`CreateNewMapPoints`函数

数据来源：与当前关键帧共视程度最高的相邻关键帧；

更新的信息：成功三角化的3D点，添加到成员变量`mpMap`局部地图句柄。另外将生成的3D点添加到待检验的地图点队列中，进行地图点检验。

Step 1 获取数据

1、在处理新的关键帧函数中更新连接关系函数已经将与当前关键帧具有共视关系的关键帧进行排序，因此，此处只需要获取权值最高的几组关键帧数据即可。

2、获取当前关键帧的变换矩阵：由旋转矩阵和平移矩阵组成

3、获取相机中心

计算世界坐标系到相机坐标系的平移向量：

$$R_{cw} \cdot P_w + t_{cw} = P_c$$
$$P_w = R_{cw}^{-1} \cdot P_c - R_{cw}^{-1} \cdot t_{cw}$$

$$P_w = T_{wc} \cdot P_c = R_{wc} \cdot P_c + t_{wc}$$

当$P_c = [0,0,0]^T$时，上下两式相等，可以等到世界坐标到相机坐标的平移向量$t_{wc} = -R_{cw}^{-1} \cdot t_{cw}$同时由于旋转矩阵 R_{cw}与R_{wc}是反对称矩阵，因此其转置就是求逆。

Step 2 遍历每个关键帧

计算当前关键帧与相邻的每个关键帧之间通过三角化形成的地图点。

判断是否需要对每个相邻的关键进行三角化，需要满足一定的稀疏性。

双目：判断两帧图像之间的基线与双目相机本身的基线的大小关系？

单目：两帧图像之间的基线与相邻关键帧的景深的比例？

利用当前帧特征点对应的深度信息的中值近似为景深：

$$P_c = R_{cw} \cdot P_w + T_{cw}$$

只计算P_c对应的深度信息，取对应的向量进行计算即可。

Step 3 计算当前关键帧与相邻关键帧的基本矩阵

对极几何计算基本矩阵(Fundamental Matrix)：

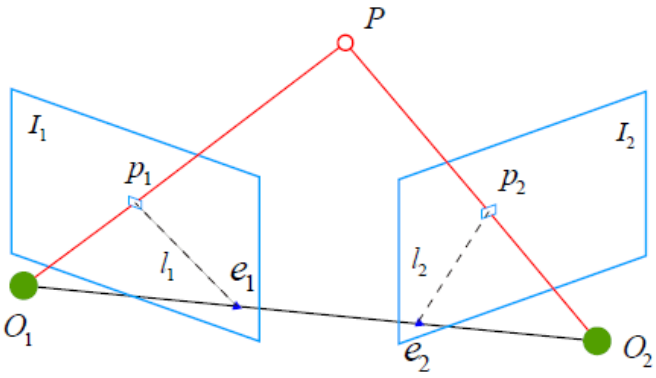


图 7-7　对极几何约束。

假设空间中点P的空间位置为：$P = [X, Y, Z]^T$

两个像素点p_1, p_2的像素位置为：

$$s_1 p_1 = K_1(R1_{cw} P_w + T1_{cw})$$
$$s_2 p_2 = K_2(R2_{cw} P_w + T2_{cw})$$

如果使用齐次坐标，上式在乘以非零常数下仍然成立。

$$p_1 = K_1(R1_{cw} P_w + T1_{cw})$$
$$p_2 = K_2(R2_{cw} P_w + T2_{cw})$$

假设x_1, x_2是两个像素点的归一化平面上的坐标。

$$x_1 = K_1^{-1} p_1 = R1_{cw} P_w + T1_{cw}$$
$$x_2 = K_2^{-1} p_2 = R2_{cw} P_w + T2_{cw}$$
$$=> x_2 = R2_{cw} R_{cw}^{-1} x_1 - R2_{cw} R1_{cw}^{-1} T1_{cw} + T2_{cw}$$

假设：$R21_{cw} = R2_{cw} R_{cw}^{-1}$
$T21_{cw} = -R2_{cw} R1_{cw}^{-1} T1_{cw} + T2_{cw}$

$$x_2 = R21_{cw} x_1 + T21_{cw}$$

两侧同时与$T21_{cw}$做外积：

$$T21_{cw}^{\wedge} x_2 = T21_{cw}^{\wedge} R21_{cw} x_1$$

两侧同时左乘x_2^T：

$$x_2^T T21_{cw}^{\wedge} x_2 = x_2^T T21_{cw}^{\wedge} R21_{cw} x_1$$

等式左侧，$T21_{cw}^{\wedge} x_2$与$T21_{cw}$和x_2都垂直的向量。因此等式左侧等于0。

$$x_2^T T21_{cw}^{\wedge} R21_{cw} x_1 = 0$$

重新代入p_1, p_2：

$$p_2^T K_2^{-T} T21_{cw}^{\wedge} R21_{cw} K_1^{-1} p_1 = 0$$

这个关系式就称为对极约束。

基础矩阵：$F = K_2^{-T} T21_{cw}^{\wedge} R21_{cw} K_1^{-1}$

Step 4 通过极线约束限制匹配时的搜索范围，进行特征点匹配，但不检查旋转

(1) 计算得到关键帧1的相机中心在关键帧2中对应的像素坐标

$$关键帧1的相机中心在关键帧2的相机坐标系下的坐标：O2_c=[X,Y,Z]^T$$

$$O2_c=R2_{cw}O1_w+t2_{cw}$$

根据针孔相机模型，计算相机坐标下的坐标对应的像素坐标：

$$u=f_x\frac{X}{Z}+c_x$$

$$v=f_y\frac{Y}{Z}+c_y$$

(2) 利用ORB词典加速未被跟踪的特征点匹配

只要pKF或F帧有一个FeatureVector的迭代器遍历到了map容器尾部，就会停止整个循环（同结点查找）

- 第一个if判断如果成立，说明两个迭代器同时访问到了相同的结点，那么就会进入匹配的主程序
- 如果两个迭代器暂时没有访问到相同的结点，就是进行迭代器递增的操作，但是要分成两个情况
- 情况一：如果KFit迭代器访问的结点小于Fit访问的结点，那么KFit就按顺序递增到第一个比Fit访问的结点ID大于或等于的结点上
- 情况二：如果KFit迭代器访问的结点大于于Fit访问的结点，那么Fit就按顺序递增到第一个比KFit访问的结点ID大于或等于的结点上
- 其中**lower_bound()** 函数的功能就是从数组的begin位置到end-1位置二分查找第一个大于或等于num的数字，找到返回该数字的地址，不存在则返回end。通过返回的地址减去起始地址begin,得到找到数字在数组中的下标。

`lower_bound()`和`upper_bound()`函数：利用二分查找的方法在一个排好序的数组中进行查找的。

在从小到大的排序数组中：

`lower_bound( begin,end,num)`：从数组的begin位置到end-1位置二分查找第一个大于或等于num的数字，找到返回该数字的地址，不存在则返回end。通过返回的地址减去起始地址begin,得到找到数字在数组中的下标。

`upper_bound( begin,end,num)`：从数组的begin位置到end-1位置二分查找第一个大于num的数字，找到返回该数字的地址，不存在则返回end。通过返回的地址减去起始地址begin,得到找到数字在数组中的下标。

在从大到小的排序数组中，重载`lower_bound()`和`upper_bound()`

`lower_bound( begin,end,num,greater<type>())`:从数组的begin位置到end-1位置二分查找第一个小于或等于num的数字，找到返回该数字的地址，不存在则返回end。通过返回的地址减去起始地址begin,得到找到数字在数组中的下标。

`upper_bound( begin,end,num,greater<type>())`:从数组的begin位置到end-1位置二分查找第一个小于num的数字，找到返回该数字的地址，不存在则返回end。通过返回的地址减去起始地址begin,得到找到数字在数组中的下标。

相同节点下，进行特征匹配，仅仅对两个关键帧中未匹配的特征点进行再次匹配：

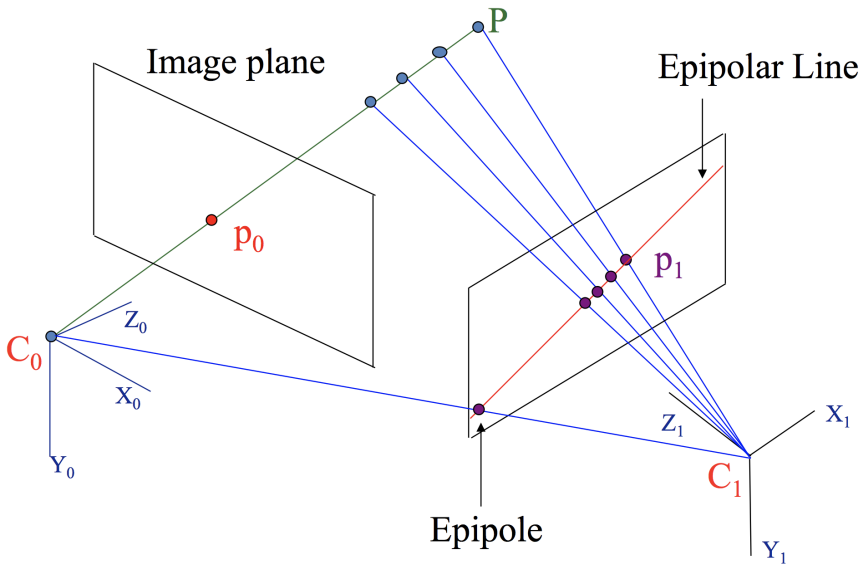
(3) 计算两个特征点的描述子距离

Hamming distance：两个二进制串之间的汉明距离，指的是其不同位数的个数。

高翔SLAM十四讲第二版P164页有关于特征匹配的加速实现方法，利用了SSE指令集中的`_mm_popcnt_u32`函数计算一个`unsigned int`变量中1的个数。

(4)检验是否满足对极约束

基础矩阵：F，两个特征点在各自图像中的像素坐标：p_0和p_1，对极约束为：$p_1^T F p_0 = 0$。



特征点p_0在C_1相机对应的关键帧上的极线方程为：

$$[a,b,c]^T = Fp_0$$

极线方程：$ax + by + c = 0$

$[x,y]$为C_1相机对应的关键帧上的特征点的像素坐标。

p_1到极线方程的距离：

$$dist = \frac{|au + bv + c|}{sqrt(a^2 + b^2)}$$

(5)统计特征点对之间的旋转情况

目的：提供旋转不变性

需要理解`cv::KeyPoint`类：

- pt(x,y):关键点的点坐标。
- angle：角度，表示关键点的方向，通过Lowe大神的论文可以知道，为了保证方向不变形，SIFT算法通过对关键点周围邻域进行梯度运算，求得该点方向。-1为初值。
- octacv:从哪一层金字塔得到的此关键点。

Step 5 对每对匹配通过三角化生成3D点，类似Triangulate函数

(1) 计算视差角

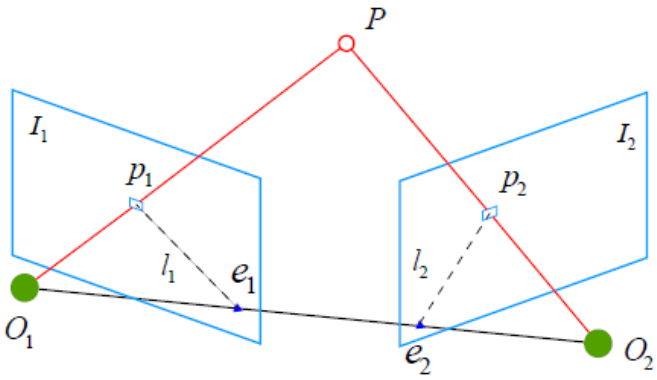


图 7-7　对极几何约束。

实际是计算O_1P以及O_2P在世界坐标系下的方向向量(斜率)。

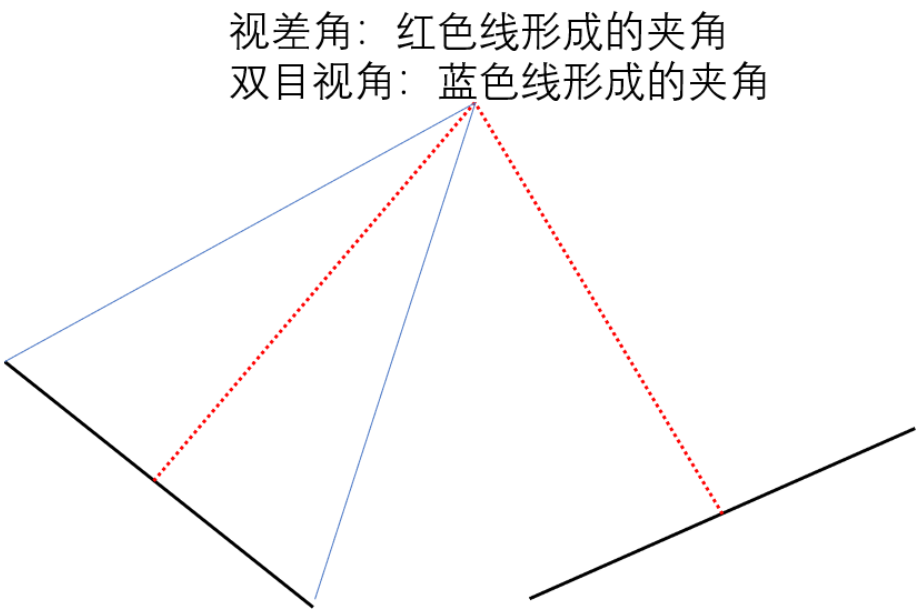
O_1和O_2为两个关键帧对应相机中心在世界坐标系下的坐标，为$O=t_{wc}$。

由于$P_w=R_{wc}P_c+t_{wc}$，所以O_1P的方向向量$P_w-O=P_w-t_{wc}=R_{wc}P_c$。

视角差的余弦值为$\langle K_{O_1P},K_{O_2P}\rangle/(\pmod{K_{O_1P}}\pmod{K_{O_2P}})$。

单目相机必须经过三角化才能恢复3D点；

双目相机需要判断视角差与双目基线和对应的深度信息得到的开角的大小关系：



当视差角大于双目视角时，采用双目恢复3D点，即利用双目的深度信息；

当视差角小于双目视角时，采用三角化方法恢复3D点。

(2) 三角化

先理解叉乘：

向量[a, b, c]的叉乘相当于将此向量扩展为一个反对称矩阵：$\begin{vmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{vmatrix}$

两个向量的叉乘表示为两个向量形成的法向量。

假设某空间点P，其齐次坐标为$P=(X,Y,Z,1)^T$。在图像I_1中，投影到特征点$x_1=(u_1,v_1,1)^T$(以归一化平面齐次坐标表示)。相机位姿为R,t，定义增广矩阵$[R|t]$为一个3x4的矩阵，包含旋转和平移信息，展开如下：

$$s[u_1,v_1,1]^T=\begin{bmatrix} t_1 & t_2 & t_3 & t_4 \\ t_5 & t_6 & t_7 & t_8 \\ t_9 & t_{10} & t_{11} & t_{12} \end{bmatrix}[X,Y,Z,1]^T$$

下列解法来自高翔：

用最后一行把s消去，得到两个约束：

$$u_1=\frac{t_1X+t_2Y+t_3Z+t_4}{t_9X+t_{10}Y+t_{11}Z+t_{12}}$$
$$u_2=\frac{t_5X+t_6Y+t_7Z+t_8}{t_9X+t_{10}Y+t_{11}Z+t_{12}}$$

定义T的行向量为$[t_1',t_2',t_3']^T$，得到

$$t_1^TP-t_3^TPu_1=0$$
$$t_2^TP-t_3^TPv_1=0$$

对图像I_1，满足同样的约束。

$$\begin{bmatrix} t_1'^T - t_3'^T u_1 \\ t_2'^T - t_3'^T v_1 \\ t_1''^T - t_3''^T u_2 \\ t_2''^T - t_3''^T v_2 \end{bmatrix} P = AP = 0 \text{公式一}$$

对矩阵A进行SVD分解，可得到P的解。

解法2：

$$x_1 = \frac{1}{s}[t_1', t_2', t_3']^T P$$

两边同时左叉乘x_1，

$$x_1^\wedge x_1 = \frac{1}{s} x_1^\wedge [t_1', t_2', t_3']^T P = 0$$

$$\begin{bmatrix} v_1 t_3' - t_2' \\ t_1' - u_1 t_3' \\ u_1 t_2' - v_1 t_1' \end{bmatrix} P = 0$$

同样能够得到公式一；

由于利用SVD求的解不一定为归一化坐标，因此需要转化为归一化坐标，并前三维作为其3D信息。

(3) 双目反投影得到3D信息

根据针孔相机模型，利用深度和像素坐标求出对应的相机坐标，然后利用变换矩阵，求出世界坐标3D信息。

(4) 选择正确的解

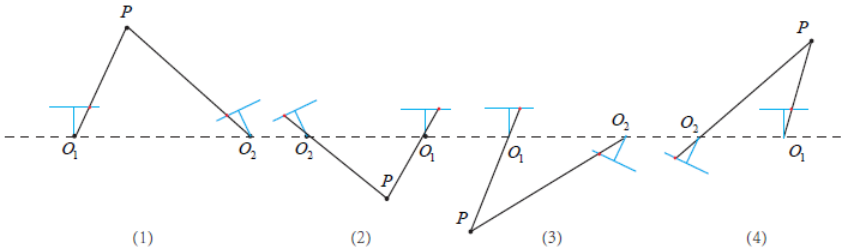


图 7-8　分解本质矩阵得到的四个解。在保持投影点（红点）不变的情况下，两个相机以及空间点一共有四种可能的情况。

利用求取的3D点信息，通过变换矩阵计算深度信息，通过深度信息的正负，判断3D点是否在相机前方。

(5) 重投影误差

1、将世界坐标3D信息转换为相机坐标；

2、将相机坐标转换为像素坐标，并和特征点对应的像素坐标做差，得到像素偏差。单目相机：2自由度的像素偏差（u, v）；双目相机：3自由度的像素偏差（u_l, v_l, u_r）；双目视差公式$d = u_L - u_r = \frac{fb}{z}$。

(6) 检查尺度连续性

（不太理解）