手写 VIO ch6

allenthree **Homework 6** 2021 年 10 月 23 日

一. 证明题

证明式 (15) 中，取 $y = u_4$ 是该问题的最优解。

答:

目标函数：$\min\limits_{y} \|Dy\|_2^2 \qquad s.t. \|y\| = 1$

证明:

通过拉格朗日乘子法，将约束加入目标函数中得到:

$F(y) = y^\top D^\top Dy - \lambda(y^\top y - 1)$

令 $F'(y) = 0$ 有:

$D^\top Dy - \lambda y = 0$

$D^\top Dy = \lambda y$

将上式带入目标函数 $F(x)$ 中，用 λy 替换 $D^\top Dy$ 得到:

$F(y) = \lambda y^\top y - \lambda(y^\top y - 1) = \lambda$

因此要求目标函数的最小值，就是求 λ 的最小值，就是要求最小的矩阵特征值，然后要求的向量 $\mathbf{y}$(点的坐标) 就是最小的矩阵特征值所对应的特征向量。

因为投影矩阵 $P_k = [R_k, t_k] \in \mathbb{R}^{3\times 4}$ (PPT 中将这个矩阵记为 World 坐标系到 Camera 坐标系的投影矩阵)。然后据此推导出来的后面的 D 也就是一个 $2n \times 4$ 的矩阵。所有进行奇异值分解也就最多得到 4 个奇异值。然后选择一个最小的奇异值 σ_4，这个奇异值对应的特征向量就是 u_4。

证毕。

二. Section 2

题目：根据本节课的公式，完成特征点三角化代码，并通过仿真测试

答：代码中添加的部分主要是对 D 矩阵进行构建，并通过 SVD 对 $D^\top D$ 进行奇异值求解，然后拿最小的奇异值对应的特征向量作为 landmark 坐标。

```
/* your code begin */
const auto D_rows = 2*(end_frame_id-start_frame_id);
Eigen::MatrixXd D(Eigen::MatrixXd::Zero(D_rows, 4));
```

```
// 构建矩阵 D
for(auto i=start_frame_id;i<end_frame_id;++i)
{
    // 我对这个下标的含义一直有点模糊，感觉好像是跟机器人学里面的不太一样
    // 可能机械臂那些更多是考虑机械臂末端相对世界坐标系
    // 而VIO则是考虑通过路标点得到camera 和 landmarks 在世界坐标系中的位置
    Eigen::Matrix3d Rcw = camera_pose[i].Rwc.transpose();
    Eigen::Vector3d tcw = -Rcw *camera_pose[i].twc;
    D.block(2*(i-start_frame_id), 0,1,3).noalias() =    // u(像素坐标)*R3行- R1行
            camera_pose[i].uv(0)*Rcw.block(2,0,1,3)-Rcw.block(0,0,1,3);
    D.block(2*(i-start_frame_id),3,1,1).noalias() =    // 最后一列是 translation
            camera_pose[i].uv(0)*tcw.segment(2,1)-tcw.segment(0,1);
    D.block(2*(i-start_frame_id)+1, 0,1,3).noalias() =
            camera_pose[i].uv(1)*Rcw.block(2,0,1,3)-Rcw.block(1,0,1,3);
    D.block(2*(i-start_frame_id)+1,3,1,1).noalias() =
            camera_pose[i].uv(1)*tcw.segment(2,1) - tcw.segment(1,1);
}
// 对矩阵DTD进行SVD特征值分解
Eigen::JacobiSVD<Eigen::MatrixXd>
        svd(D.transpose()*D,Eigen::ComputeThinU| Eigen::ComputeThinV);
// 将四个特征值存入向量中
Eigen::Vector4d lambda = svd.singularValues();
if(lambda(2)/lambda(3)<1e-3) return -1;        // 比较sigma3 和 sigma4 的大小
// SVD分解得到的结果 U 矩阵，取最后一列 u4 as we just proved
Eigen::Vector4d u4 = svd.matrixU().block(0,3,4,1);
if(u4(3)!=0 && u4(2)/u4(3)>0){
    P_est(0) = u4(0)/u4(3);
    P_est(1) = u4(1)/u4(3);
    P_est(2) = u4(2)/u4(3);
}
/* your code end */
```

程序运行结果如下，可以看到通过 SVD 求出来的 landmark 坐标与真实值是一样的。

```
./estimate_depth
ground truth:
  -2.9477 -0.330799  8.43792
your  result :
  -2.9477 -0.330799  8.43792
```

三．测试不同噪声和观测对奇异值后两位的比值产生的影响

问题 3.1: 对测量值加上不同噪声 (增大测量噪声方差)，观察奇异值后两位的比值变化。

答：直接在将 landmark 投影到像素坐标系的 for 循环中，对 U,V 添加随机噪声，模拟实际中相机像素上产生的噪声量，添加噪声的代码如下：

```
Eigen::Vector3d Pw(tx, ty, tz);
// 这个特征从第三帧相机开始被观测，i=3
int start_frame_id = 3;
int end_frame_id = poseNums;
for (int i = start_frame_id; i < end_frame_id; ++i) {
    Eigen::Matrix3d Rcw = camera_pose[i].Rwc.transpose();
    Eigen::Vector3d Pc = Rcw * (Pw - camera_pose[i].twc);
    double x = Pc.x();
    double y = Pc.y();
    double z = Pc.z();
    const double mean = 0.0;          // 均值
    const double stddev = 0.01;        // 标准差 standard deviation
    std::normal_distribution<double> noise(mean,stddev);
    Pc = Pc/Pc[2];
    Pc[0] += noise(generator);        // 生成一个随机的误差
    Pc[1] += noise(generator);
    camera_pose[i].uv = Eigen::Vector2d(Pc[0], Pc[1]);
}
```

现在程序中的噪声标准差为 0.01，然后对其进行修改，以得到不同的运行结果。奇异值最后两位的比值变化情况如图 1 所示：

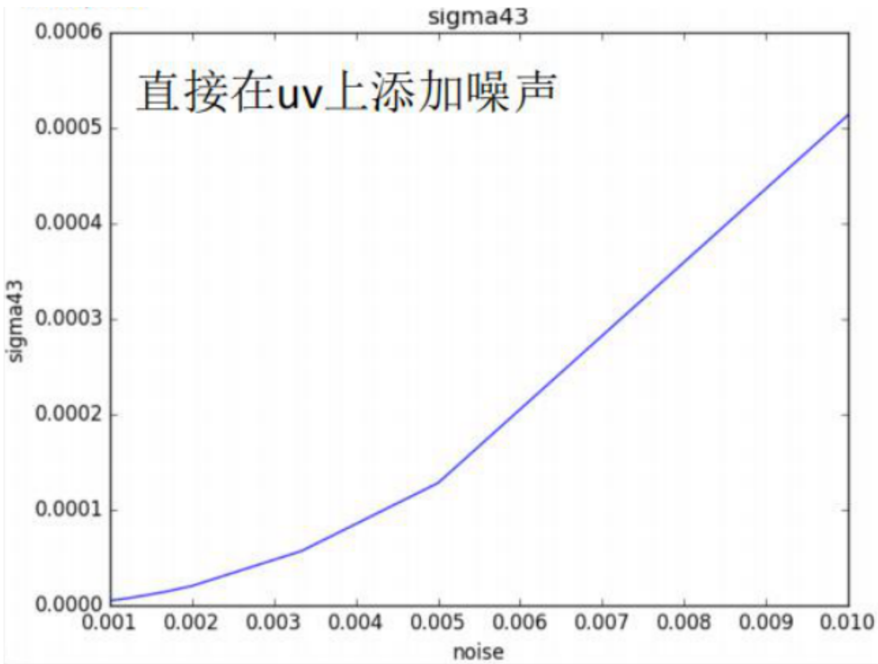

图 1: ratio vary with sigma

从图中可以看出，随着像素误差的标准差的增大，奇异值 (其实应该不是奇异值 sigma，而是特征值 lambda 吧，whatever lambda = sigma * sigma) 后两位的比值逐渐增大，同时对 landmark 的三角化测量结果也与 ground truth 相差较大。(不过当 sigma 设为 0.1 或者更大的时候，特征点的位姿估计 error 就相当大了，会有好几个 1 的变化，不知道实际情况下这种 sigma 有多大，难道只有 0.01 个像素吗，我不相信，-_-)

问题 3.2: 固定噪声方差参数，将观测图像帧扩成多帧，观测最小奇异值和第二小奇异值之间的比例变化规律。

答：这里只需要固定噪声的标准差，然后观测图像帧的帧数修改就可以了。最后的结果整理 (其实是抄的) 如图 2 所示：

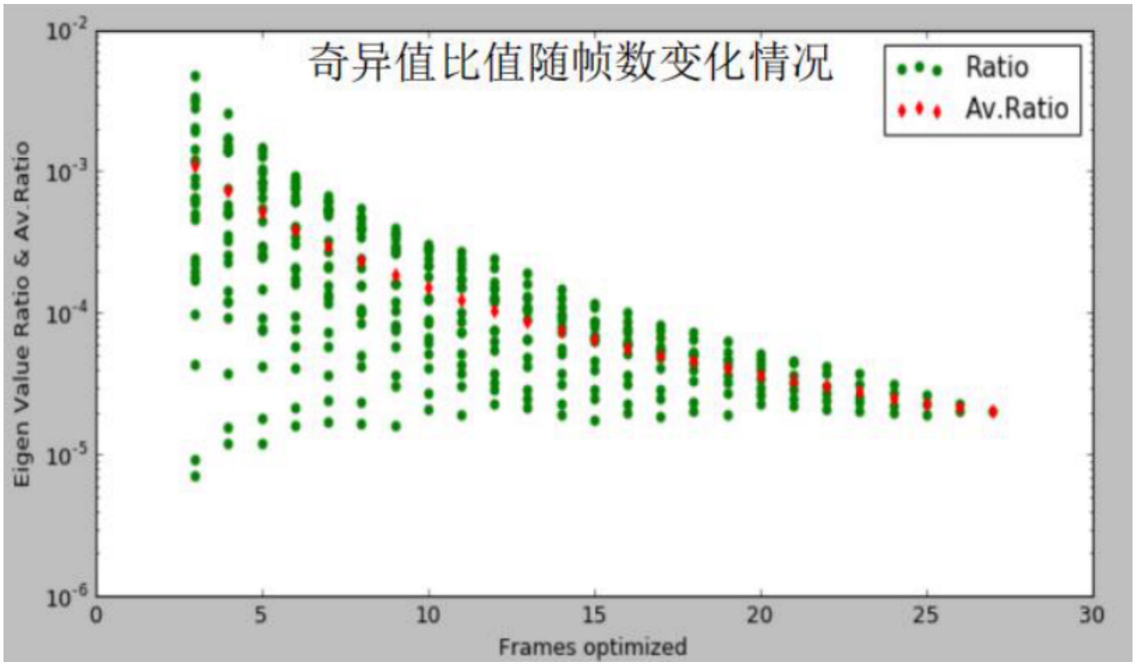

图 2: ratio varies with frame number

可以看出，随着观测帧数的增加，奇异值后两位比值逐渐收敛 (green)，在观测帧较小的时候，也会出现这个比值较小的情况，只是其均值更大，方差也较大，这说明这个比值能够在一定程度上判断三角化的质量，但不是绝对的。同时随着观测次数的增加，这个比值的均值逐渐减小 (red)，即这个比值的变化还是很有规律的，只不过一开始方差较大，所以不能每次都相信，但是统计学或者概率上来说，是可以相信的。