1.

40 自动驾驶中的 SLAM 技术

根据前面介绍的运动学，可以简单地把连续时间的运动学方程列写出来[1]：

$$\dot{\boldsymbol{R}} = \boldsymbol{R}\boldsymbol{\omega}^{\wedge}, \quad \text{或 } \dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q}\boldsymbol{\omega} \tag{3.1a}$$

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \tag{3.1b}$$

$$\dot{\boldsymbol{v}} = \boldsymbol{a} \tag{3.1c}$$

其中旋转部分既可以用旋转矩阵来表示，见式（2.70），也可以用四元数来表示，见式（2.72）。这些物体量带上脚标之后，应该写作 $\boldsymbol{R}_{wb}, \boldsymbol{p}_{wb}$。由于 $\boldsymbol{p}_{wb}$ 又对应车辆的世界坐标，它在求导之后就是车辆在世界坐标系下的速度与加速度 $\boldsymbol{a}_w, \boldsymbol{v}_w$。这种写法是直观的，所以后文会省略与坐标系相关的下标。其他材料里可能会定义诸如 ${}_w\boldsymbol{v}_{wb}$ 这样的变量来区分世界系速度和车体系速度，而本书统一使用世界系下的物理量，有特殊情况单独再作说明。

以上公式假设了世界系是固定不动的，类似于宇宙空间或者虚拟空间。这时 IMU 的测量值 $\tilde{\boldsymbol{\omega}}, \tilde{\boldsymbol{a}}$ 就是车辆本身的角速度，以及车体系下的加速度[2]：

$$\tilde{\boldsymbol{a}} = \boldsymbol{R}^{\mathrm{T}}\boldsymbol{a} \tag{3.2a}$$

$$\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega} \tag{3.2b}$$

注意 $\boldsymbol{R}^{\mathrm{T}}$ 带下标之后就是 $\boldsymbol{R}_{bw}$。它将世界系下的物理量转换到车体系。

然而，实际的车辆、机器人都在地球表面运行。这些系统日常受到重力的影响，我们也应该把重力写在系统方程中。在绝大多数 IMU 系统中，我们可以忽略地球自转的干扰[3]，从而把 IMU 测量值写为：

感觉这个内容在此处前后没有联系。

2.

因此，在离散时间系统中（也是我们平时操作的系统），两个噪声都是非常便于处理的。而且，在很多系统实现中，甚至不考虑用**协方差矩阵**来表达 IMU 测量噪声和零偏随机游走，而是简单地将它们表达为**对角矩阵**，这实际上忽略了各个轴之间的相关性。在程序里，我们往往使用诸如 σ_g, σ_a 的参数来表达 IMU 的噪声标准差，用 σ_{bg}, σ_{ba} 参数来表达零偏游走的**标准差**。这时，离散时间下的噪声标准差应该写为：

$$\sigma_g(k) = \frac{1}{\sqrt{\Delta t}}\sigma_g, \quad \sigma_a(k) = \frac{1}{\sqrt{\Delta t}}\sigma_a \tag{3.10a}$$

$$\sigma_{bg}(k) = \sqrt{\Delta t}\sigma_{bg}, \quad \sigma_{ba}(k) = \sqrt{\Delta t}\sigma_{ba} \tag{3.10b}$$

在后文介绍滤波器和预积分时，我们会使用这些符号来配置 IMU 的噪声情况。从物理单位上来看，离散时间的噪声是直接加在被测的物理量上的，很容易确定它们的物理单位。而离散时间的

[1] 需要舍掉 [49] 中的一些小项。

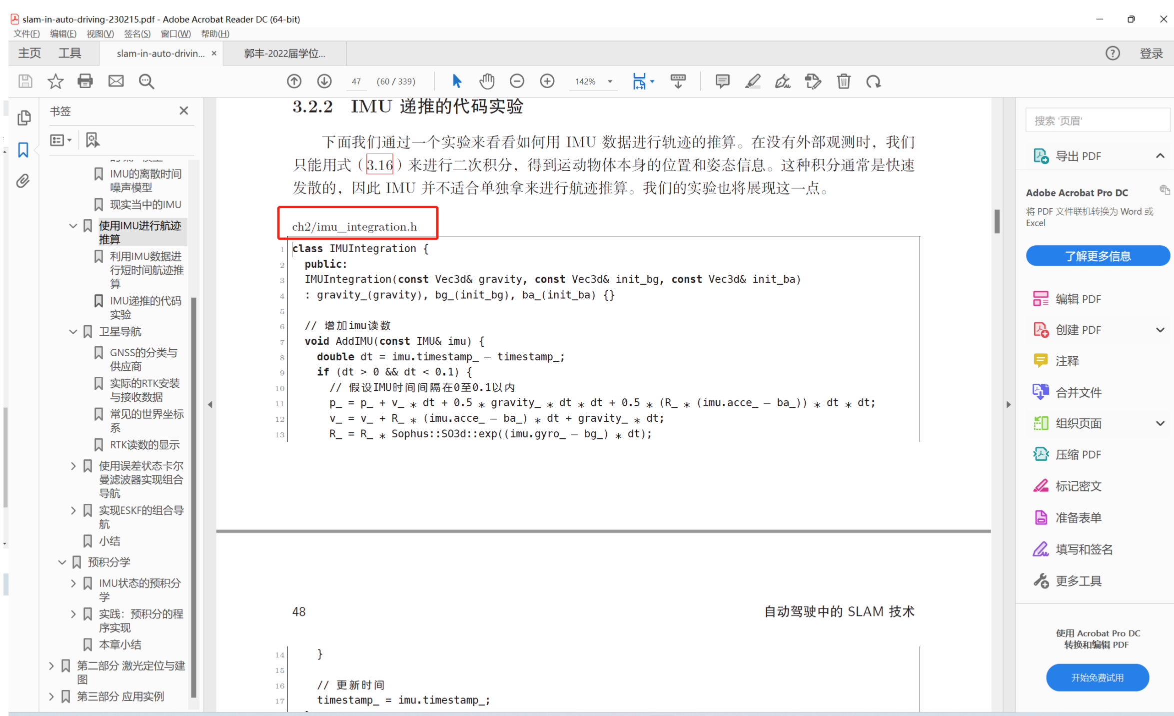
零偏本身是加在被测物理量上的，因此它们与被测物理量具有相同单位 [50]。

$$\sigma_g(k) \to \frac{rad}{s}, \quad \sigma_a(k) \to \frac{m}{s^2}, \quad \sigma_{bg}(k) \to \frac{rad}{s}, \quad \sigma_{ba}(k) \to \frac{m}{s^2} \tag{3.11}$$

而连续时间的方差需要在离散方差上乘或除以一个平方时间单位，因此它们的物理单位变为：

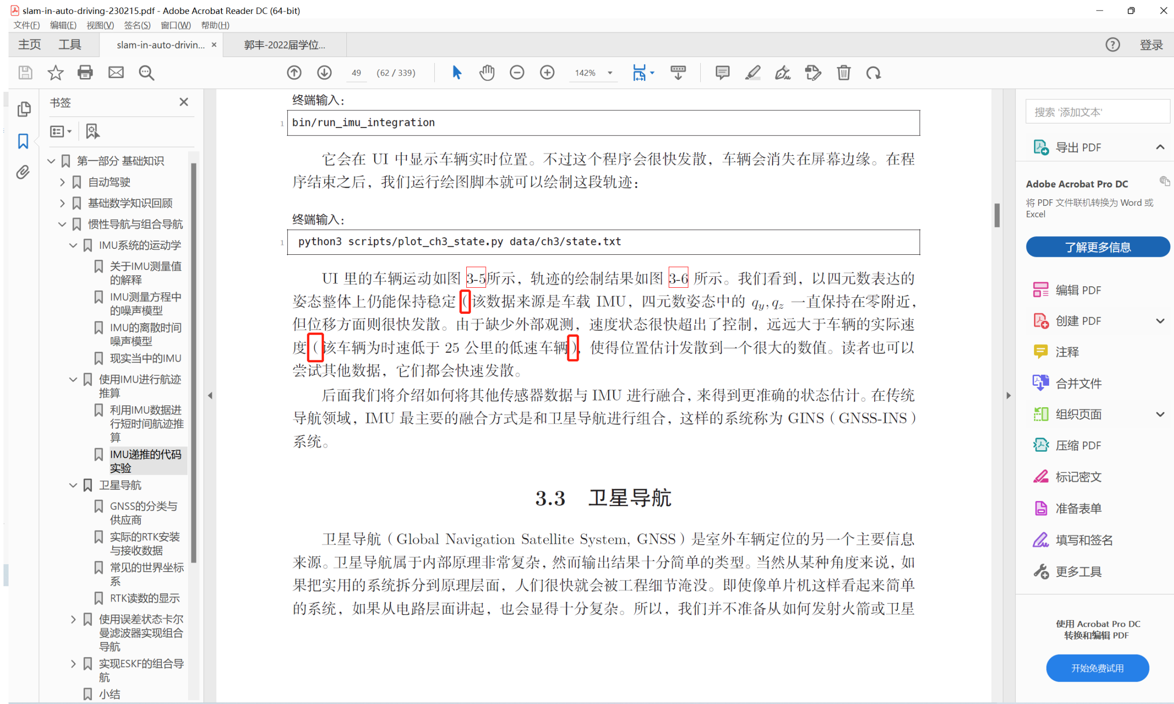
应该是测量噪声，这两句话直接用一句：直接加在被测物理量的单位和被测物理量一致。

3.



应该是ch3

4.


终端输入：

```
bin/run_imu_integration
```

它会在 UI 中显示车辆实时位置。不过这个程序会很快发散，车辆会消失在屏幕边缘。在程序结束之后，我们运行绘图脚本就可以绘制这段轨迹：

终端输入：

```
python3 scripts/plot_ch3_state.py data/ch3/state.txt
```

UI 里的车辆运动如图 3-5所示，轨迹的绘制结果如图 3-6 所示。我们看到，以四元数表达的姿态整体上仍能保持稳定（该数据来源是车载 IMU，四元数姿态中的 q_y, q_z 一直保持在零附近，但位移方面则很快发散。由于缺少外部观测，速度状态很快超出了控制，远远大于车辆的实际速度（该车辆为时速低于 25 公里的低速车辆），使得位置估计发散到一个很大的数值。读者也可以尝试其他数据，它们都会快速发散。

后面我们将介绍如何将其他传感器数据与 IMU 进行融合，来得到更准确的状态估计。在传统导航领域，IMU 最主要的融合方式是和卫星导航进行组合，这样的系统称为 GINS（GNSS-INS）系统。

3.3 卫星导航

卫星导航（Global Navigation Satellite System, GNSS）是室外车辆定位的另一个主要信息来源。卫星导航属于内部原理非常复杂，然而输出结果十分简单的类型。当然从某种角度来说，如果把实用的系统拆分到原理层面，人们很快就会被工程细节淹没。即使像单片机这样看起来简单的系统，如果从电路层面讲起，也会显得十分复杂。所以，我们并不准备从如何发射火箭或卫星

少了右括号

5.

---

1. GNSS 提供的定位精度是否满足要求？
2. GNSS 的定位频率是否足够下游使用？
3. GNSS 的定位可用性如何？是否能够全天候全场地使用？

实际上，GNSS 也存在许多个细分种类，对上述问题的答案也并没有统一的回答。有些 GNSS 定位方法可以提供很高精度，但必须要求物体静止一段时间（通常十分钟以上）；也有的方法可以提供较好的动态物体定位，但需要事先架设一个或多个基站。几乎所有 GNSS 都存在“看天吃饭”的问题——它们的稳定性与场景、结构甚至的当天的天气有关，使得在自动驾驶行业中，卫星导航大多数时候扮演一种精度够用，但稳定性很难控制的状态。

3.3.1　GNSS 的分类与供应商

整体而言，GNSS 系统通过测量自身与地球周围各卫星的距离来确定自身的位置，而与卫星的距离主要是通过测量时间间隔来确定的。一个卫星信号从卫星上发出时，带有一个发送时间。而 GNSS 接收机接收到它时，又有一个接收时间。我们通过比较自身接收时间与卫星发送时间，就能估算各卫星离自身距离。而各种 GNSS 系统或测量方法的主要差异，就是如何减少这个时间的误差。从这种角度来看，GNSS 本质上可以看成一种高精度的授时系统。

多余

6.

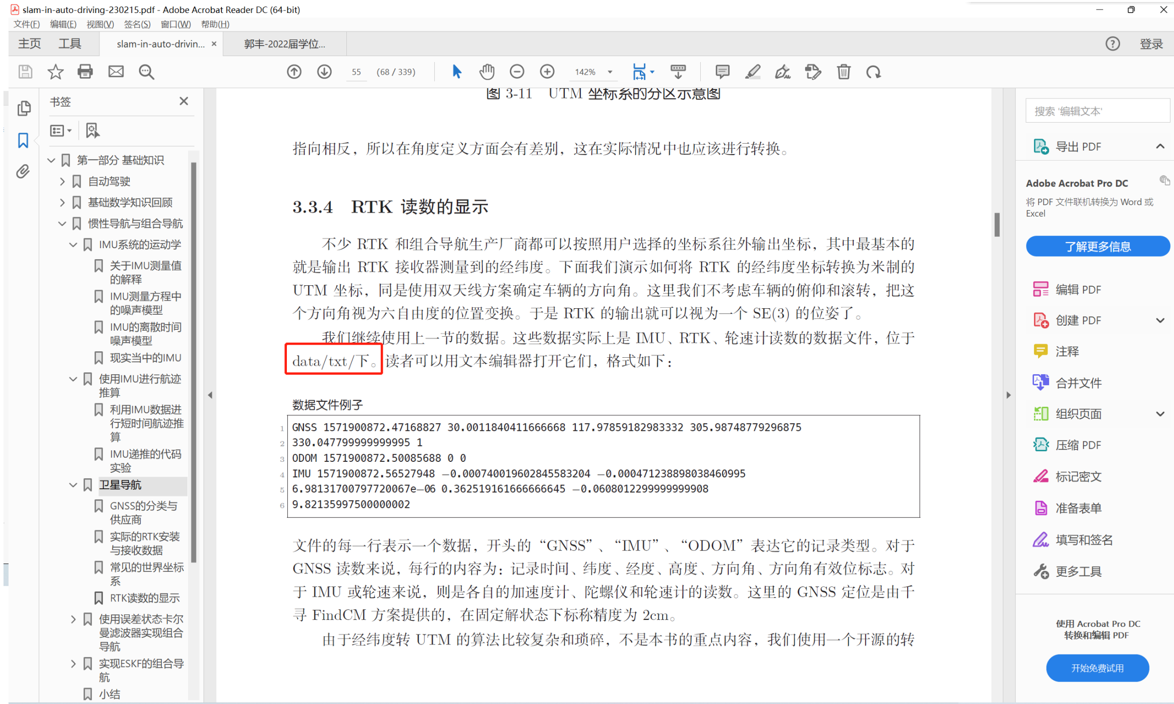


图 3-11 UTM 坐标系的分区示意图

指向相反，所以在角度定义方面会有差别，这在实际情况中也应该进行转换。

3.3.4 RTK 读数的显示

不少 RTK 和组合导航生产厂商都可以按照用户选择的坐标系往外输出坐标，其中最基本的就是输出 RTK 接收器测量到的经纬度。下面我们演示如何将 RTK 的经纬度坐标转换为米制的 UTM 坐标，同是使用双天线方案确定车辆的方向角。这里我们不考虑车辆的俯仰和滚转，把这个方向角视为六自由度的位置变换。于是 RTK 的输出就可以视为一个 SE(3) 的位姿了。

我们继续使用上一节的数据。这些数据实际上是 IMU、RTK、轮速计读数的数据文件，位于 $\boxed{\text{data/txt/下。}}$读者可以用文本编辑器打开它们，格式如下：

数据文件例子

```
GNSS 1571900872.47168827 30.0011840411666668 117.97859182983332 305.98748779296875
330.047799999999995 1
ODOM 1571900872.50085688 0 0
IMU 1571900872.56527948 −0.000740019602845583204 −0.000471238898038460995
6.98131700797720067e−06 0.362519161666666645 −0.0608012299999999908
9.82135997500000002
```

文件的每一行表示一个数据，开头的“GNSS”、“IMU”、“ODOM”表达它的记录类型。对于 GNSS 读数来说，每行的内容为：记录时间、纬度、经度、高度、方向角、方向角有效位标志。对于 IMU 或轮速来说，则是各自的加速度计、陀螺仪和轮速计的读数。这里的 GNSS 定位是由千寻 FindCM 方案提供的，在固定解状态下标称精度为 2cm。

由于经纬度转 UTM 的算法比较复杂和琐碎，不是本书的重点内容，我们使用一个开源的转

有误

7. 截距。对于平面上的点 $\boldsymbol{p}$，它与平面参数之间的关系为：

$$\boldsymbol{n}^T\boldsymbol{p} + d = 0 \tag{7.5}$$

而对于平面外的一个点，它离平面的距离可以写为 $\boldsymbol{n}^T\boldsymbol{p}+d$，于是我们可以建立 $\boldsymbol{q}_i$ 与它最近邻点构成的那个平面之间的误差函数：

$$e_i = \boldsymbol{n}^T(\boldsymbol{R}\boldsymbol{q}_i + \boldsymbol{t}) + d \tag{7.6}$$

224 自动驾驶中的 SLAM 技术

注意由于 $|\boldsymbol{n}| = 1$，这里的距离不必再除以 $|\boldsymbol{n}|$。同时，这是一个有向误差，可能产生正值或负值。很容易求出它对 $\boldsymbol{R}$ 和 $\boldsymbol{t}$ 的导数：

$$\frac{\partial e}{\partial \boldsymbol{R}} = -\boldsymbol{n}^T\boldsymbol{R}\boldsymbol{q}_i^\wedge, \quad \boxed{\frac{\partial e}{\partial \boldsymbol{t}} = \boldsymbol{n}^T} \tag{7.7}$$

另一方面，如果 $\boldsymbol{q}_i$ 的最近邻构成了直线，我们可以设直线方程为：

$$\boldsymbol{p} = \boldsymbol{d}\tau + \boldsymbol{p}_0 \tag{7.8}$$

其中 $\boldsymbol{d}$ 是单位长度的方向矢量，$\boldsymbol{p}_0$ 是直线上一点，τ 为参数。对于不在直线上的某个点 $\boldsymbol{q}_i$，我们利用叉乘的定义，将它到直线的垂直矢量长度作为误差函数。注意到两个矢量叉乘长度即为垂直距离，所以我们优化叉乘矢量即可：

没有转置