深蓝学院 VIO 第二次课程作业

温焕宇

2019.6.18

1 题一

1.1 设置 IMU 仿真代码中不同参数，生成 Allen 方差标定曲线

答：使用工具：ros 版本的 IMU 数据生成包、港科大的 imu-utils。ubuntu16.04。matlab2018b

步骤：

1. 安装 ubuntu16.04 版本对应的 ros,参考官网：`http://roswiki.autolabor.com.cn/cn(2f)kinetic(2f)Installation(2f)Ubuntu.html`

2. 创建 imu.bag。

首先创建工作空间：在终端输入

```
mkdir vio_sim_ws
cd vio_sim_w/
mkdir src
cd src/
// 将课程下载的包放入
cd ..
catkin_make
```

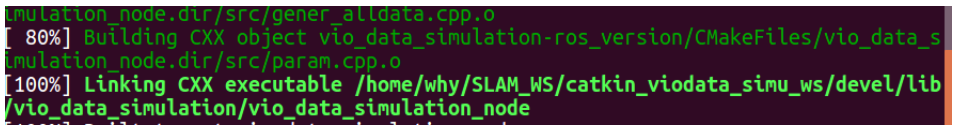

图 1: 编译完成截图

执行生成 rosbag 包：先打开一个终点输入 roscore，然后在工作空间目录下再打开一个在输入

```
source devel/setup.bash
rosrun vio_data_simulation vio_data_simulation_node
```

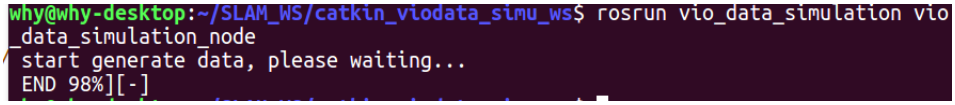

图 2: 运行完成截图

3. 分析 bag：

创建工作空间：在终端中输入

```
mkdir imu_utils
cd imu_utils/
mkdir src
cd src/
git clone https://github.com/gaowenliang/code_utils
cd ..
catkin_make
cd src/
git clone https://github.com/gaowenliang/imu_utils.git
cd ..
catkin_make
```

执行 node，进入到工作空间在终端输入

```
source devel/setup.bash
roslaunch imu_utils imu_test.launch
```

在刚才生成的 bag 文件夹下打开终端输入以下，回放 imu 数据

```
rosbag play -r 200 imu.bag
```

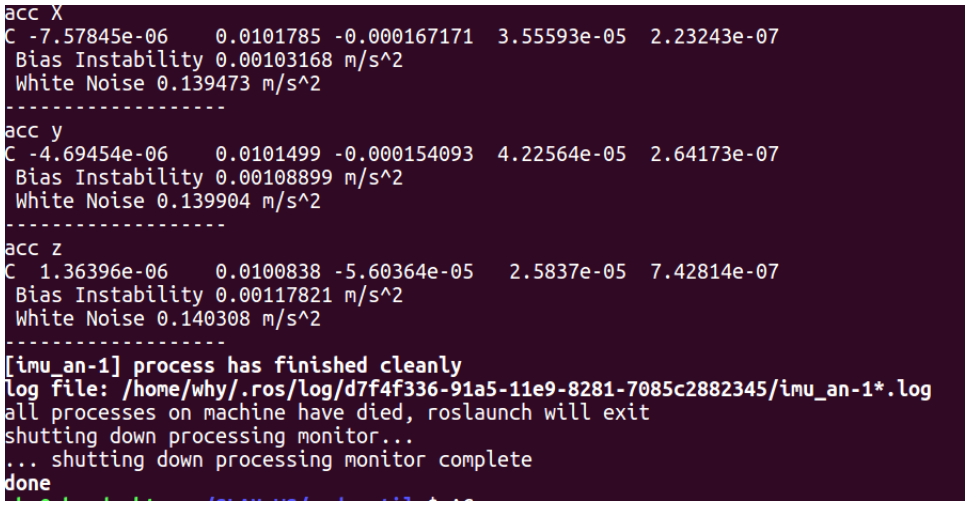

图 3: imu-utils node 运行完成截图

【注意两点：1. 这里应该先将 code-utils 放入工作空间进行编译，然后再放入 imu-utils 放入编译，否则找不到 code-utils 2. 将 vio-data 下的 genr-alldata.cpp 文件下的 imu 话题名称前面加/，然后修改 launch 文件话题订阅名称为：/imu】

4. 画 Allen 曲线图

使用 imu-utils 里带的 matlab 文件画 allen 曲线，注意修改 gypr 单位，由 deg/h 改为 rad/s。

本次试验修改两次数据，yaml 输出结果基本和设置值相近，但是会高一个数量级。

【问题】: Allen 曲线和 yaml 结果不是很匹配。。。。

i. noise 和 bias 设置如图:

```
// noise
double gyro_bias_sigma = 0.00005;
double acc_bias_sigma = 0.00005;

//double gyro_bias_sigma = 1.0e-5;
//double acc_bias_sigma = 0.0001;

double gyro_noise_sigma = 0.010;    // rad/s
double acc_noise_sigma = 0.010;      // m/(s^2)

double pixel_noise = 1;              // 1 pixel noise
```

图 4: 设定的 noise 及 bias

输出结果如下:

```
%YAML:1.0
---
type: IMU
name: imu_sim
Gyr:
   unit: " rad/s"
   avg-axis:
      gyr_n: 1.4192898330973605e-01
      gyr_w: 8.1367081730533480e-04
Acc:
   unit: " m/s^2"
   avg-axis:
      acc_n: 1.4145198458415834e-01
      acc_w: 8.6941884691120084e-04
```

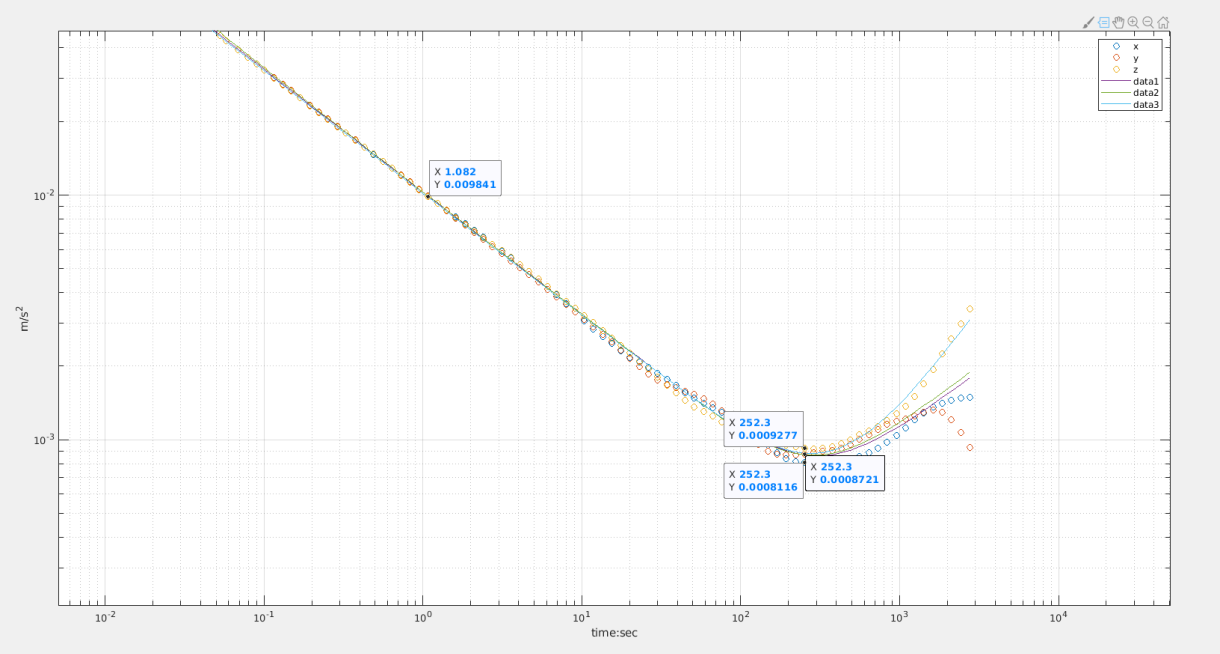

图 5: 加速度 Allen 曲线

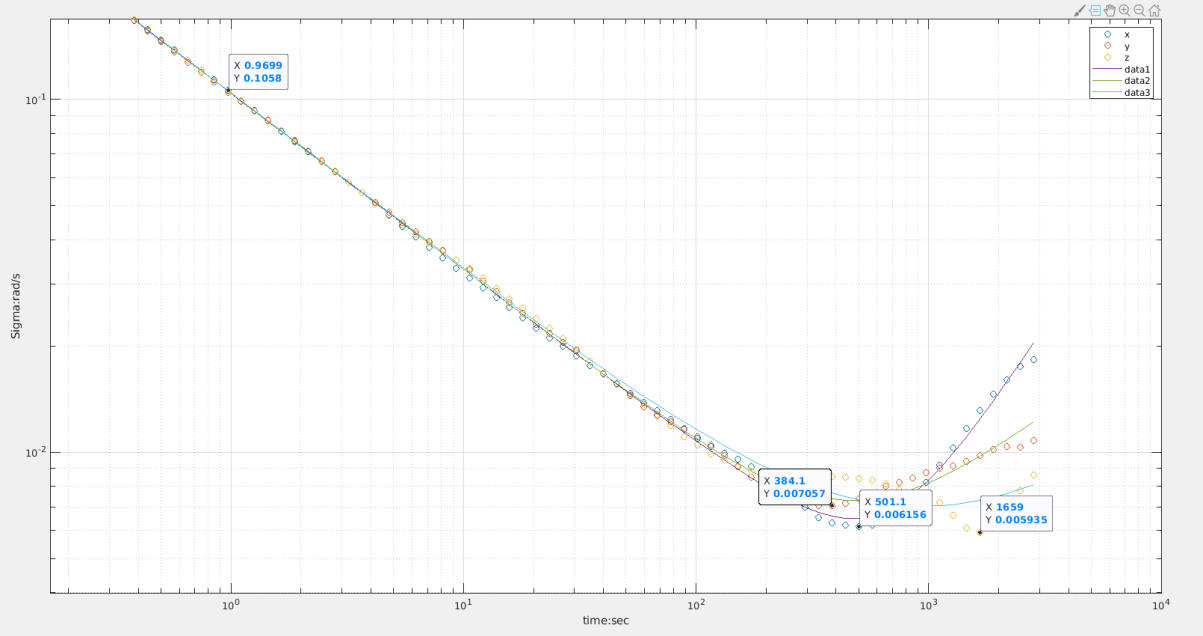

图 6: 陀螺仪 Allen 曲线

ii. noise 和 bias 设置如图:

```
// noise
double gyro_bias_sigma = 0.00001;
double acc_bias_sigma = 0.0001;

//double gyro_bias_sigma = 1.0e-5;
//double acc_bias_sigma = 0.0001;

double gyro_noise_sigma = 0.030;     // rad/s
double acc_noise_sigma = 0.010;      // m/(s^2)
```

图 7: 设定的 noise 及 bias

输出结果如下:

```
%YAML:1.0
---
type: IMU
name: imu_sim
Gyr:
   unit: " rad/s"
   avg-axis:
      gyr_n: 4.1927943422936353e-01
      gyr_w: 7.1326873851905220e-04
Acc:
   unit: " m/s^2"
   avg-axis:
      acc_n: 1.3989513145889299e-01
      acc_w: 1.0996256048798265e-03
```

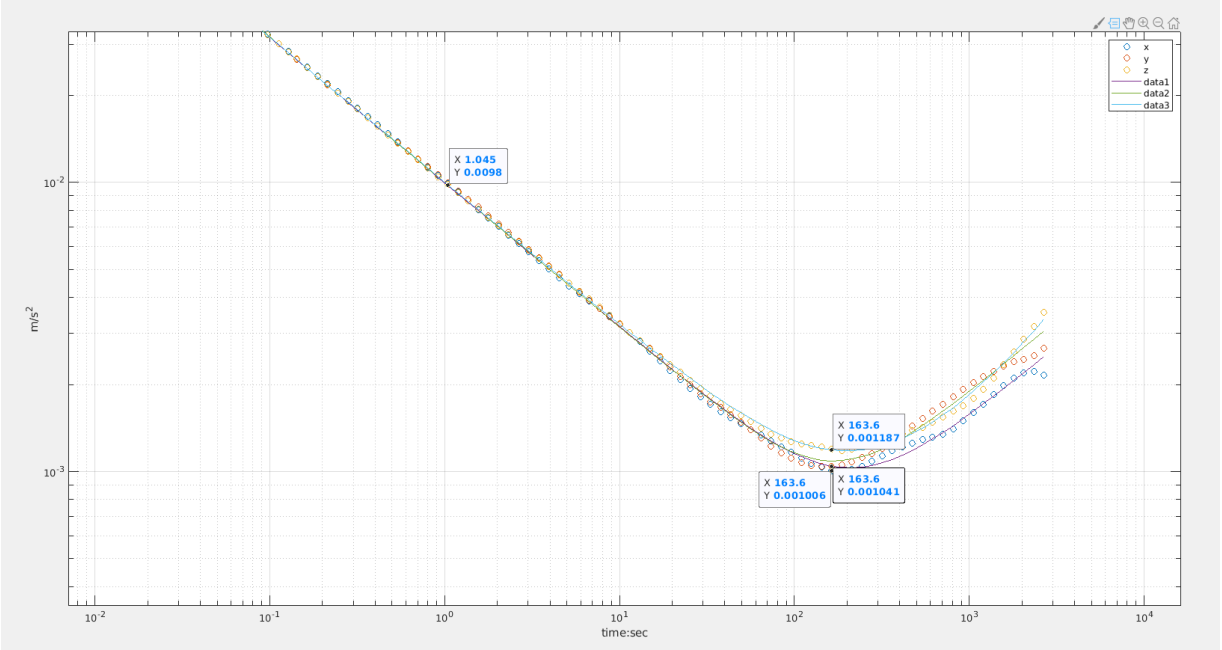

图 8: 加速度 Allen 曲线

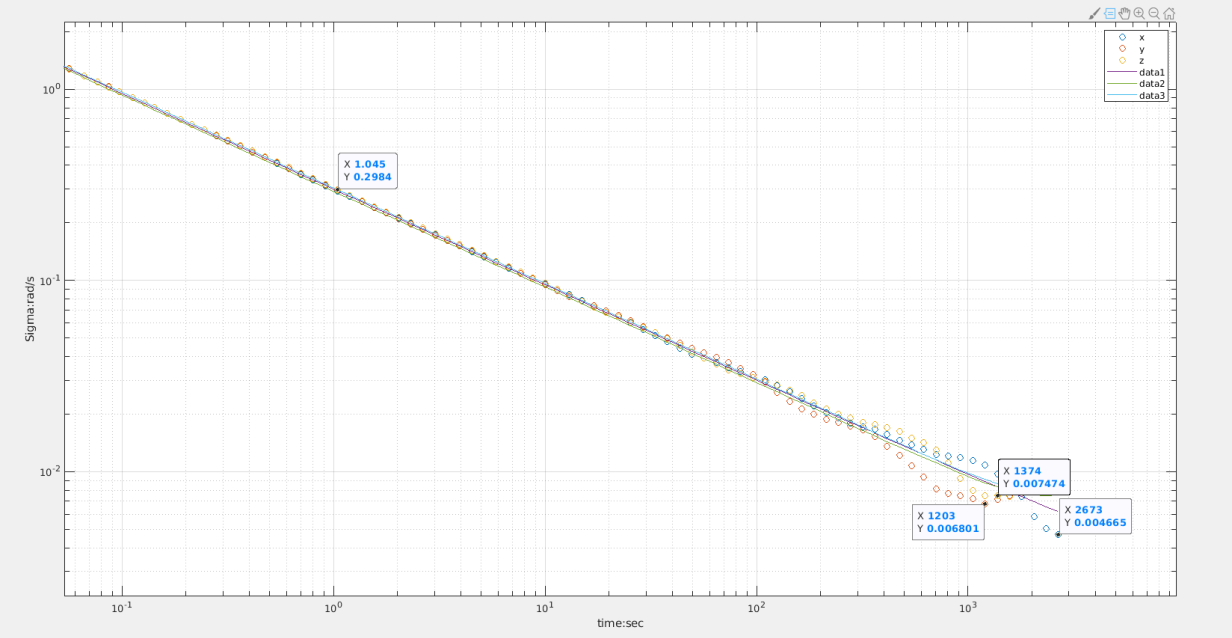

图 9: 陀螺仪 Allen 曲线

2 题二

2.1 将 IMU 仿真代码中的欧拉积分替换成中值积分

答：本部分代码在作业提供的 vio-data-simulation-master 文件里的 imu.cpp 文件中修改。【注：原版欧拉积分并不是按照上一时刻 Qwb 更新加速度，而是当前时刻。猜测可能是工程经验调试当前时刻结果更好。在更新加速度 acc-w-1 时要先更新 Qwb 为当前时刻。】

修改代码如下：

```
void IMU::testImu(std::string src, std::string dist)
{
    std::vector<MotionData>imudata;
    LoadPose(src,imudata);
    std::ofstream save_points;
    save_points.open(dist);
    double dt = param_.imu_timestep;
    Eigen::Vector3d Pwb = init_twb_;              // position :    from  imu measurements
    Eigen::Quaterniond Qwb(init_Rwb_);            // quaterniond:  from imu measurements
    Eigen::Vector3d Vw = init_velocity_;          // velocity  :   from imu measurements
    Eigen::Vector3d gw(0,0,-9.81);    // ENU frame
    Eigen::Vector3d temp_a;
    Eigen::Vector3d theta;

    Eigen::Vector3d acc_w_0, acc_w_1, acc_w ; // 上一时刻和当前时刻及中值加速度
    Eigen::Vector3d gyro_w_0, gyro_w_1, gyro_w ; // 上一时刻和当前时刻及中值角速度
    for (int i = 1; i < imudata.size(); ++i) {

        MotionData imupose_0 = imudata[i-1]; // 上一时刻imu数据
        MotionData imupose_1 = imudata[i]; // 当前时刻imu数据
        gyro_w_0 = imupose_0.imu_gyro; // 上一时刻角速度
        gyro_w_1 = imupose_1.imu_gyro; // 当前时刻角速度
        gyro_w = 0.5 * (gyro_w_0 + gyro_w_1);// 角速度中值
        // 四元素导数
        Eigen::Quaterniond dq;
        Eigen::Vector3d dtheta_half = gyro_w * dt /2.0;
        dq.w() = 1;
        dq.x() = dtheta_half.x();
        dq.y() = dtheta_half.y();
        dq.z() = dtheta_half.z();

        /// 中值积分
        acc_w_0 = Qwb * (imupose_0.imu_acc) + gw; // 上一时刻加速度，用初始Qwb计算
        Qwb = Qwb * dq; // 更新四元数
        acc_w_1 = Qwb * (imupose_1.imu_acc) + gw; // 当前时刻加速度，用当前Qwb更新
        acc_w = 0.5 * (acc_w_0 + acc_w_1); // acc_w中值
        Vw += acc_w * dt; // 更新速度
        Pwb += Vw * dt + 0.5 * dt * dt * acc_w; // 更新位移

        /// imu 动力学模型 欧拉积分
//        Eigen::Vector3d acc_w = Qwb * (imupose.imu_acc) + gw;  //       aw = Rwb * ( acc_body - acc_bias ) + gw
//        Qwb = Qwb * dq;
//        Vw = Vw + acc_w * dt;
//        Pwb = Pwb + Vw * dt + 0.5 * dt * dt * acc_w;

        //  按着imu postion, imu quaternion , cam postion, cam quaternion 的格式存储，由于没有cam，所以
```

```
    imu存了两次
        save_points<<imupose_1.timestamp<<" "
                   <<Qwb.w()<<" "
                   <<Qwb.x()<<" "
                   <<Qwb.y()<<" "
                   <<Qwb.z()<<" "
                   <<Pwb(0)<<" "
                   <<Pwb(1)<<" "
                   <<Pwb(2)<<" "
                   <<Qwb.w()<<" "
                   <<Qwb.x()<<" "
                   <<Qwb.y()<<" "
                   <<Qwb.z()<<" "
                   <<Pwb(0)<<" "
                   <<Pwb(1)<<" "
                   <<Pwb(2)<<" "
                   <<std::endl;

    }
    std::cout<<"test  end"<<std::endl;
}
```

改为中值法精度有些许提升:

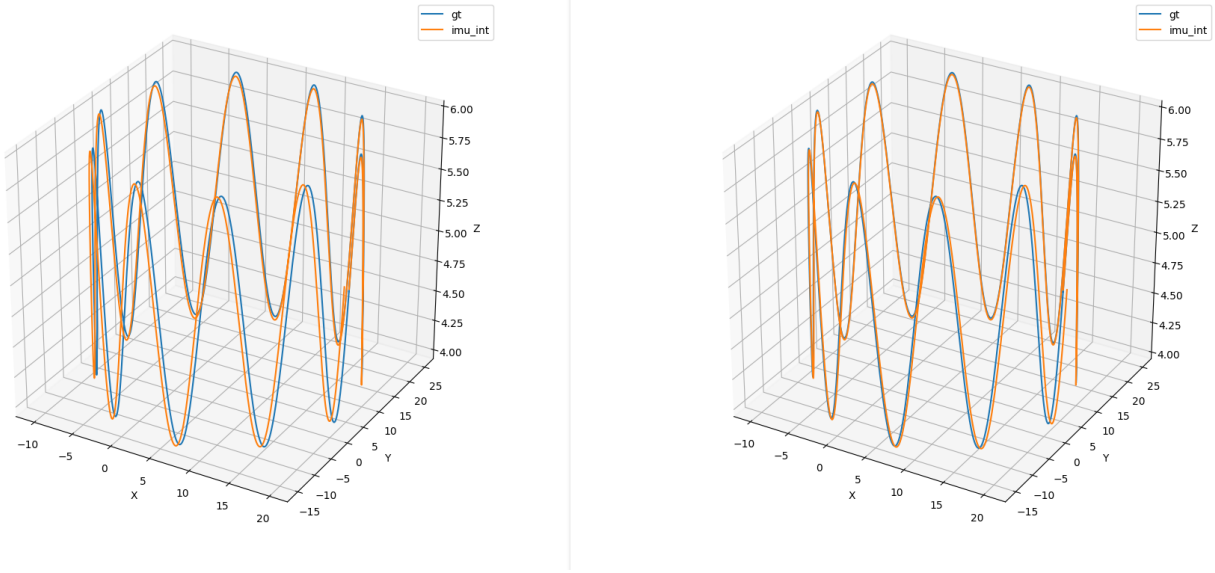

图 10: 欧拉积分（左）和中值积分（右）结果对比

3 提升作业

3.1 阅读从已有轨迹生成 IMU 了数据的论文，撰写总结推导

Spline Fusion: A continuous-timerepresentation for visual-inertial fusion withapplication to rolling shutter cameras

样条融合：视觉惯性融合的连续时间表示，适用于卷帘快门相机

答：

3.1.1 Abstract

本文描述一种通用的用于 VIO 和标定的连续时间框架。用参数化样条曲线拟合传感器空间运动轨迹。对比传统的离散时间情况，连续时间公式对于解决高帧率传感器和多个非同步设备的问题特别有用。主要工作：1、建立多个非同步设备的相对姿态和内参，证明方法在多传感器 VIO 和标定有效。2、在 VIO 中对全局和卷帘相机进行评估，证明能提高轨迹精度

【补充知识点：全局快门（global shutter）：所有像素点同时收集光线，同时曝光。如 CCD；卷帘快门（rolling shutter）：逐行曝光。不同行像元曝光时间不同】

3.1.2 Introduction

i. 利用相机轨迹的连续时间模型可以融合来自多个不同步高频传感器的信息，同时限制状态规模。

ii. 连续时间模型不仅适用于 VIO，还适用于其它传感器，如 SICK Laser。

iii. 适用于不同步的多传感器外参标定。

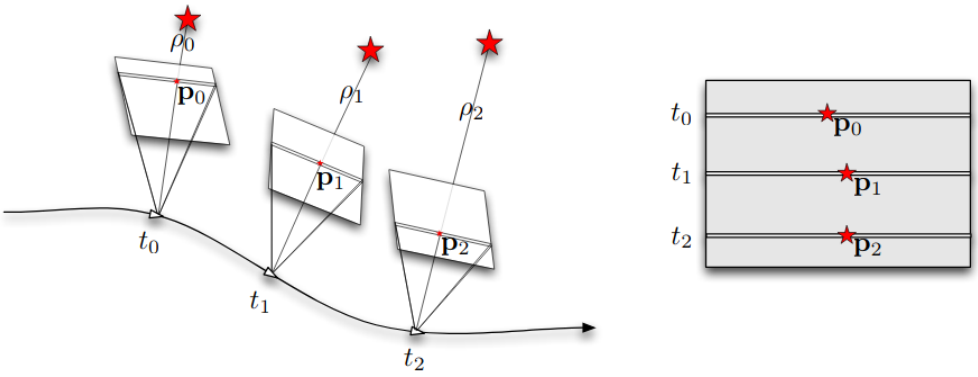

图 11: 像素位置 Pi 观察到的 LandMarks 由其对应的逆深度 p 和时刻值 ti 表示

3.1.3 Continuous-time representation

a) 流形空间下的矩阵李群 $\mathbb{SE}3$ 的李代数 $\mathfrak{se}3$ 表示。仅局部帧，不是全局帧。b) 这种参数化没有任何奇异性，为最小扭矩轨迹提供很好近似解析。

连续时间下的轨迹公式推导如下：

相机位姿矩阵

说明：为了避免欧式空间下的奇异性，本文用流形空间下的矩阵李群 SE3 的李代数 se3 表示。

$$\mathbf{T}_{b,a} = \begin{bmatrix} \mathbf{R}_{b,a} & \mathbf{a}_b \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \mathbf{T}_{b,a} \in \mathbb{SE}3, \quad \mathbf{R}_{b,a} \in \mathbb{SO}3,$$

上式表示：同一点从 a 帧转换到 b 帧的旋转平移矩阵。

用累计基函数表示 B 样条

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

$\mathbf{p}_i$ 是在 t_i 时刻的控制点，B 是基函数（由德布尔-考克斯递归公式计算），也可以将（1）式写成下边（2）式：

$$\mathbf{p}(t) = \mathbf{p}_0 \tilde{B}_{0,k}(t) + \sum_{i=1}^{n} (\mathbf{p}_i - \mathbf{p}_{i-1}) \tilde{B}_{i,k}(t) \tag{2}$$

为了使用对数和指数映射描述，重写（2）式来描述 $\mathbb{SE}3$ 下的轨迹，通过控制点间对数映射 $\Omega_i = \log(\mathbf{T}_{w,i-1}^{-1}\mathbf{T}_{w,i})$ 代替控制点差 $p_i - p_{i-1}$，得到下式（3）：

$$\mathbf{T}_{w,s}(t) = \exp(\tilde{B}_{0,k}(t)\log(\mathbf{T}_{w,0})) \prod_{i=1}^{n} \exp(\tilde{B}_{i,k}(t)\Omega_i), \tag{3}$$

其中 $\mathbf{T}_{w,s}(t) \in \mathbb{SE}3$ 是 t 时刻沿着样条曲线的 pose。$\mathbf{T}_{w,i} \in \mathbb{SE}3$ 是控制点的位姿，下标 w 表示世界坐标系。

累计三次 B 样条（k=4）

定义控制点为 $[t_{i-1}, t_i, t_{i+1}, t_{i+2}]$,$t \in [t_i, t_{i+1}]$，将德布尔-考克斯递归公式写成矩阵形式：

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

在样条轨迹下的 pose 可以被定义为（4）：

$$\mathbf{T}_{w,s}(u) = \mathbf{T}_{w,i-1} \prod_{j=1}^{3} \exp\left(\tilde{\mathbf{B}}(u)_j \Omega_{i+j}\right), \tag{4}$$

其中下标 i 由 $u(t)$ 间隔定义，给定时间 $s_i \leqslant s(t) < s_{i+1}$，定义 $u(t) = s(t) - s_i, s(t) = (t - t_0)/\delta t$。$s_i \in [0, 1, 2..., n]$。下标 j 是 B 的索引。记 $\widetilde{B}(u)_0 = 1, \forall u$。这样就可以用下式表示位姿对时间的一阶二阶导数（5）（6）：

$$\begin{aligned}
\dot{\mathbf{T}}_{w,s}(u) &= \mathbf{T}_{w,i-1}\left(\dot{\mathbf{A}}_0\mathbf{A}_1\mathbf{A}_2 + \mathbf{A}_0\dot{\mathbf{A}}_1\mathbf{A}_2 + \mathbf{A}_0\mathbf{A}_1\dot{\mathbf{A}}_2\right), && (5)\\
\ddot{\mathbf{T}}_{w,s}(u) &= \mathbf{T}_{w,i-1}\begin{pmatrix}\ddot{\mathbf{A}}_0\mathbf{A}_1\mathbf{A}_2 + \mathbf{A}_0\ddot{\mathbf{A}}_1\mathbf{A}_2 + \mathbf{A}_0\mathbf{A}_1\ddot{\mathbf{A}}_2 + \\ 2\left(\dot{\mathbf{A}}_0\dot{\mathbf{A}}_1\mathbf{A}_2 + \dot{\mathbf{A}}_0\mathbf{A}_1\dot{\mathbf{A}}_2 + \mathbf{A}_0\dot{\mathbf{A}}_1\dot{\mathbf{A}}_2\right)\end{pmatrix}, && (6)\\
\mathbf{A}_j &= \exp\left(\Omega_{i+j}\tilde{\mathbf{B}}(u)_j\right), \quad \dot{\mathbf{A}}_j = \mathbf{A}_j\Omega_{i+j}\dot{\tilde{\mathbf{B}}}(u)_j,\\
\ddot{\mathbf{A}}_j &= \dot{\mathbf{A}}_j\Omega_{i+j}\dot{\tilde{\mathbf{B}}}(u)_j + \mathbf{A}_j\Omega_{i+j}\ddot{\tilde{\mathbf{B}}}(u)_j
\end{aligned}$$

3.1.4 Generative model of visual-inertial data

参数化模型

使用从沿着样条曲线的第一个观测姿态的逆深度来参数化系统的 landmarks。在 a 帧投影的逆深度点转换到帧 b 的图像坐标系下（7）：

$$\mathbf{p}_b = \mathcal{W}(\mathbf{p}_a; \mathbf{T}_{b,a}, \rho) = \pi\left(\left[\mathbf{K}_b \,|\, \mathbf{0}\right]\mathbf{T}_{b,a}\begin{bmatrix}\mathbf{K}_a^{-1}\begin{bmatrix}\mathbf{p}_a\\1\end{bmatrix};\rho\end{bmatrix}\right), \tag{7}$$

其中 π 表示均匀投影函数，k 表示内参。

累积 B 样条参数化可以计算分析时间导数，如式（5）（6）。这样可以轻松地合成加速度计和陀螺仪测量，我们可以使用这些测量来形成观察到的测量误差（8）（9）：

$$\begin{aligned}
\text{Gyro}(u) &= \mathbf{R}_{w,s}^{\intercal}(u)\cdot\dot{\mathbf{R}}_{w,s}(u) + \text{bias}, && (8)\\
\text{Accel}(u) &= \mathbf{R}_{w,s}^{\intercal}(u)\cdot(\ddot{\mathbf{s}}_w(u) + g_w) + \text{bias}, && (9)
\end{aligned}$$

其中 $\dot{R}$ 和 $\ddot{S}$ 是 $\dot{T}$ 和 $\ddot{T}$ 的子矩阵，g_w 是重力加速度。

最小化误差

通过最小化由测量值与预测观测值的差形成的目标函数，来批量或在滑窗内求解样条和相机参数。

通过使用连续时间的模拟，可以统一处理重投影误差和惯性误差，并通过从设备规格或标定中计算出的各个信息矩阵进行加权。

构建如下（10）：

$$\begin{aligned}
E(\theta) &= \sum_{\hat{\mathbf{p}}_m}\left(\hat{\mathbf{p}}_m - \mathcal{W}(\mathbf{p}_r; \mathbf{T}_{c,s}\mathbf{T}_{w,s}(u_m)^{-1}\mathbf{T}_{w,s}(u_r)\mathbf{T}_{s,c}, \rho)\right)_{\Sigma_p}^2 + \\
&\quad \sum_{\hat{\omega}_m}\left(\hat{\omega}_m - \text{Gyro}(u_m)\right)_{\Sigma_\omega}^2 + \sum_{\hat{\mathbf{a}}_m}\left(\hat{\mathbf{a}}_m - \text{Accel}(u_m)\right)_{\Sigma_\mathbf{a}}^2, && (10)
\end{aligned}$$

其中视觉惯性测量值 p, w, a。下标 m 表示在时刻 u 下的测量值，u_r 表示和第一次观测的共视帧。θ 代表要优化的参数向量，包括：样条控制点、相机内参、路标点逆深度、相机和 IMU 外参、IMU 的

bias。（本文使用 Ceres 库解最小二乘）

3.1.5 Projection into a rolling shutter camera

此部分主要是用于卷帘快门相机中，将 y 轴作为时间连续参数，子像素值表示不同的时间间隔。

$$\mathbf{p}_b(t+\Delta t) = \mathcal{W}(\mathbf{p}_a; \mathbf{T}_{b,a}(t), \rho) + \Delta t \frac{d\,\mathcal{W}(\mathbf{p}_a; \mathbf{T}_{b,a}(t), \rho)}{d\,t} \tag{12}$$

$$y_{b(t+\Delta t)} = \frac{h(t+\Delta t - s)}{e-s}, \quad \Delta t = -\frac{h.t_0 + s.(y_b(t) - h) - e.y_b(t)}{(s-e)\frac{d\,\mathcal{W}_y(\mathbf{p}_a; \mathbf{T}_{b,a}(t), \rho)}{d\,t} + h} \tag{13}$$

由于不知道哪个垂直子像素对应时间及路标点下降的问题，使用恒速模型可以解决。但是式（11）与隐含的时间方程不匹配，时间方程为：$y_b(t) = h(t-s)/(e-s)$，s 是开始帧时间，e 是结束帧时间，h 是以像素为单位的高度。

本文提出以下：首先在帧间隔内一次投影地标，该帧间隔对应于我们对 y 轴位置的最佳猜测，可能是从最后一帧初始化的，然后我们在这个时间周围执行第一阶泰勒级数展开的地标投影，并解决未知时间的差异，式（12）（13）：

$$\mathbf{p}_b(t+\Delta t) = \mathcal{W}(\mathbf{p}_a; \mathbf{T}_{b,a}(t), \rho) + \Delta t \frac{d\,\mathcal{W}(\mathbf{p}_a; \mathbf{T}_{b,a}(t), \rho)}{d\,t} \tag{12}$$

$$y_{b(t+\Delta t)} = \frac{h(t+\Delta t - s)}{e-s}, \quad \Delta t = -\frac{h.t_0 + s.(y_b(t) - h) - e.y_b(t)}{(s-e)\frac{d\,\mathcal{W}_y(\mathbf{p}_a; \mathbf{T}_{b,a}(t), \rho)}{d\,t} + h} \tag{13}$$

其中在每次迭代后设置 $t \leftarrow t + \delta t$。在两三次迭代后，即使对于严重的卷帘快门产生的果冻效果，t 也可以高精度收敛到一致的投影/观察时间。

[1]

[1]