# 2 ORB 特征点(4 分，约2.5 小时)

## 自行书写ORB 的提取、描述子的计算以及匹配的代码

计算角度

A screenshot of a cell phone

Description automatically generated

计算描述子

A screenshot of a cell phone

Description automatically generated

暴力匹配



特征点筛选结果

A picture containing drawing

Description automatically generated

匹配结果

A picture containing table

Description automatically generated

## 最后，请结合实验，回答下面几个问题：

### 为什么说ORB 是一种二进制特征？

因为在ORB中我们一般用128/256位二进制数来描述特征点的特征。

### 为什么在匹配时使用50 作为阈值，取更大或更小值会怎么样？

因为每一个点都会找到一个对应的匹配，如果不设置阈值即便是256这样的“远”的距离也可能被作为最佳匹配输出。取更大的阈值会找到更多的匹配则更有可能输出错误匹配，更小的阈值则获得的匹配对数量可能更少。

### 3. 暴力匹配在你的机器上表现如何？你能想到什么减少计算量的匹配方法吗？

暴力匹配每次都要独立的比较P某个点和Q中所有点的距离，明显要等一下才出现结果。

可以考虑用树来表示两组点，比较树结构中的上层节点，而不需要遍历每个叶子节点。

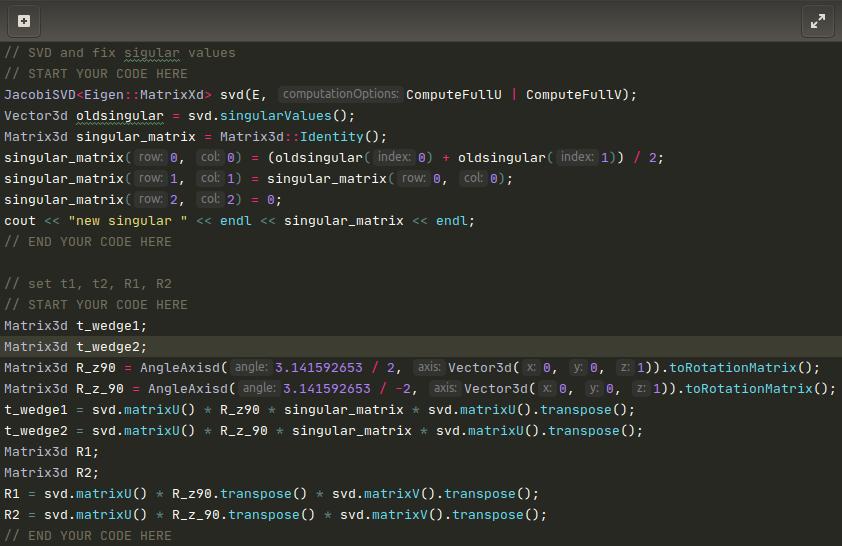
亦或者利用机器学习的方法离线训练匹配算法。

也可以根据运动模型，预测每个特征点在下一帧图像的位置，在那个预测位置附近寻找匹配。

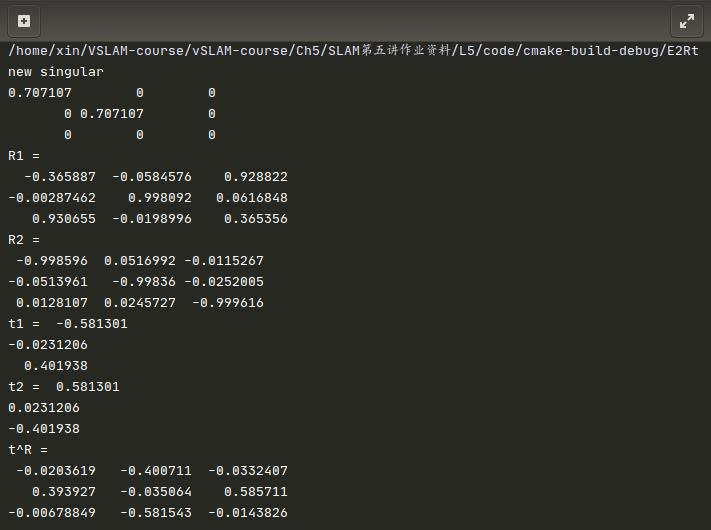
# 从E 恢复R*;* t (3 分，约1 小时)

代码，包括：SVD分解，奇异值重构成（sig1+sig2/2, sig1+sig2/2, 0）的形式。

从UV，singular和R\_z(90), R\_z(-90)中恢复t和R



结果满足t^R=E

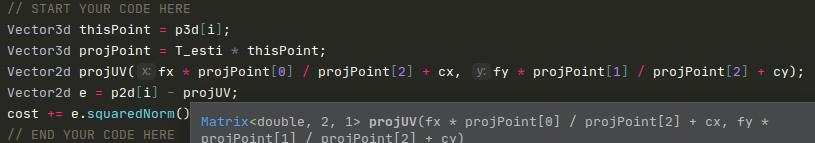


# 4 用G-N 实现Bundle Adjustment 中的位姿估计(3 分，约2 小时)

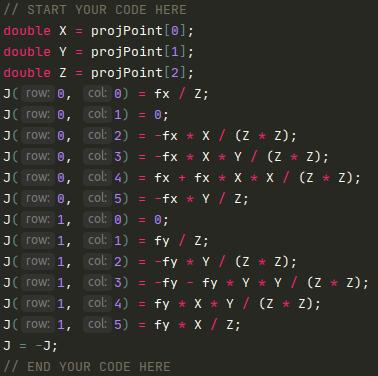
## 在书写程序过程中，回答下列问题：

### 如何定义重投影误差？

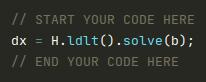
Error = 观测 – 投影。

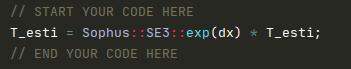


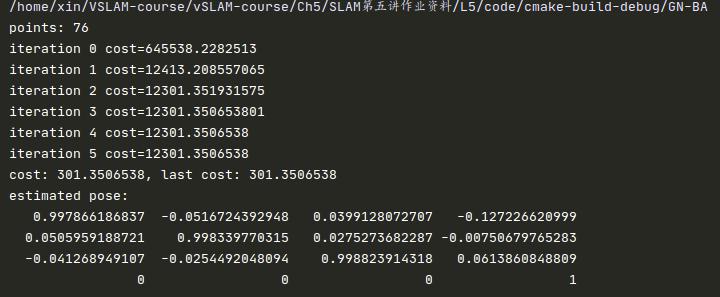
### 该误差关于自变量的雅可比矩阵是什么？



### 解出更新量之后，如何更新至之前的估计上？

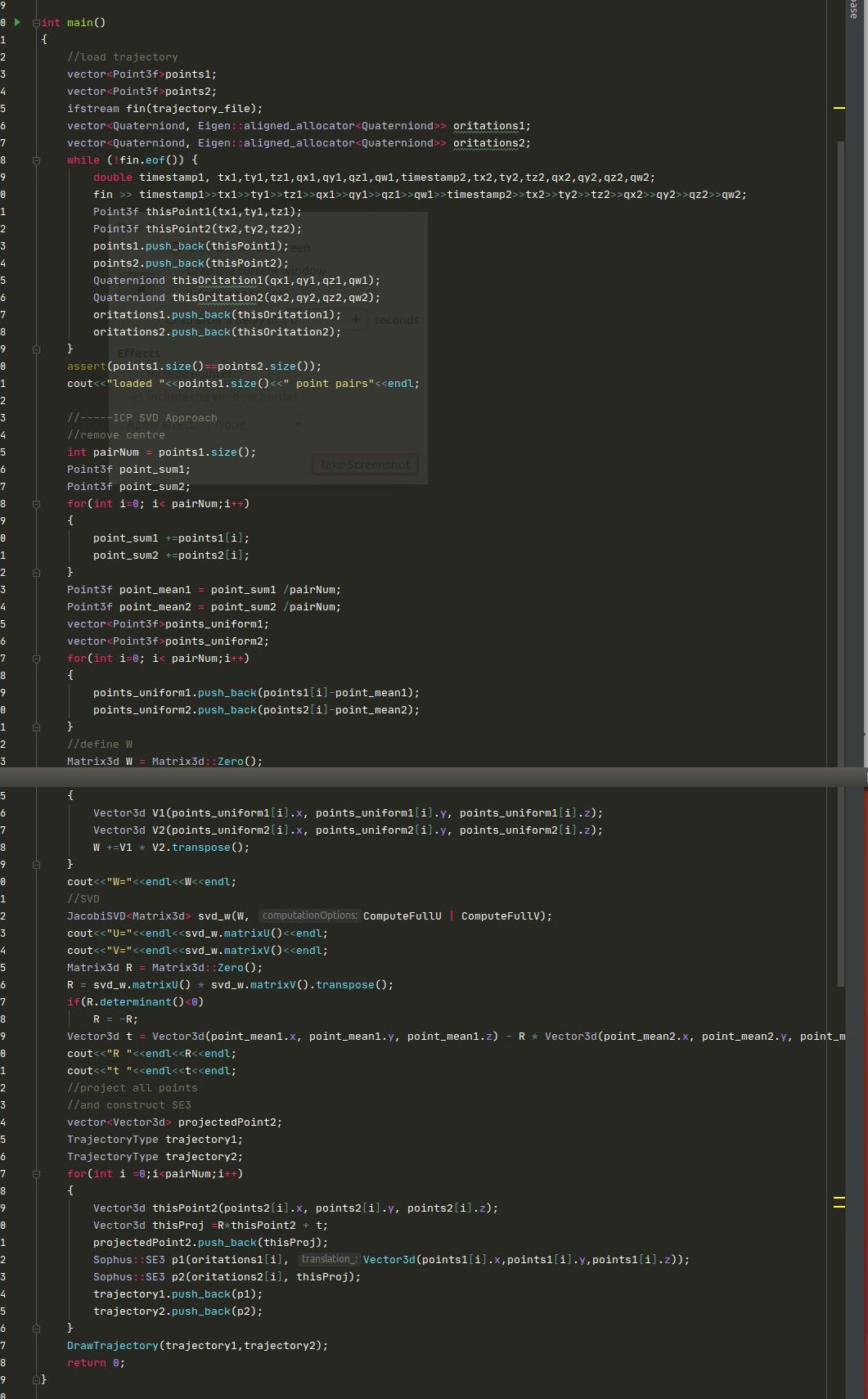


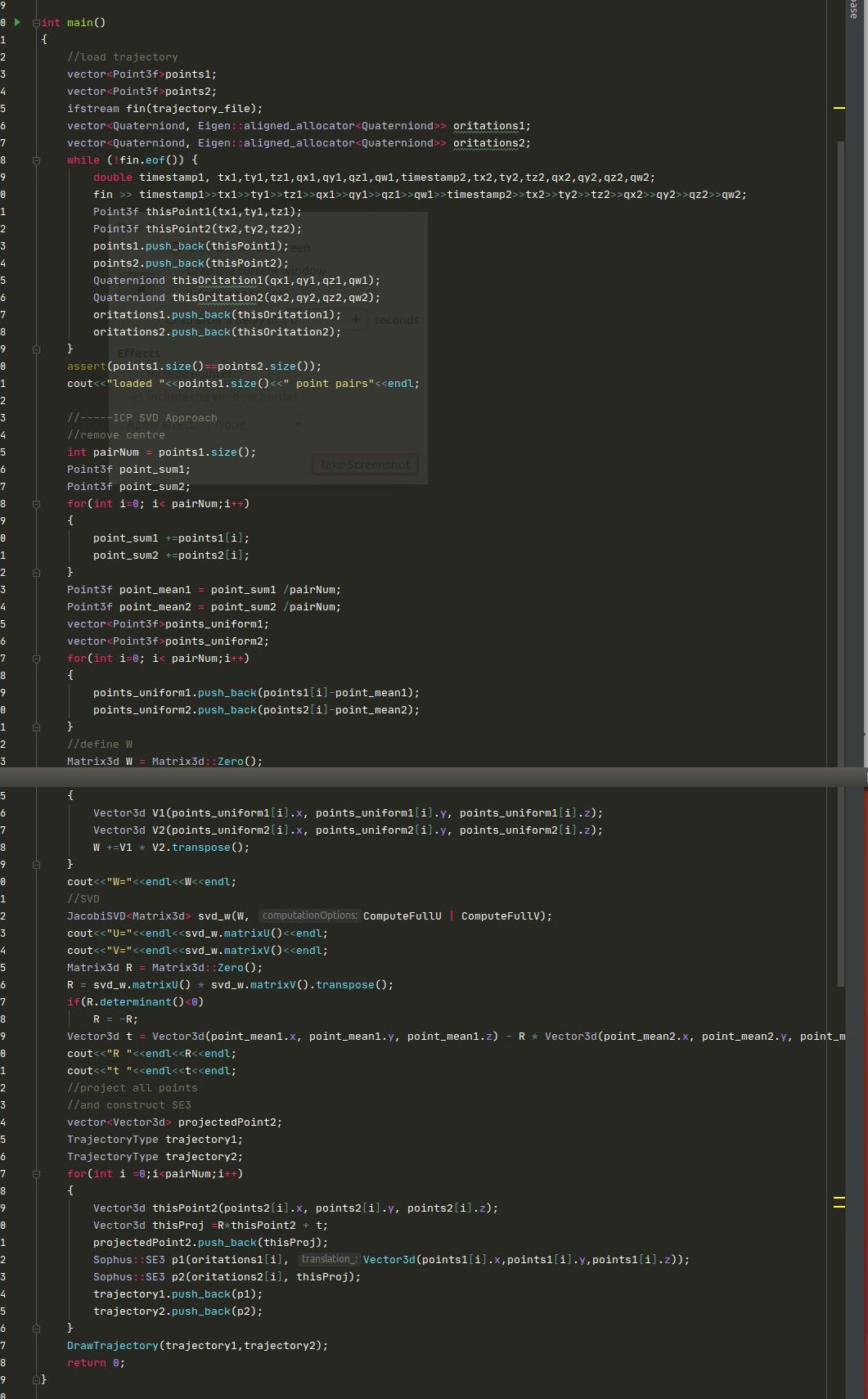


最后的结果跟题干的T接近

# 5 \* 用 ICP 实现轨迹对齐 (2 分，约 2 小时)

代码如下，分为读取轨迹，去重心，构建W，从SVD恢复Rt，投影第一条轨迹，构建成se3的形式。调用之前的drawjractory函数显示（没修改此函数所以截图省略）





投影到一起显示

