深蓝学院
shenlanxueyuan.com

视觉SLAM进阶课程
从零开始手写VIO-第5期

第2讲 IMU_motion_measurement 作业讲评

主讲人 吕华龙

【作业】IMU仿真实践

① 设置IMU仿真代码中不同的参数，生成Allan方差标定曲线。

https://github.com/gaowenliang/imu_utils

https://github.com/rpng/kalibr_allan

② 将IMU仿真代码中欧拉积分替换成中值积分。

③ 提升作业：阅读文献，撰写总结推导

Lovegrove S, Patron-Perez A, Sibley G. Spline Fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras[C]//BMVC. 2013, 2(5): 8.

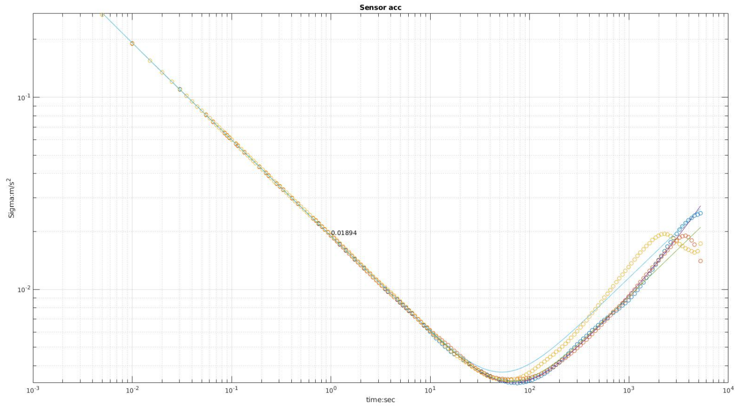
Allan方差标定

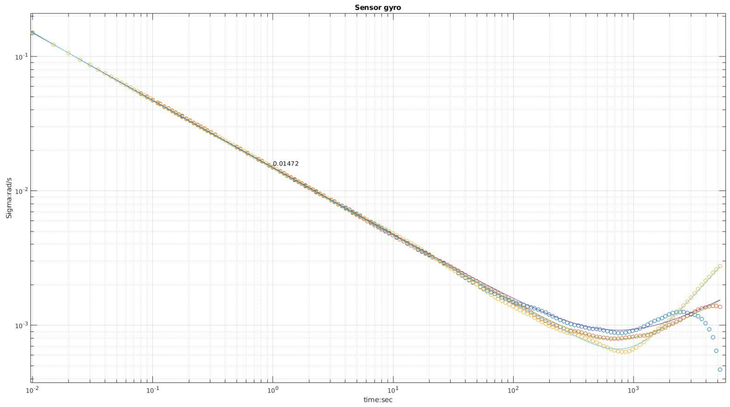
仿真数据的产生：

设定每一个时刻位置的运动方程以及旋转(欧拉角)的运动方程，对方程进行求导，得到每一个timestamp 的 acc 和 gyro，(需要注意欧拉角速度到Body角速度的变换)

基于ROS版本的imu_utils的IMU误差标定：

vio_data_simulation ==> imu.bag ==> launch imu_utils simulation.launch





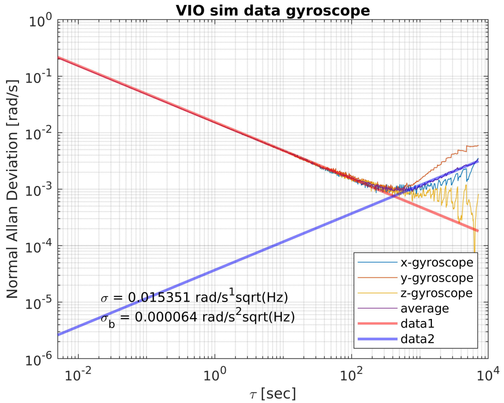
Allan方差标定

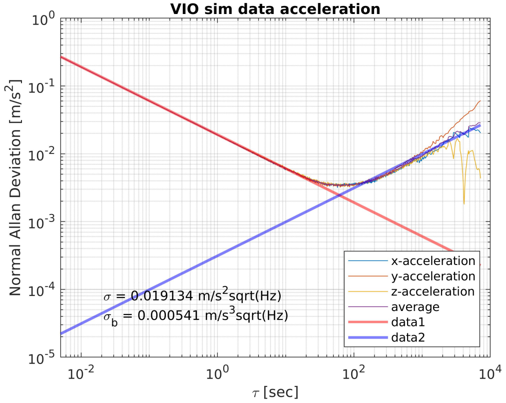
基于非ROS版本的kalibr_allan的IMU误差标定：

imu.bag ==> bag convert ==> kalibr allan

很多同学进行了imu_utils与kalibr_allan两种不同标定方法的结果对比，以及设定不同大小的误差，进行多次实验比较。

误差类型	仿真设定值	imu_utils	kalibr_allan
加速度计白噪声($\frac{m}{s^2}\frac{1}{\sqrt{Hz}}$)	0.019	0.0190	0.019169
加速度计随机游走($\frac{m}{s^3}\frac{1}{\sqrt{Hz}}$)	0.0005	0.00024269（估计）	0.000571
陀螺仪白噪声($\frac{rad}{s}\frac{1}{\sqrt{Hz}}$)	0.015	0.0148	0.015028
陀螺仪随机游走($\frac{rad}{s^2}\frac{1}{\sqrt{Hz}}$)	0.00005	0.00005596（估计）	0.000041





IMU积分

IMU传感器可以测量当前Body的角速度和加速度，
因此对IMU测量进行积分，可以得到Body的当前P、V、Q

$$\tilde{\boldsymbol{\omega}}^b = \boldsymbol{\omega}^b + \mathbf{b}^g + \mathbf{n}^g$$
$$\tilde{\mathbf{a}}^b = \mathbf{q}_{bw}(\mathbf{a}^w + \mathbf{g}^w) + \mathbf{b}^a + \mathbf{n}^a$$

测量模型

$$\mathbf{p}_{wb_{k+1}} = \mathbf{p}_{wb_k} + \mathbf{v}_k^w \Delta t + \frac{1}{2}\mathbf{a}\Delta t^2$$
$$\mathbf{v}_{k+1}^w = \mathbf{v}_k^w + \mathbf{a}\Delta t$$
$$\mathbf{q}_{wb_{k+1}} = \mathbf{q}_{wb_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\boldsymbol{\omega}\delta t \end{bmatrix}$$

运动模型

$$\mathbf{a} = \mathbf{q}_{wb_k}\left(\mathbf{a}^{b_k} - \mathbf{b}_k^a\right) - \mathbf{g}^w$$
$$\boldsymbol{\omega} = \boldsymbol{\omega}^{b_k} - \mathbf{b}_k^g$$

欧拉积分

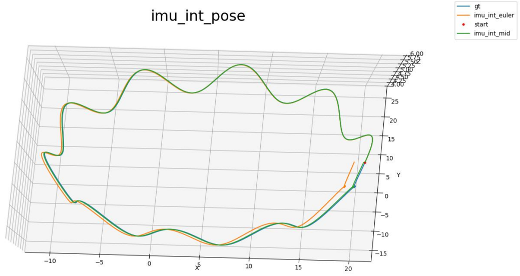
中值积分

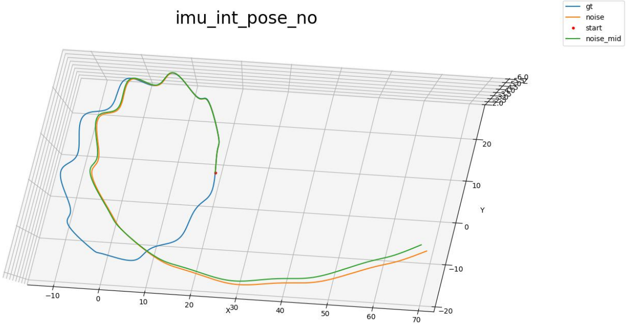
$$\mathbf{a} = \frac{1}{2}\left[\mathbf{q}_{wb_k}\left(\mathbf{a}^{b_k} - \mathbf{b}_k^a\right) - \mathbf{g}^w + \mathbf{q}_{wb_{k+1}}\left(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a\right) - \mathbf{g}^w\right]$$
$$\boldsymbol{\omega} = \frac{1}{2}\left[\left(\boldsymbol{\omega}^{b_k} - \mathbf{b}_k^g\right) + \left(\boldsymbol{\omega}^{b_{k+1}} - \mathbf{b}_k^g\right)\right]$$

```
// ***************** 欧拉积分 ***********************************************
// delta_q = [1 , 1/2 * thetax , 1/2 * theta_y, 1/2 * theta_z]
// Eigen::Quaterniond dq;
// Eigen::Vector3d dtheta_half = imupose.imu_gyro * dt / 2.0;
// dq.w() = 1;
// dq.x() = dtheta_half.x();
// dq.y() = dtheta_half.y();
// dq.z() = dtheta_half.z();
// dq.normalize();

// Eigen::Vector3d acc_w = Qwb * (imupose.imu_acc) + gw;  // aw = Rwb * ( acc_body - acc_bias ) + gw
// Qwb = Qwb * dq;
// Pwb = Pwb + Vw * dt + 0.5 * dt * dt * acc_w;
// Vw = Vw + acc_w * dt;

// ***************** 中值积分 ***********************************************
MotionData imupose_pre = imudata[i - 1];
Eigen::Vector3d mid_omega = (imupose_pre.imu_gyro + imupose.imu_gyro) * 0.5;
Eigen::Vector3d dtheta_half = mid_omega * dt / 2.0;
Eigen::Quaterniond dq(1, dtheta_half.x(), dtheta_half.y(), dtheta_half.z());
dq.normalize();

Eigen::Quaterniond Qwb_newx = Qwb * dq;

Eigen::Vector3d acc_w = 0.5 * (Qwb * (imupose.imu_acc) + gw + Qwb_newx * (imupose_pre.imu_acc) + gw);
Vw = Vw + acc_w * dt;
Pwb = Pwb + Vw * dt + 0.5 * dt * dt * acc_w;
Qwb = Qwb_newx;
```

IMU积分

通过比较可以发现，中值积分的结果会比欧拉积分略好一些。
在仿真中我们可以设定IMU数据是否带噪声，在实际中使用中要考虑噪声和偏置的影响。






```cpp
void IMU::eulerIntegration(double _dt,
                           const Eigen::Vector3d &acc_k, const Eigen::Vector3d &gyro_k,                // 第k帧 IMU data
                           const Eigen::Vector3d &acc_bias, const Eigen::Vector3d &gyro_bias,          // IMU 偏置项，这里假定为常数
                           Eigen::Vector3d &delta_p, Eigen::Quaterniond &delta_q, Eigen::Vector3d &delta_v //前一帧result,以及updated当前帧积分result
)
{
    Eigen::Vector3d gw(0, 0, -9.81); // ENU frame

    Eigen::Vector3d un_gyro = gyro_k - gyro_bias;                                   //  w = gyro_body - gyro_bias
    Eigen::Vector3d un_acc = delta_q.toRotationMatrix() * (acc_k - acc_bias) + gw; // aw = Rwb * ( acc_body - acc_bias ) + gw

    //  △q  delta_q = [1 , 1/2 * thetax , 1/2 * theta_y, 1/2 * theta_z]
    delta_q = delta_q * Eigen::Quaterniond(1, un_gyro(0) * _dt / 2, un_gyro(1) * _dt / 2, un_gyro(2) * _dt / 2);
    // △p
    delta_p = delta_p + delta_v * _dt + 0.5 * un_acc * _dt * _dt;
    // △v
    delta_v = delta_v + un_acc * _dt;
}

void IMU::midPointIntegration(double _dt,
                              const Eigen::Vector3d &acc_0, const Eigen::Vector3d &gyro_0,
                              const Eigen::Vector3d &acc_1, const Eigen::Vector3d &gyro_1,
                              const Eigen::Vector3d &acc_bias, const Eigen::Vector3d &gyro_bias,
                              Eigen::Vector3d &delta_p, Eigen::Quaterniond &delta_q, Eigen::Vector3d &delta_v)
{
    Eigen::Vector3d gw(0, 0, -9.81); // ENU frame
    Eigen::Vector3d un_gyro = 0.5 * (gyro_0 + gyro_1) - gyro_bias;

    Eigen::Vector3d un_acc_0 = delta_q.toRotationMatrix() * (acc_0 - acc_bias) + gw;
    // △q
    delta_q = delta_q * Eigen::Quaterniond(1, un_gyro(0) * _dt / 2, un_gyro(1) * _dt / 2, un_gyro(2) * _dt / 2);

    Eigen::Vector3d un_acc_1 = delta_q.toRotationMatrix() * (acc_1 - acc_bias) + gw;
    Eigen::Vector3d un_acc = 0.5 * (un_acc_0 + un_acc_1);
    // △p
    delta_p = delta_p + delta_v * _dt + 0.5 * un_acc * _dt * _dt;
    // △v
    delta_v = delta_v + un_acc * _dt;
}
```

基于累积3次B-Spline的SE3插值

Spline Fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras

Cubic B-Splines:

常被用来表示 R^3 的轨迹，但在处理3D旋转时有困难，比如 $SO(3)$

所以本文采用**Cumulative B-Spline basis functions**

- 累积形式确保了对流形插值所需的局部连续性；
- 3次样条确保**C²**连续性，可以计算沿轨迹任意点的加速度；
- 能够很好的逼近最小转矩轨迹

累计基函数的 B 次样条曲线如下：

$$\mathbf{p}(t)=\sum_{i=0}^{n}\mathbf{p}_i B_{i,k}(t) \tag{1}$$

其中，$\mathbf{p}_i \in \mathbb{R}^N$，表示 t_i 时刻的控制点，$B_{i,k}(t)$ 为基函数，公式如下：

$$B_{i,0}(x)=\begin{cases}1, & t_i \le x < t_{i+1}\\ 0, & otherwise\end{cases} \tag{2}$$

$$B_{i,p}(x)=\frac{x-t_i}{t_{i+p}-t_i}B_{i,p-1}(x)+\frac{t_{i+p+1}-x}{t_{i+p+1}-t_{i+1}}B_{i+1,p-1}(x) \tag{3}$$

公式(1)可以改写为：

$$\mathbf{p}(t)=\mathbf{p}_0\tilde{B}_{0,k}(t)+\sum_{i=0}^{n}(\mathbf{p}_i-\mathbf{p}_{i-1})\tilde{B}_{i,k}(t) \tag{4}$$

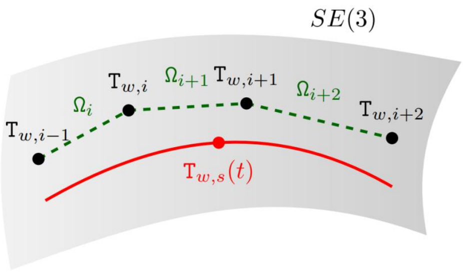
其中，$\tilde{B}_{i,k}(t)=\sum_{j=i}^{n}B_{j,k}(t)$ 为累计基函数，然后通过用控制点之间的对数变换 $\Omega_i=\log\left(\mathbf{T}_{w,i-1}^{-1},\mathbf{T}_{w,i}\right)$ 操作代替控制点之差来描述 SE3 中的轨迹，公式(4)可以进一步变换成：

$$\mathbf{T}_{w,s}(t)=\exp\left(\tilde{B}_{0,k}(t)\log\mathbf{T}_{w,0}\right)\prod_{i=1}^{n}\exp\left(\tilde{B}_{0,k}(t)\Omega_i\right) \tag{5}$$

其中，$\mathbf{T}_{w,s}(t)\in$ SE3 是样条曲线上 t 时刻的位姿，而 $\mathbf{T}_{w,i}\in$ SE3 就是世界坐标系下控制点位姿。

基于累积3次B-Spline的SE3插值

对于四次累计 B 样条曲线，$t \in [t_i, t_{i+1}]$这段时间中一共有$[t_{i-1}, t_i, t_{i+1}, t_{i+2}]$四个控制点来确定时刻 t 的样条曲线上的值，使用 $s(t)=(t-t_0)/\Delta t$ 来表示平均时间的函数，控制点的时刻 t_i 就可以由平均时间函数 $s_i \in [0,1,\cdots,n]$来表示，对于时间 $s_i \le s(t) < s_{i+1}$，定义 $u(t)=s(t)-s_i$ 来表示。重写矩阵形式的 B 样条曲线以及它的一阶二阶微分函数如下：



$$\tilde{\mathbf{B}}(u)=\mathbf{C}\begin{bmatrix}1\\u\\u^2\\u^3\end{bmatrix},\quad \dot{\tilde{\mathbf{B}}}(u)=\frac{1}{\Delta t}\mathbf{C}\begin{bmatrix}0\\1\\2u\\3u^2\end{bmatrix},\quad \ddot{\tilde{\mathbf{B}}}(u)=\frac{1}{\Delta t^2}\mathbf{C}\begin{bmatrix}0\\0\\2\\6u\end{bmatrix},\quad \mathbf{C}=\frac{1}{6}\begin{bmatrix}6&0&0&0\\5&3&-3&1\\1&3&3&-2\\0&0&0&1\end{bmatrix} \tag{6}$$

样条轨迹上的位姿可以定义为：

$$\mathbf{T}_{w,s}(u)=\mathbf{T}_{w,i-1}\prod_{j=1}^{3}\exp\left(\tilde{\mathbf{B}}(u)_j\,\Omega_{i+j}\right) \tag{7}$$

其中 i 下标表示时间 t 所在的时间间隔区间，求上式的一阶导数和二阶导数，也就是对应的速度和加速度，如下：

$$\begin{aligned}
\dot{\mathbf{T}}_{w,s}(u) &= \mathbf{T}_{w,i-1}\left(\dot{\mathbf{A}}_0\mathbf{A}_1\mathbf{A}_2+\mathbf{A}_0\dot{\mathbf{A}}_1\mathbf{A}_2+\mathbf{A}_0\mathbf{A}_1\dot{\mathbf{A}}_2\right),\\
\ddot{\mathbf{T}}_{w,s}(u) &= \mathbf{T}_{w,i-1}\begin{pmatrix}\ddot{\mathbf{A}}_0\mathbf{A}_1\mathbf{A}_2+\mathbf{A}_0\ddot{\mathbf{A}}_1\mathbf{A}_2+\mathbf{A}_0\mathbf{A}_1\ddot{\mathbf{A}}_2+\\ 2\left(\dot{\mathbf{A}}_0\dot{\mathbf{A}}_1\mathbf{A}_2+\dot{\mathbf{A}}_0\mathbf{A}_1\dot{\mathbf{A}}_2+\mathbf{A}_0\dot{\mathbf{A}}_1\dot{\mathbf{A}}_2\right)\end{pmatrix},\\
\mathbf{A}_j &= \exp\left(\Omega_{i+j}\tilde{\mathbf{B}}(u)_j\right),\quad \dot{\mathbf{A}}_j=\mathbf{A}_j\Omega_{i+j}\dot{\tilde{\mathbf{B}}}(u)_j,\\
\ddot{\mathbf{A}}_j &= \dot{\mathbf{A}}_j\Omega_{i+j}\dot{\tilde{\mathbf{B}}}(u)_j+\mathbf{A}_j\Omega_{i+j}\ddot{\tilde{\mathbf{B}}}(u)_j
\end{aligned}$$

基于累积3次B-Spline的SE3插值

最后根据推导的结论，利用已知的轨迹在B样条曲线上插值生成观测路标点和IMU测量

视觉惯性数据模型：在 a 帧投影的逆深度转换到 b 帧图像系下，

$$\mathbf{p}_b = \mathcal{W}(\mathbf{p}_a;\mathbf{T}_{b,a},\rho) = \pi\left([\mathbf{K}_b \,|\, \mathbf{0}]\,\mathbf{T}_{b,a}\left[\mathbf{K}_a^{-1}\begin{bmatrix}\mathbf{p}_a\\1\end{bmatrix};\rho\right]\right) \tag{9}$$

利用推导出来 B 样条曲线的公式，生成加速度计和陀螺仪的测量模型：

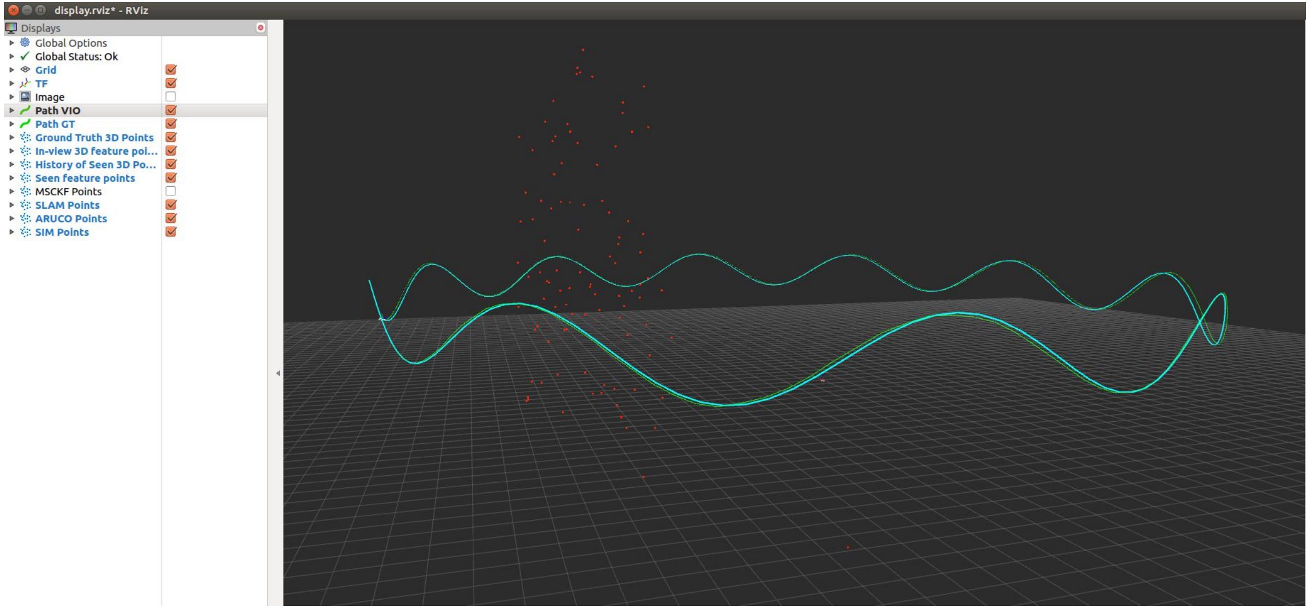
$$\begin{aligned}\mathrm{Gyro}(u) &= \mathbf{R}_{w,s}^{\mathsf{T}}(u)\cdot\dot{\mathbf{R}}_{w,s}(u)+\mathrm{bias},\\ \mathrm{Accel}(u) &= \mathbf{R}_{w,s}^{\mathsf{T}}(u)\cdot(\ddot{\mathbf{s}}_w(u)+g_w)+\mathrm{bias}.\end{aligned} \tag{10}$$

误差最小化：

$$\begin{aligned}E(\theta) = &\sum_{\hat{\mathbf{p}}_m}\left(\hat{\mathbf{p}}_m-\mathcal{W}(\mathbf{p}_r;\mathbf{T}_{c,s}\mathbf{T}_{w,s}(u_m)^{-1}\mathbf{T}_{w,s}(u_r)\mathbf{T}_{s,c},\rho)\right)^2_{\Sigma_p}+\\ &\sum_{\hat{\omega}_m}\left(\hat{\omega}_m-\mathrm{Gyro}(u_m)\right)^2_{\Sigma_\omega}+\sum_{\hat{\mathbf{a}}_m}\left(\hat{\mathbf{a}}_m-\mathrm{Accel}(u_m)\right)^2_{\Sigma_\mathbf{a}},\end{aligned} \tag{11}$$

最后，通过最小化观测值与测量值的误差函数来估计曲线参数。

基于累积3次B-Spline的SE3插值



- https://github.com/Zihan-Wang/IMUsimulation
- http://udel.edu/~pgeneva/downloads/notes/2018_notes_mueffler2017arxiv.pdf
- Continuous-Time Visual-Inertial Odometry for Event Cameras](https://ieeexplore.ieee.org/abstract/document/8432102/)
- A Spline-Based Trajectory Representation for Sensor Fusion and Rolling Shutter Cameras]
  (https://link.springer.com/article/10.1007/s11263-015-0811-3)

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening