激光 SLAM 第二次作业报告

用户名:sanguin

1.补充直接线性方法的里程计标定模块代码；（6 分）

1）cal_delta_distance()函数对应代码：

```
Eigen::Vector3d  cal_delta_distence(Eigen::Vector3d odom_pose)
{

    Eigen::Vector3d d_pos;  //return value
    now_pos = odom_pose;

    //TODO:
    Eigen::Matrix3d t_now_pos, t_last_pos, t_last_now;
    double c = cos(now_pos(2)), s = sin(now_pos(2));

    //当前帧数据坐标变换矩阵
    t_now_pos << c, -s, now_pos(0),
                 s,  c, now_pos(1),
                 0.0,0.0,1.0;

     //上一帧数据坐标变换矩阵
     c = cos(last_pos(2)); s= sin(last_pos(2));
    t_last_pos << c, -s, last_pos(0),
                  s,  c, last_pos(1),
                  0.0,0.0,1.0;
     t_last_now = t_last_pos.inverse() * t_now_pos;

     // 相对上一帧的坐标
    d_pos << t_last_now(0, 2),
             t_last_now(1, 2),
             atan2(t_last_now(1, 0), t_last_now(0, 0));

    //end of TODO:

    return d_pos;
}
```

2）Add_Data()函数代码：

```
bool OdomCalib::Add_Data(Eigen::Vector3d Odom,Eigen::Vector3d scan)

{

    //这里补充了是否满的判断，防止数据溢出
```

```
    if(now_len<INT_MAX && !is_full())
    {
        //TODO: 构建超定方程组
        A.block<3,9>(now_len*3, 0) << Odom(0), Odom(1), Odom(2), 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                                      0.0, 0.0, 0.0, Odom(0), Odom(1), Odom(2), 0.0, 0.0, 0.0,
                                      0.0, 0.0, 0.0, 0.0, 0.0, 0.0, Odom(0), Odom(1), Odom(2);

        b.segment(now_len*3, 3) << scan(0), scan(1), scan(2);
        now_len++;
        //end of TODO
        return true;
    }
    else
    {
        return false;
    }
}
```

3）Solve()函数代码：

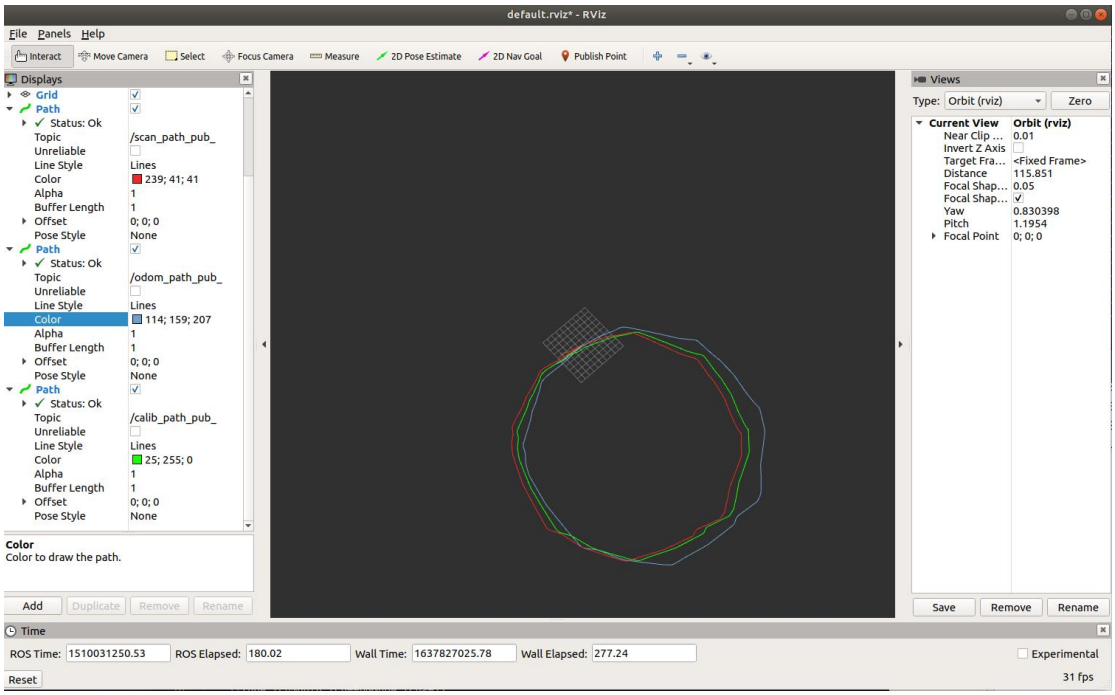
```
Eigen::Matrix3d OdomCalib::Solve()
{
    Eigen::Matrix3d correct_matrix;

    //TODO: 求解线性最小二乘
    Eigen::Matrix<double,9,1> X;
    // 这里判断是否为满
    if (is_full())
    {
        X  = A.colPivHouseholderQr().solve(b);
    }
```

```
    else
    {
        Eigen::MatrixXd AA = A.topRows(now_len * 3);
        Eigen::VectorXd bb = b.head(now_len * 3);
        X = AA.colPivHouseholderQr().solve(bb);
    }
    correct_matrix << X(0), X(1), X(2), X(3), X(4), X(5), X(6), X(7), X(8);
    //end of TODO

    return correct_matrix;
}
```

程序运行结果如下图，其中蓝色为 odom，红色为激光，绿色为矫正后的轨迹。



```
/home/sgl/odom_ws/src/calib_odom/launch/odomCalib.launch http://localhost:11311
File  Edit  View  Search  Terminal  Help
Data Cnt:2418
Data Cnt:2419
Data Cnt:2420
Data Cnt:2421
Data Cnt:2422
Data Cnt:2423
Data Cnt:2424
Data Cnt:2425
Data Cnt:2426
Data Cnt:2427
Data Cnt:2428
Data Cnt:2429
Data Cnt:2430
Data Cnt:2431
Data Cnt:2432
Data Cnt:2433
Data Cnt:2434
Data Cnt:2435
correct_matrix:
  0.936175     1.8629 -0.0444788
-0.0129252    7.14211   0.144319
0.00128128    23.3585   0.381159
calibration over!!!!
```

2. 补充基于模型方法的里程计标定模块代码；（2 分）

1）填充 A、b 矩阵

```
// 填充A, b矩阵
//TODO: (3~5 lines)
A(id_s, 0) = w_Lt;
A(id_s, 1) = w_Rt;
b(id_s) = s_th;
//end of TODO
```

2）进行最小二乘求解 J21J22

```
// 进行最小二乘求解
Eigen::Vector2d J21J22;
//TODO: (1~2 lines)
J21J22 = A.colPivHouseholderQr().solve(b);
//end of TODO
```

3）填充 C，S 矩阵

```
// 填充C, S矩阵
//TODO: (4~5 lines)
C(2*id_s) = cx;
C(2*id_s+1) = cy;
S(2*id_s) = s_x;
S(2*id_s+1) = s_y;
//end of TODO
```

4）进行最小二乘求解，计算 b，r_L，r_R

```
//TODO: (3~5 lines)
b_wheel = C.colPivHouseholderQr().solve(S)(0);
r_L = -J21 * b_wheel;
r_R = J22 * b_wheel;
//end of TODO
```

5）输出结果

```
sgl@ubuntu:~/slamshenlan/odom_calib$ ./odom_calib
J21: -0.163886
J22: 0.170575
b: 0.59796
r_L: 0.0979974
r_R: 0.101997
参考答案：轮间距b为0.6m左右，两轮半径为0.1m左右
```

3. 通过互联网总结学习线性方程组 Ax=b 的求解方法，回答以下问题：（2 分）

（1）对于该类问题，你都知道哪几种求解方法？

目前有人总结了线性方程组 Ax=b 的求解方法[1]，截图如下：

1 方程组的解法概要

对于线性方程组

$a_{11}x_1+\ldots+a_{1n}x_n=b_1$

…

$a_{m1}x_1+\ldots+a_{mn}x_n=b_m$

可以写成矩阵的形式

AX=b

其中 A_{mn}，b_{m1}，X_{n1} 均为矩阵，由线性代数知识可知。关于其解的理论如下：解可能出现三种情形：无解、有唯一解和有无穷多组解。这主要取决于系数矩阵 A 的秩与增广矩阵(A|b)的秩是否相等、秩与变量个数是否相等，具体地：

(1)若 R(A)≠ R(A|b)，则 AX=b 无解；

(2)若 R(A)=R(A|b)=n(n 为变量个数)，则 AX=b 有唯一组解；

(3)若 R(A)=R(A|b)<n，则 AX=b 有无穷多组解。求解方法大概可以划分为两类：直接消去方法(这主要是指高斯消去法)、迭代数值解法。

2 解法比较

(1)直接消去法：理论上，经过有限次算术运算能够求出方程组的精确解；实际上由于计算存在舍入误差，因此，可能得到的只是近似解。高斯消去法可以用 LU 分解来表示。如果方程组的系数矩阵 A 满足一定条件，则 A 可以分解为一个下三角阵 L 与一个上三角阵 U 的乘积：A=LU

于是由 AX=b 得到：

$X=A^{-1}b=(LU)^{-1}b=U^{-1}L^{-1}b$。

用消元法解线性方程组。

(2)迭代法：常用的有雅可比迭代法和高斯－塞德尔迭代法等。其一般形式如下：

将 AX=b 等价变形为(形式不唯一)：

X = BX + f

由此构造迭代公式：

$X^{(n+1)}=BX^{(n)}+f, n=0,1,\ldots$

可以证明：在 B 满足谱半径 r(B)<1 时，当 n®¥ 时，$X^{(n)}$收敛，其极限X*就是原方程组的解。

2.1 雅可比迭代法

设 A=L+D+U(矩阵分解；D 为对角矩阵，L 下三角矩阵，U 为上三角矩阵)，由 AX=b 得到：

(L+D+U)X=b，$X=D^{-1}[-(L+U)X+b]=BX+f$

得到以上(矩阵)形式的迭代公式，其中：$B=-D^{-1}(L+U)$，$f=D^{-1}b$。并且可以证明

$\|X^{(n+1)}-X^*\| \le \|B\|^{n+1}\|X^{(0)}X^*\|$

迭代公式的分量形式如下：

$$x_i^{(k+1)}=\frac{1}{a_{ii}}\left(-\sum_{j\neq i}a_{ij}x_i^{(k)}+b_i\right), i=1,\ldots,n; k=0,1,\ldots$$

2.2 高斯－塞德尔迭代法

类似的方法可以得到高斯－塞德尔迭代公式 X=BX+f，其中 $B=-(D+L)^{-1}U$，$f=(D+L)^{-1}b$。

迭代公式的分量形式如下：

$$x_i^{(k+1)}=\frac{1}{a_{ii}}\left(-\sum_{j<i}a_{ij}x_i^{(k+1)}+\sum_{j>i}a_{ij}x_i^{(k)}+b_i\right), i=1,\cdots\cdots,n; k=0,1,\cdots\cdots$$

加速收敛方法——SOR 方法：如果迭代算法产生的数列{X(k)}收敛，极限记为 X，则由此产生迭代算法 $X^{(k+1)}=X^{(k)}+wR^{(k)}$。基于如上高斯－塞德尔迭代法的加速收敛法如下：

$$x_i^{(k+1)}=x^{(k)}+w\left[\frac{1}{a_{ii}}\left(-\sum_{j<i}a_{ij}x_i^{(k+1)}+\sum_{j>i}a_{ij}x_i^{(k)}+b_i\right)-x^{(k)}\right], i=1,\cdots\cdots,n; k=0,1,\cdots\cdots$$

矩阵形式为：X(k+1)=(D+wL)−1{[(1−w)D−wU]X(k)+wb}

比较：如果 w=1，就是标准的高斯－塞德尔迭代法；如果 w>1，就称为 SOR(Successive Over-Relaxation)方法。

2.3 数值方法－迭代逼近

利用图形的方法或连续函数的零点存在性定理，可以推知 f(x)在某一区间内有根，下面就基于此来讨论求根的数值方法。

a.区间迭代

(1)对分法：对分法是重复应用零点存在性定理，每次将区间压缩一半且其中至少包含一个根，直至最终区间长度(根的精度)满足要求为止。

(2)黄金分割法：黄金分割法与对分法本质上一致，只不过每次压缩区间的比例不是一半，而是压缩比为 $0.618=\frac{\sqrt{5}-1}{2}$。(黄金分割比例)

b.点迭代

(1)简单迭代法：基于构造与方程 f(x)=0 等价的方程 j(x)=x，得到迭代函数 j(x)，然后求迭代函数 j(x)的不动点。注意到 j(x)形式不唯一，可能其迭代差异很大。剩下来的工夫花在 j(x)的构造上，即保证迭代算法的收敛性。对此，有如下结论：

定理：设 j(x)在[a,b]上连续，且满足

(1)对任意的 x∈[a,b]，j(x)∈[a,b]

(2)存在常数 L∈[0,1]，使得对于任意的 x，y∈[a,b]

|j(x)−j(y)| £ L|x−y|

则对任意初值 x0∈[a,b]，迭代过程 $x_{n+1}=j(x_n)$产生的序列{x_n}收敛到 j(x)在[a,b]上的不动点。下面几个迭代方法是根据 f(x)的几何性质与解析性质，从不同角度构造出迭代公式：$x_{n+1}=j(x_n)$

(2)单点割线法：给定初值 x_0(点 P_0)及当前值 x_n(点 P_n)，将曲线 y=f(x)上当前点和初始点构成的割线(代替曲线)与 x 轴的交点的横坐标，作为本次迭代的输出 x_{n+1}，也为下次迭代的输入。其迭代公式为

$$x_{n+1}=x_n-\frac{f(x_n)}{f(x_n)-f(x_0)}(x_n-x_0)$$

(3)两点割线法：给定初值 x_0 及 x_1(两个点)，将曲线上最新的相邻(按迭代先后为序)两点的割线(代替曲线)与 x 轴的交点的横坐标，作为迭代输出。其迭代公式如下：

$$x_{n+1}=x_n-\frac{f(x_n)}{f(x_n)-f(x_{n-1})}(x_n-x_{n-1})$$

(4)三点抛物线法：给定初值 x_0，x_1 及 x_2(三个点)，过曲线上最新三个点的抛物线(代替曲线 f(x))与 x 轴的交点的横坐标 x_{n+1} 就是迭代的输出。其迭代公式如下：

$$x_{n+1}=x_n-\frac{2f(x_n)}{\omega\pm\sqrt{\omega^2-4f(x_n)\cdot\nabla^2 f(x_{n-2})}}(x_n-x_{n-1})$$

其中：±符号的选取以使分母绝对值最大为原则；

$$\omega=\nabla f(x_{n-1})+(x_n-x_{n-1})\nabla^2 f(x_{n-1})$$

$$\nabla^2 f(x_{n-1})=\nabla f(x_{n-1})-\nabla f(x_{n-2})=f(x_n)-2f(x_{n-1})+f(x_{n-2})$$

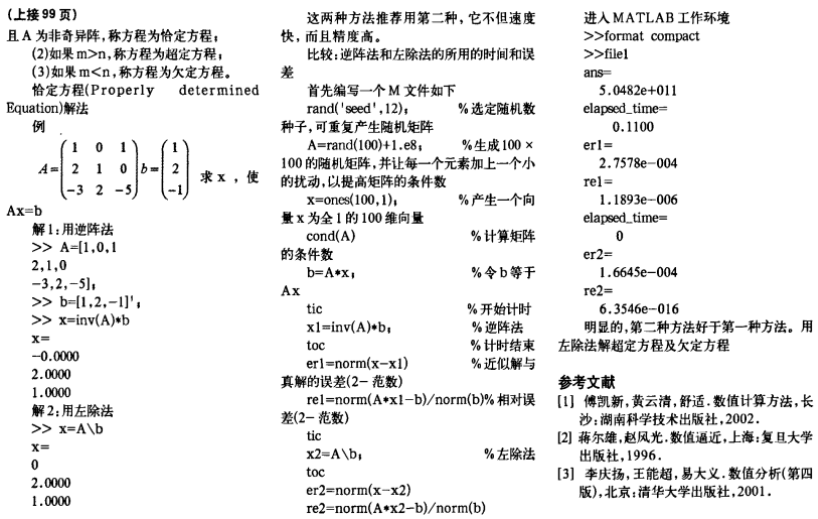
(5)牛顿法：给定初值 x_0(点 P_0)，将过曲线上当前点的切线(代替曲线 f(x))与 x 轴的交点的横坐标 x_{n+1} 作为本次迭代的输出。其迭代公式如下：

$$x_{n+1}=x_n-\left[F'(x^{(n)})\right]^{-1}F(x^{(n)})$$

比较：对分法是重复应用零点存在性定理，每次将区间压缩一半且其中至少包含一个根，直至最终区间长度(根的精度)满足要求为止。黄金分割法与对分法本质上一致，只不过每次压缩区间的比例不是一半，而是压缩比例为 $0.618=\frac{\sqrt{5}-1}{2}$，(黄金分割比例)和简单跌代法、单点割线法、两点割线法、三点抛物线法比较对于正定二次函数，牛顿法一步即可达到最优解。

(1)如果系数矩阵 A 的行数 m 等于列数 n，

(下转 101 页)



(上接99页)

且A为非奇异阵,称方程为恰定方程;

(2)如果m>n,称方程为超定方程,

(3)如果m<n,称方程为欠定方程。

恰定方程(Properly determined Equation)解法

例

$$A=\begin{pmatrix}1&0&1\\2&1&0\\-3&2&-5\end{pmatrix}\quad b=\begin{pmatrix}1\\2\\-1\end{pmatrix}$$

求x,使Ax=b

解1:用逆阵法

```
>> A=[1,0,1
2,1,0
-3,2,-5];
>> b=[1,2,-1]';
>> x=inv(A)*b
x=
-0.0000
2.0000
1.0000
```

解2:用左除法

```
>> x=A\b
x=
0
2.0000
1.0000
```

这两种方法推荐用第二种,它不但速度快,而且精度高。

比较:逆阵法和左除法的所用的时间和误差

首先编写一个M文件如下

```
rand('seed',12);          %选定随机数种子,可重复产生随机矩阵
A=rand(100)+1.e8;         %生成100×100的随机矩阵,并让每一个元素加上一个小的扰动,以提高矩阵的条件数
x=ones(100,1);            %产生一个向量x为全1的100维向量
cond(A)                   %计算矩阵的条件数
b=A*x;                    %令b等于Ax
tic                       %开始计时
x1=inv(A)*b;              %逆阵法
toc                       %计时结束
er1=norm(x-x1)            %近似解与真解的误差(2-范数)
re1=norm(A*x1-b)/norm(b)% 相对误差(2-范数)
tic
x2=A\b;                   %左除法
toc
er2=norm(x-x2)
re2=norm(A*x2-b)/norm(b)
```

进入MATLAB工作环境

```
>>format compact
>>file1
ans=
   5.0482e+011
elapsed_time=
    0.1100
er1=
   2.7578e-004
re1=
   1.1893e-006
elapsed_time=
     0
er2=
   1.6645e-004
re2=
   6.3546e-016
```

明显的,第二种方法好于第一种方法。用左除法解超定方程及欠定方程

参考文献

[1] 傅凯新,黄云清,舒适.数值计算方法,长沙:湖南科学技术出版社,2002.

[2] 蒋尔雄,赵风光.数值逼近,上海:复旦大学出版社,1996.

[3] 李庆扬,王能超,易大义.数值分析(第四版),北京:清华大学出版社,2001.

综合来看，有直接法和迭代法两种。其中直接法又包括 LU 分解法,QR 分解法,SVD(奇异值分解)、特征值分解等,迭代法包括雅可比迭代法和高斯-赛德尔迭代法、超松弛迭代等。

参考文献:

[1]赵慎行. 线性方程组 AX=b 几种解法的比较研究[J]. 科技咨询导报, 2007(13):99-99.

（2）各方法的优缺点有哪些？分别在什么条件下较常被使用？

直接求解对于小矩阵可以,但是对于维度高的矩阵,运算起来效率很低,在大矩阵计算的时候不使用。LU 需要 A 可逆的条件,QR 速度快但是没有 SVD 稳定,SVD 是目前求解最小二乘最好的矩阵分解法。一般在时间效率允许的情况下,选择 SVD 分解。线性方程组的迭代算法是目前计算大型稀疏矩阵方程组的最主要方法之一。

4. 简答题，开放性答案：设计里程计与激光雷达外参标定方法。（2 分）

我们一般把传感器内自身要调节的参数称为内参,比如前面作业中里程计模型的两轮间距与两个轮子的半径。把传感器之间的信息称为外参，比如里程计与激光雷达之间的时间延迟，位姿变换等。请你选用直接线性方法或基于模型的方法，设计一套激光雷达与里程计外参的标定方法，并回答以下问题：

（1）你设计的方法是否存在某些假设？基于这些假设下的标定观测值和预测值分别是什么？

（2）如何构建你的最小二乘方程组求解该外参？

对于以上开放式问题，经过论文搜集整理，目前两篇比较有参考价值的文章，一篇是《移动机器人里程计系统误差及激光雷达安装误差在线标定》[1]，一篇是《Simultaneous calibration of odometry and sensor parameters for mobile robots》[2]。现在结合两篇论文回答（1）、（2）：

问题（1）你设计的方法是否存在某些假设？基于这些假设下的标定观测值和预测值分别是什么？

回答：假设 1：对于激光雷达坐标系在机器人坐标系下的位姿误差模型，假设激光雷达坐标系 xy 平面与机器人坐标系的 xy 平面平行；假设 2：在已经标定好了里程计内参的情况下，标定里程计和激光雷达之间的外参。

标定观测值，激光雷达数据：

$$l = (l_x, l_y, l_{y\theta}) \epsilon SE(2)$$

预测值为：通过激光雷达到机器人底盘的转换矩阵计算后的激光雷达轨迹，公式为：

$$s^k = \ominus\, l \oplus r^k(r_L, r_R, b) \oplus l$$

其中设定 k 时刻与 k+1 时刻，里程计坐标系位姿变化量为r^k，激光雷达坐标系位姿变化量为s^k，r_L, r_R, b分别为左右轮线速度和轮间距。

关于s^k推导过程和符号约定，原论文截图如下：

Table I
SYMBOLS USED IN THIS PAPER

calibration parameters to be estimated	
r_R, r_L	wheel radii
b	distance between wheels
$\boldsymbol{\ell}$	sensor pose relative to robot frame
Robot kinematics	
$\boldsymbol{q}$	robot pose relative to world frame
ω_L, ω_R	left/right wheel velocity
v, ω	driving/steering robot velocities
$\boldsymbol{J}$	linear map between wheel and robot velocities
Sensing process	
$\boldsymbol{m}^k$	exteroceptive measurements, available at time t_k
$\boldsymbol{r}^k$	robot displacement in the k-th interval $[t_k, t_{k+1}]$
$\boldsymbol{s}^k$	sensor displacement in the k-th interval
$\hat{\boldsymbol{s}}^k$	sensor displacement estimated from $\boldsymbol{m}^k$ and $\boldsymbol{m}^{k+1}$
$\boldsymbol{\nu}$	sensor velocity in the sensor frame
Other symbols	
$\oplus, \ominus$	"$\oplus$" is the group operation on SE(2):

$$\begin{pmatrix} a_x \\ a_y \\ a_\theta \end{pmatrix} \oplus \begin{pmatrix} b_x \\ b_y \\ b_\theta \end{pmatrix} = \begin{pmatrix} a_x + b_x \cos(a_\theta) - b_y \sin(a_\theta) \\ a_y + b_x \sin(a_\theta) + b_y \cos(a_\theta) \\ a_\theta + b_\theta \end{pmatrix}$$

"$\ominus$" is the group inverse:

$$\ominus \begin{pmatrix} a_x \\ a_y \\ a_\theta \end{pmatrix} = \begin{pmatrix} -a_x \cos(a_\theta) - a_y \sin(a_\theta) \\ +a_x \sin(a_\theta) - a_y \cos(a_\theta) \\ -a_\theta \end{pmatrix}$$

$$\boldsymbol{s}^k = \ominus \left(\boldsymbol{q}^k \oplus \boldsymbol{\ell}\right) \oplus \left(\boldsymbol{q}^{k+1} \oplus \boldsymbol{\ell}\right).$$

$$\dot{\boldsymbol{q}} = \begin{pmatrix} \cos q_\theta & 0 \\ \sin q_\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix}.$$

$\boldsymbol{r}^k = \ominus \boldsymbol{q}^k \oplus \boldsymbol{q}^{k+1}$：*机器人里程计推算的机器人位姿*

（2）如何构建你的最小二乘方程组求解该外参？

回答：上面的s^k是标定预测值，即是通过里程计和设定的外参计算得到的，而激光雷达自身可以通过匹配得到 k 时刻到 k+1 时刻的转换位姿$\hat{s}^k$，对应可以使用最小二乘法来优化下面的目标函数：

$$\mathcal{J} = -\frac{1}{2}\sum_{k=1}^{n} ||\hat{\boldsymbol{s}}^k - \ominus\boldsymbol{\ell} \oplus \boldsymbol{r}^k(r_{\mathrm{L}}, r_{\mathrm{R}}, b) \oplus \boldsymbol{\ell}||^2_{\boldsymbol{\Sigma}_k^{-1}}$$

解法及目标方程截图：

1) Treatable likelihood approximation: The first step is simplifying the expression (10) for the log-likelihood. For the standard 2-norm, the following equivalence holds:

$$||\boldsymbol{s}^k - \ominus\boldsymbol{\ell} \oplus \boldsymbol{r}^k \oplus \boldsymbol{\ell}||_2 = ||\boldsymbol{\ell} \oplus \boldsymbol{s}^k - \boldsymbol{r}^k \oplus \boldsymbol{\ell}||_2.$$

Intuitively, the two vectors on the left and right hand side represent the same quantity in two different reference frames, so they have the same norm. This is not true for a generic matrix norm. However, it is true for the $\boldsymbol{\Sigma}_k^{-1}$-norm, which, thanks to Assumption 1 above is isotropic in the x and y directions, and hence rotation-invariant. Therefore, the log-likelihood (10) can be written as

$$\mathcal{J} = -\frac{1}{2}\sum_{k} ||\boldsymbol{\ell} \oplus \hat{\boldsymbol{s}}^k - \boldsymbol{r}^k \oplus \boldsymbol{\ell}||^2_{\boldsymbol{\Sigma}_k^{-1}}. \tag{13}$$

3) Formulation as a quadratic system: We now use the trick of treating $\cos\ell_\theta$ and $\sin\ell_\theta$ as two independent variables. If we group the remaining parameters in the vector $\boldsymbol{\varphi} \in \mathbb{R}^5$ as

$$\boldsymbol{\varphi} = \begin{pmatrix} b & \ell_x & \ell_y & \cos\ell_\theta & \sin\ell_\theta \end{pmatrix}^T,$$

then (13) can be written as a quadratic function of $\boldsymbol{\varphi}$. More in detail, defining the 2×5 matrix $\boldsymbol{Q}_k$ of known coefficients as

$$\boldsymbol{Q}_k = \frac{1}{\sigma_{xy}^k}\begin{pmatrix} -c_x^k & 1-\cos\hat{r}_\theta^k & +\sin\hat{r}_\theta^k & +\hat{s}_x^k & -\hat{s}_x^k \\ -c_y^k & -\sin\hat{r}_\theta^k & 1-\cos\hat{r}_\theta^k & +\hat{s}_y^k & +\hat{s}_x^k \end{pmatrix}, \tag{20}$$

the log-likelihood function (13) can be written compactly as $-\frac{1}{2}\boldsymbol{\varphi}^T\boldsymbol{M}\boldsymbol{\varphi}$ + constant with $\boldsymbol{M} = \sum_k \boldsymbol{Q}_k^T\boldsymbol{Q}_k$. We have reduced the maximization of the likelihood to a quadratic problem with a quadratic constraint:

$$\min \quad \boldsymbol{\varphi}^T\boldsymbol{M}\boldsymbol{\varphi}, \tag{21}$$

$$\text{subject to} \quad \varphi_4^2 + \varphi_5^2 = 1. \tag{22}$$

Constraint (22), corresponding to $\cos^2\ell_\theta + \sin^2\ell_\theta = 1$, is necessary to enforce geometric consistency.

The correct solution $\hat{\boldsymbol{\varphi}}$ to (21) can be chosen between $\boldsymbol{\varphi}^{(1)}$ and $\boldsymbol{\varphi}^{(2)}$ by computing the value of the objective function. Given $\hat{\boldsymbol{\varphi}}$ and the previously estimated values of $\hat{J}_{21}, \hat{J}_{22}$, all six parameters can be recovered as follows:

$$\begin{aligned} \hat{b} &= \hat{\varphi}_1, \\ \hat{r}_{\mathrm{L}} &= -\hat{\varphi}_1 \hat{J}_{21}, \\ \hat{r}_{\mathrm{R}} &= +\hat{\varphi}_1 \hat{J}_{22}, \\ \hat{\boldsymbol{\ell}} &= (\hat{\varphi}_2, \hat{\varphi}_3, \operatorname{arctan2}(\hat{\varphi}_5, \hat{\varphi}_4)). \end{aligned} \tag{30}$$

参考文献：

[1]达兴鹏， 曹其新， 王雯珊. 移动机器人里程计系统误差及激光雷达安装误差在线标定[J]. 机器人, 2017(2):9.

[2] Censi A ， Franchi A ， Marchionni L ， et al. Simultaneous Calibration of Odometry and Sensor Parameters for Mobile Robots[J]. IEEE Transactions on Robotics, 2013, 29(2):475-492.