智能传感第二次作业

截止日期：2024 年 6 月 10 日 23:59

假设机器人携带声源在室内围绕麦克风阵列运动同时多次发出声音，我们需要基于到达时间差（Time Difference of Arrival, TDOA）测量和机器人里程计信息去同时估计麦克风阵列参数（位置、时间补偿和时钟漂移）和所有声音事件位置。实验示例场景见**视频附件**。视频中，机器人在黑色标记点停下并发出声音，麦克风阵列由三个阵列中各取一个麦克风组成。

给定 N 个麦克风和 K 个声音事件，TDOA 的测量模型如下所示：

$$\mathrm{T_{i,j}} = \frac{||\mathbf{x_i} - \mathbf{s_j}|| - ||\mathbf{x_1} - \mathbf{s_j}||}{\mathrm{c}} + \tau_{\mathrm{i,1}} + \delta_{\mathrm{i,1}}\Delta\mathrm{t_j}$$

其中，$\mathbf{x_i}$是第 i 个麦克风位置（i>1），$\mathbf{s_j}$是机器人发出第 j 个声音时的位置，$\tau_{\mathrm{i,1}}$是第 i 个麦克风相对第一个麦克风的时间补偿参数，$\delta_{\mathrm{i,1}}$是第 i 个麦克风相对于第一个麦克风的时钟漂移参数，$\Delta\mathrm{t_j}$是第 j 个声音发出时刻相对第一个声音发出时刻的时间差。

里程计测量模型如下所示：

$$\Delta\mathbf{s_j} = \mathbf{s_{j+1}} - \mathbf{s_j}$$

TDOA 和里程计对应的测量值是$t_{i,j} = \mathrm{T_{i,j}} + \omega_{i,j}$和$\boldsymbol{m}_j = \Delta\boldsymbol{s}_j + \boldsymbol{v}_j$，其中噪声$\omega_{i,j}$和$\boldsymbol{v}_{i,j}$独立同分布且服从高斯分布，$\omega_{i,j}\sim N(0,\sigma_{tdoa}^2)$和$\boldsymbol{v}_j\sim N(0,\sigma_{odo}^2\boldsymbol{I_3})$。

符号约定：$\boldsymbol{I_K}$是一个$K\times K$的单位矩阵，若$\boldsymbol{y}=\begin{bmatrix}y_1\\y_2\end{bmatrix}$和$\boldsymbol{x}=\begin{bmatrix}x_1\\x_2\end{bmatrix}$，则$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}=\begin{bmatrix}\frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2}\\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2}\end{bmatrix}$，若$a$是一个标量，$[\boldsymbol{x},\boldsymbol{y},a]^T=\begin{bmatrix}x_1\\x_2\\y_1\\y_2\\a\end{bmatrix}_{5\times1}$，加粗的字母表示向量或者矩阵，未加粗的字母表示标量。

进一步的，将所有测量组成一个向量$\boldsymbol{z}=[\boldsymbol{t},\boldsymbol{m}]^T$，对应的测量函数组成一个函数向量$f(\boldsymbol{x})=[\boldsymbol{T},\Delta\boldsymbol{s}]^T$，具体而言，$\boldsymbol{t}=[\boldsymbol{t_1},\boldsymbol{t_2},\dots,\boldsymbol{t_K}]^T$,$\boldsymbol{t_j}=\left[t_{2,j},t_{3,j},\dots,t_{N,j}\right]^T$,

$\boldsymbol{T}=[\boldsymbol{T_1},\boldsymbol{T_2},\dots,\boldsymbol{T_K}]^T$，$\boldsymbol{T_j}=\left[T_{2,j},T_{3,j},\dots,T_{N,j}\right]^T$，$\boldsymbol{m}=[\boldsymbol{m_1},\boldsymbol{m_2},\dots,\boldsymbol{m_{K-1}}]^T$，$\Delta\boldsymbol{s}=[\Delta\boldsymbol{s_1},\Delta\boldsymbol{s_2},\dots,\Delta\boldsymbol{s_{K-1}}]^T$。

另外，基于$\boldsymbol{x_1}=\boldsymbol{0},(\boldsymbol{x_2})_y=(\boldsymbol{x_2})_z=(\boldsymbol{x_3})_z=0$建立全局坐标系，因此所有需要估计的参数为$\boldsymbol{x}=[\boldsymbol{x_{mic}},\boldsymbol{s}]^T$，具体而言，

$\boldsymbol{x_{mic}}=\left[(\boldsymbol{x_2})_x,\tau_{2,1},\delta_{2,1},(\boldsymbol{x_3})_x,(\boldsymbol{x_3})_y,\tau_{3,1},\delta_{3,1},\boldsymbol{x_4},\tau_{4,1},\delta_{4,1},\dots,\boldsymbol{x_N},\tau_{N,1},\delta_{N,1}\right]^T$ 和 $\boldsymbol{s}=[\boldsymbol{s_1},\boldsymbol{s_2},\dots,\boldsymbol{s_K}]^T$。

最后，该同时标定和定位问题可转化为求解基于极大似然估计的非线性最小二乘：

$$\min_{\boldsymbol{x}}(f(\boldsymbol{x})-\boldsymbol{z})^T\mathbf{W}^{-1}(f(\boldsymbol{x})-\boldsymbol{z}) \tag{1}$$

其中，$\mathbf{W}=\boldsymbol{diag}(\sigma_{tdoa}^2\mathbf{I_{K(N-1)}},\sigma_{odo}^2\mathbf{I_{3(K-1)}})$，$\boldsymbol{diag}(\cdot)$将内部子矩阵放置于对角线处，$\mathbf{W}$是对角矩阵。简单理解极大似然估计和最小二乘之间关系的网址链接：https://zhuanlan.zhihu.com/p/55793850。

高斯牛顿法是基于梯度的经典迭代求解算法，步骤如下：

1. 给定初值$\boldsymbol{x_0}$，一般随机生成。

2. 给定$\boldsymbol{x_k}$ $(k \geq 0)$，开始迭代：

2.1 线性化：

$$f(\boldsymbol{x}) = \frac{\partial f(x)}{\partial x}|_{x=x_k}(\boldsymbol{x} - \boldsymbol{x_k}) + f(\boldsymbol{x_k}) = \boldsymbol{J_k d_k} + f(\boldsymbol{x_k}) \quad (2)$$

2.2 将(2)带入(1)，写出$\boldsymbol{d_k}$的推导结果（提示：2.2 的优化问题是关于$\boldsymbol{d_k}$的无约束线性最小二乘），结果用$\boldsymbol{J_k}$和$\boldsymbol{r_k}$表示，其中$\boldsymbol{J_k} = \frac{\partial f(x)}{\partial x}|_{x=x_k}$，$\boldsymbol{r_k} = f(\boldsymbol{x_k}) - \boldsymbol{z}$。

2.3 $\boldsymbol{x_{k+1}} = \boldsymbol{x_k} + \boldsymbol{d_k}$

2.4 如果$||\boldsymbol{x_{k+1}} - \boldsymbol{x_k}|| \leq 0.01$或者$f(\boldsymbol{x_{k+1}}) \leq 30$，则收敛并退出，否则继续迭代。

2.5 若 2 的执行时间未超过 T/10，继续迭代

2.6 若满足某些条件则发散且退出 2，重新执行 1 和 2

2.7 若整个算法执行时间超过 T，则退出且宣告估计失败

2.8 $k \leftarrow k + 1$

整个高斯牛顿法代码已在 `GN_Solver.m` 中展示，2.2 需要自行推导并做代码实现。

任务 1：推导高斯牛顿法中的 2.2 并展示推导过程，然后在代码的 `GN_Solver.m` 中完成 `dk` 的计算。

任务 2：推导雅可比矩阵中重要子矩阵$\boldsymbol{E_{ij}}, \boldsymbol{G_{ij}}$并展示推导过程，然后在代码的 `compute_J.m` 中完成$\boldsymbol{E_{ij}}, \boldsymbol{G_{ij}}, r_{tdoa_{i,j}}, \boldsymbol{r_{odo_j}}$的计算。其中，$\boldsymbol{E_{ij}} = \frac{\partial \boldsymbol{T_{i,j}}}{\partial [x_i, \tau_{i,1}, \delta_{i,1}]^T}, \boldsymbol{G_{ij}} = \frac{\partial \boldsymbol{T_{i,j}}}{\partial \boldsymbol{s_j}}$分别是$1 \times 5$和$1 \times 3$的矩阵，$r_{tdo\ i,j} = \mathrm{T}_{i,j} - t_{i,j}, \boldsymbol{r_{odo_j}} = \Delta \mathbf{s_j} - \boldsymbol{m_j}$。计算$\boldsymbol{E_{ij}}$时，$i = 4,5,\ldots,N$。`compute_J.m` 里涉及计算的变量名和本文档定义的变量名之间的关系已在 `compute_J.m` 标出。

任务 3：基于任务 1 和任务 2 完成的代码，在 `main.m` 实现同时麦克风阵列标定和声音事件位置估计（原始 `main.m` 只实现了 1 轮估计）。三种初值位置噪声标准差（越大意味着初值误差越大），分别为$\sigma_{init} = 0.5, 1, 2$，要求在每种$\sigma_{init}$下，重复 50 轮（估计失败的情况不用来计算均值，运行时建议注释掉可视化代码）并计算这 50 轮的平均麦克风位置估计误差，`su_err(6)`或者 `sg.rec(6)`是每轮麦克风阵列位置估计误差。每轮重新生成的 `ori_g` 分别在三种σ_{init}下进行估计。最后分析实验结果并解释：为什么传统非线性优化结果依赖初值？

提交要求：任务 1 和任务 2 的推导结果以及任务 3 的实验结果分析和问题解释请写在一个文档内并以 pdf 格式提交，将实现任务 1,2,3 的代码压缩打包.zip。将.zip 和.pdf 放在一个文件夹内且该文件夹、.zip 和.pdf 名称均为：智能传感第二次作业+姓名+学号，例如：智能传感第二次作业-张三-12312323