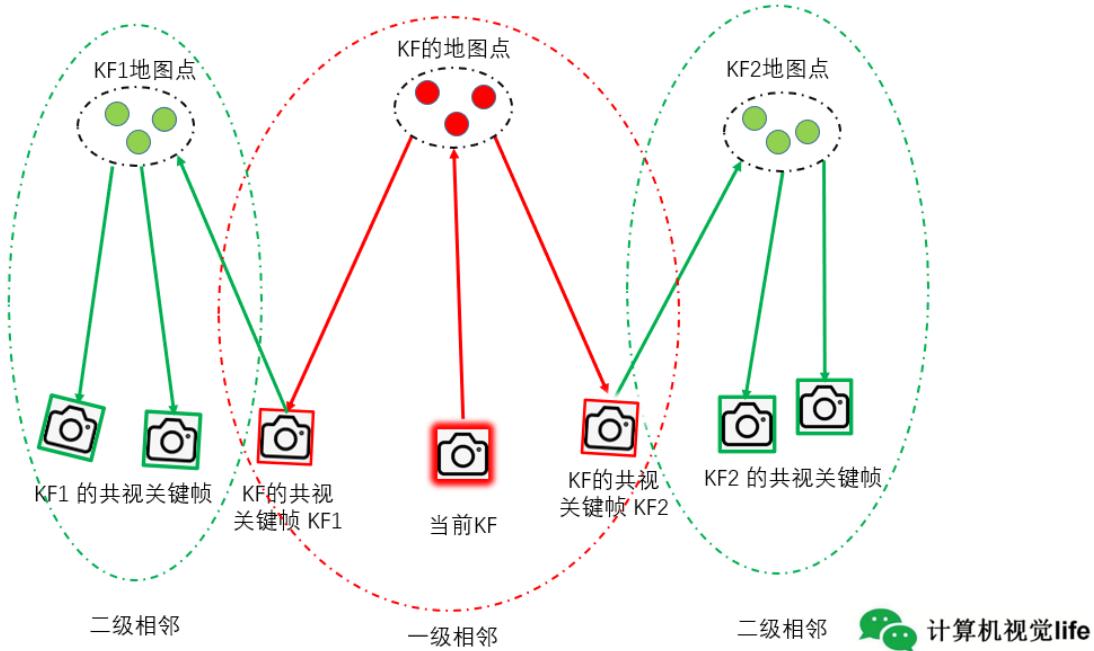


## 地图点法线朝向的计算



## 共视图 (Covisibility Graph)

共视图是无向加权图，每个节点是关键帧，如果两个关键帧之间满足一定的共视关系（至少15个共同观测地图点）他们就连成一条边，边的权重就是共视地图点数目



## EdgeSE3ProjectXYZ 误差计算

```

// 误差 = 观测 - 投影
_error = obs-cam_project(v1->estimate().map(v2->estimate()));

//其中:
// 相机2世界坐标系下的三维点坐标
v2->estimate()

//map 函数是用R, t把某个坐标系下三维点变换到另一个坐标系下
vector3d map(const Vector3d & xyz) const
{
    return _r*xyz + _t;
}

// 把相机2地图点用相机1变换矩阵变换到相机1坐标系下
v1->estimate().map(v2->estimate())

// 反投影到图像坐标系
vector2d EdgeSE3ProjectXYZ::cam_project(const Vector3d & trans_xyz) const{
    vector2d proj = project2d(trans_xyz);
    Vector2d res;
    res[0] = proj[0]*fx + cx;
    res[1] = proj[1]*fy + cy;
    return res;
}

```

[《全网最详细的ORB-SLAM2精讲：原理推导+逐行代码分析》](#)（点击可跳转课程详情）