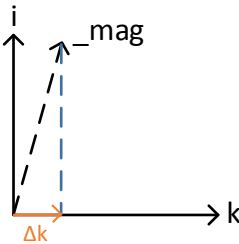


# PX4 四元数姿态估计算法

代码位置 PX4/Firmware/src/.../Attitude\_estimate\_q\_main.cpp)

- 1. int AttitudeEstimatorQ::start() 程序启动函数
  - 2. void AttitudeEstimatorQ::task\_main() 进程入口
  - 3. float gyro[3] 获取传感器数据，DataValidatorGroup，并检验数据可靠度
  - 4. 通过 uORB 模型获取 vision 和 mocap 的数据
  - 5. 位置加速度获取 (\_pos\_acc)
  - 6. 对四元数向量\_q 进行初始化赋值与更新 update
  - 7. 将\_q 转换成欧拉角 euler 并发布 \_q.to\_euler
- 对两个坐标系进行实时的标定和修正。因为坐标系有三个轴，偏航 yaw 修正由电子罗盘（基于载体）、地磁（基于地理）对比修正误差补偿得到。俯仰 pitch 和横滚 roll 上的修正由加速度计（基于载体）、重力（基于地理）对比修正误差得到。

<div>Vector&lt;3&gt; k = -_accel;</div> <div>k.normalize();</div> <div>Vector&lt;3&gt; i = (_mag - k * (_mag * k));</div> <div>i.normalize();</div>	<div>四元数初始化:</div> <div>仅用加速度计和磁力计生成姿态四元数矩阵 。</div> <div>加速度计测量的是目标向量与 b 系三轴的夹角，假设加速度计输出为</div> <div><math display="block">\_accel = [a_1, a_2, a_3]^T, z \text{ 轴向上; 加速度计能感应重力, 重力向量指向地心, 由于定义的坐标系为 NED, } z \text{ 轴向下; } k \text{ 为加速度计测量到的加速度方向向量,}</math><math display="block">k = -[a_1, a_2, a_3]^T</math></div> <div>归一化: <math display="block">k = -[a_1, a_2, a_3]^T / \sqrt{a_1^2 + a_2^2 + a_3^2}</math><math display="block">= [k_1, k_2, k_3]^T</math></div> <div>磁力计可以感应地理的北极。假设磁力计的输出为 <math>\_mag = [m_1, m_2, m_3]^T</math>。</div> <div></div> <div>为防止磁力计向量与加速度向量 k 不垂直，计算 <math>\Delta k = k \cdot (\_mag \cdot k) = m_1 a_1'^2 + m_2 a_2'^2 + m_3 a_3'^2</math> 为 <math>\_mag</math> 在 k 轴上的投影，</div> <div>通过 <math>i = \_mag - \Delta k</math></div> <div><math display="block">= [m_1 - m_1 a_1'^2, m_2 - m_2 a_2'^2, m_3 - m_3 a_3'^2]^T</math></div> <div>可以得到一个与 k 轴垂直的向量。并进行正交化后得到指向北的单位向量。 <math>i = [i_1, i_2, i_3]^T</math></div>
---	--

<div data-bbox="150 197 384 232" data-label="Text"> <p>Vector&lt;3&gt; j = k % i;</p> </div> <div data-bbox="177 488 363 524" data-label="Text"> <p>Matrix&lt;3, 3&gt; R;</p> </div> <div data-bbox="177 696 363 813" data-label="Text"> <p>R.set_row(0, i); R.set_row(1, j); R.set_row(2, k);</p> </div> <div data-bbox="150 943 347 978" data-label="Text"> <p>_q.from_dcm(R);</p> </div> <div data-bbox="150 1899 440 1935" data-label="Text"> <p>Quaternion decl_rotation;</p> </div>	<div data-bbox="786 197 1460 313" data-label="Text"> <p>加速度计得出指向下 D 的 z 轴 k，磁力计得到指向北 N 的 x 轴 I,使用右手定则，可得到 <math>j = k \times i</math> 垂直于 i、k 并指向东 E。</p> </div> <div data-bbox="810 320 1299 358" data-label="Equation-Block"> <math display="block">j = [k_2 i_3 - k_3 i_2, k_3 i_1 - k_1 i_3, k_1 i_2 - k_2 i_1]^T</math> </div> <div data-bbox="799 488 1066 524" data-label="Text"> <p>定义个 <math>3 \times 3</math> 的矩阵</p> </div> <div data-bbox="1002 539 1246 633" data-label="Equation-Block"> <math display="block">R = \begin{bmatrix} c_{11} &amp; c_{12} &amp; c_{13} \\ c_{21} &amp; c_{22} &amp; c_{23} \\ c_{31} &amp; c_{32} &amp; c_{33} \end{bmatrix}</math> </div> <div data-bbox="786 696 1460 772" data-label="Text"> <p>给矩阵的第一行赋值，将向量 i 的三个元素值依次赋给 <math>c_{11}, c_{12}, c_{13}</math></p> </div> <div data-bbox="786 779 1114 815" data-label="Equation-Block"> <math display="block">c_{11} = i_1; \quad c_{12} = i_2; \quad c_{13} = i_3</math> </div> <div data-bbox="786 898 1225 1025" data-label="Text"> <p>将矩阵转化为四元数表示: <math>q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}</math></p> </div> <div data-bbox="799 1039 1171 1072" data-label="Section-Header"> <h3>3.6.5.1 用方向余弦表示四元数</h3> </div> <div data-bbox="842 1095 1380 1128" data-label="Text"> <p>对于小角度位移,四元数参数可以用下面的关系式推导:</p> </div> <div data-bbox="1070 1140 1353 1388" data-label="Equation-Block"> <math display="block">\begin{aligned} a &amp;= \frac{1}{2}(1 + c_{11} + c_{22} + c_{33})^{1/2} \\ b &amp;= \frac{1}{4a}(c_{32} - c_{23}) \\ c &amp;= \frac{1}{4a}(c_{13} - c_{31}) \\ d &amp;= \frac{1}{4a}(c_{21} - c_{12}) \end{aligned}</math> </div> <div data-bbox="815 1400 1141 1435" data-label="Text"> <p>令 <math>tr(\text{迹}) = c_{11} + c_{22} + c_{33}</math></p> </div> <div data-bbox="855 1442 975 1476" data-label="Equation-Block"> <math display="block">s = \sqrt{tr + 1}</math> </div> <div data-bbox="855 1485 970 1518" data-label="Equation-Block"> <math display="block">q_0 = 0.5s</math> </div> <div data-bbox="855 1534 1015 1590" data-label="Equation-Block"> <math display="block">s_1 = \frac{s}{0.5} = \frac{1}{4q_1}</math> </div> <div data-bbox="855 1610 1098 1644" data-label="Equation-Block"> <math display="block">q_1 = (c_{32} - c_{23}) * s_1</math> </div> <div data-bbox="855 1653 1098 1686" data-label="Equation-Block"> <math display="block">q_2 = (c_{13} - c_{31}) * s_1</math> </div> <div data-bbox="855 1695 1098 1729" data-label="Equation-Block"> <math display="block">q_3 = (c_{21} - c_{12}) * s_1</math> </div> <div data-bbox="786 1812 1002 1848" data-label="Text"> <p>补偿磁力计偏移:</p> </div> <div data-bbox="786 1852 1286 1980" data-label="Text"> <p>定义一个磁力计偏移量四元数 <math>q_{del} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}</math></p> </div>
---	---

<pre> decl_rotation.from_yaw(_mag_decl);  _q = decl_rotation * _q; _q.normalize();  Vector&lt;3&gt; mag_earth = _q.conjugate(_mag);  float mag_err = _wrap_pi(atan2f     (mag_earth(1), mag_earth(0))     - _mag_decl); </pre>	<p>_mag_decl 可通过自动设置磁偏移（经纬度）得到。          设 <math>\_mag\_decl = [m'_1, m'_2, m'_3]^T</math>。          仅偏航后的四元数变化量：</p> $q_{del0} = \cos\left(\frac{mag\_decl}{2}\right)$ $q_{del1} = 0$ $q_{del2} = 0$ $q_{del3} = \sin\left(\frac{mag\_decl}{2}\right)$ <p>四元数更新并归一化</p> <p><b>四元数姿态更新：</b>  <b>磁力计校正：</b>          _mag 为磁力计测得的合格值(voter).          设 <math>\_mag = [m_1, m_2, m_3]^T</math>          因为磁力计测的是飞机的机体上的磁力，需要将其转换到 n 系上。这里 conjugate（）函数就是用四元数表示将一个向量从 b 系旋转到 n 系。          用四元数表示的旋转矩阵如下</p> $C(q_e^b) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$ <p>机体坐标轴向量 <math>[v_1, v_2, v_3]^T</math> 经过旋转后得到 n 系中的表示</p> $v'_1 = v_1 \cdot (q_0^2 + q_1^2 - q_2^2 - q_3^2) + v_2 \cdot 2(q_1q_2 - q_0q_3) + v_3 \cdot 2(q_0q_2 + q_1q_3)$ <p>这里 _wrap_pi(a) 是将磁力计的误差限制在 <math>[-\pi, \pi]</math> 内：          若 <math>a \in [-\pi, \pi]</math>，返回 a          若 <math>a &gt; \pi</math>，返回 <math>2\pi - a</math>          若 <math>a &lt; -\pi</math>，返回 <math>-\pi + a</math>          只取向量在地球坐标系中 x,y 的元素做正切 atan2(y,x) 并与自动获取的地磁偏移量做差得到磁力计误差纠正量。</p>
--	--

```
corr += _q.conjugate_inversed(Vector<3>(0.0f, 0.0f, -
    mag_err)) * _w_mag;
```

```
_q.normalize();
```

```
Vector<3> k(
    2.0f * (_q(1) * _q(3) - _q(0) * _q(2)),
    2.0f * (_q(2) * _q(3) + _q(0) * _q(1)),
    (_q(0) * _q(0) - _q(1) * _q(1) -
        q(2) * _q(2) + _q(3) * _q(3)));
```

```
corr += (k % (_accel - _pos_acc).normalized()) *
    _w_accel;
```

将 n 中的向量转移到 b 系。使用 conjugate\_inversed( ) 函数，旋转矩阵为  $C(q_n^b)^T$ ：

$C_n^b$

$$= \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 + q_0q_1) \\ 2(q_0q_2 + q_1q_3) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

\_w\_mag 为互补滤波中磁力计的权重。

计算磁力矫正 corr 值等于单位化的旋转矩阵 R（n 系转 b 系）乘以  $[0, 0, -\text{mag\_err}]^T$ ，相当于 b 系绕 n 系 N 轴

（Z 轴）转动  $\arctan2(\text{mag\_earth}(1), \text{mag\_earth}(0)) - \text{mag\_decl}$  度（此处不确定）

四元数归一化

### 加速度计修正：

算法的理论基础是  $[0, 0, 1]$  与姿态矩阵乘。该差值获取的重力加速度的方向是 n 系的

z 轴，加上运动加速度之后，总加速度的方向就不是与导航坐标系的天或地平行了，所以要消除这个误差，即“\_accel-\_pos\_acc”。然后又乘 z 轴向量得到误差，进行校准。

首先就是把归一化的 n 系重力加速度通过旋转矩阵  $R_n^b$  左乘旋转到 b 系，即 k 为归一化的旋转矩阵 R（n-b）的第三列。

$$k = C(q_n^b)^T \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2(q_1q_3 - q_0q_2) \\ 2(q_2q_3 + q_0q_1) \\ q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

其中  $[0, 0, 1]^T$  为地球重力向量。

\_w\_accel 为互补滤波中加速度计的权重

\_pos\_acc 为之前得到的机体运动加速度。

关于（\_accel - \_pos\_acc）：总的合力的方向 \_accel 减去机体加速度方向 \_pos\_acc 得到 g 的方向，即总加速度（加速度获取）减去机体运动加速度获取重力加速度 g，然后姿态矩阵 R 的行或列来与纯重力加速度来做叉积，算出误差。因为运动加速度是有害的干扰，必须减掉。

corr+: corr=corr\_mag + corr\_accel

```
_gyro_bias += corr * (_w_gyro_bias * dt);
```

```
for (int i = 0; i < 3; i++) {
_gyro_bias(i)=math::constrain(_gyro_bias(i),
    -_bias_max,_bias_max); }
```

```
_rates = _gyro + _gyro_bias;
```

```
corr += _rates;
```

```
_q += _q.derivative(corr) * dt;
```

## 陀螺仪的偏移估计

\_gyro\_bias 的初始值可以获取。

\_w\_gyro\_bias 为互补滤波中陀螺仪偏移的权重。陀螺仪偏移的积分量点乘加速度计和磁力计共同的校正值加上初始值得到陀螺仪总的偏移量。

对陀螺仪偏移量的每个元素进行约束 constrain( )处理：

若 \_gyro\_bias(i) < -\_bias\_max，返回 -\_bias\_max

若 \_gyro\_bias(i) ≥ -\_bias\_max，接着判断

如果 \_gyro\_bias(i) > \_bias\_max

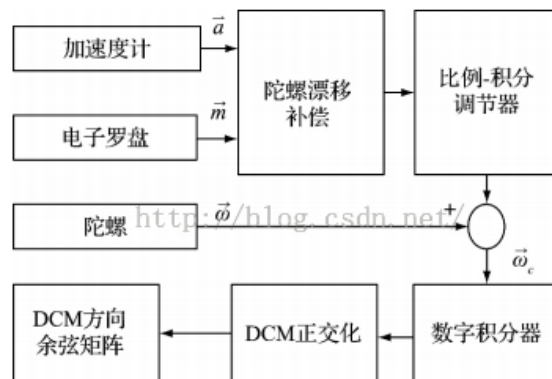
返回 \_bias\_max

如果 \_gyro\_bias(i) ≤ \_bias\_max

返回 \_gyro\_bias(i)

得到经过修正后的角速度

陀螺仪的前馈（PI 控制器的体现）



将矫正应用到四元数微分方程。

得到四元数与机体角速度的关系。

四元数微分方程（随时间的传递函数）为

$$\dot{Q} = \frac{1}{2} Q \otimes \omega_{nb}^b$$

写成矩阵形式为

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -\omega_{bx} & -\omega_{by} & -\omega_{bz} \\ \omega_{bx} & 0 & \omega_{bz} & -\omega_{by} \\ \omega_{by} & -\omega_{bz} & 0 & \omega_{bx} \\ \omega_{bz} & \omega_{by} & -\omega_{bx} & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

$$= \frac{1}{2} \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

假设在时间间隔（t,t+T）内，加速度 $\mathbf{v} = [v_1 \ v_2 \ v_3]^T$ 是一个固定值。

则四元数迭代方程为：

$$Q(k+1) = Q(k) + \dot{Q}(k) \cdot T$$

这里 derivative( $\mathbf{v}$ )的先将 $\_q$ 四元数写成矩阵形式：

$$Q = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix}$$

再将三维向量 $\mathbf{v}$ 写成增广形式：

$$\mathbf{v}' = [0 \ v_1 \ v_2 \ v_3]^T$$

返回值  $Q \cdot \mathbf{v}' / 2$  即为更新的四元数 $\dot{q}$ 。

进而 $q' = q + \dot{q}$

`_q.normalize();`

四元数归一化(得到最终的姿态四元数)

若姿态已更新，得到 $\_q$

`Vector<3> euler = _q.to_euler();`

将四元数 $\_q$ 转换成欧拉角 euler 并发布

$$\varphi_{\pm} = \arctan \frac{2(q_2 q_3 + q_0 q_1)}{q_0^2 - q_1^2 - q_2^2 + q_3^2}$$

$$\theta_{\pm} = -\arcsin 2(q_1 q_3 - q_0 q_2)$$

$$\psi_{\pm} = \arctan \frac{2(q_1 q_2 + q_0 q_3)}{q_0^2 + q_1^2 - q_2^2 + q_3^2}$$

`struct vehicle_attitude_s att = {};`  
`att.timestamp = sensors.timestamp;`

定义飞行器姿态结构体 att。

传感器时间设置为姿态时间。

`att.roll = euler(0);`  
`att.pitch = euler(1);`  
`att.yaw = euler(2);`

将获取的欧拉角赋值给 roll、pitch、yaw

<pre> att.rollspeed  = _rates(0); att.pitchspeed = _rates(1); att.yawspeed   = _rates(2);  for (int i = 0; i &lt; 3; i++) {     att.g_comp[i] = _accel(i) - _pos_acc(i); }  memcpy(&amp;att.rate_offsets, _gyro_bias.data,         sizeof(att.rate_offsets));  Matrix&lt;3, 3&gt; R = _q.to_dcm();  memcpy(&amp;att.R[0], R.data, sizeof(att.R)); att.R_valid = true;  memcpy(&amp;att.q[0], _q.data, sizeof(att.q)); att.q_valid = true;  ctrl_state.q[0] = _q(0); ctrl_state.q[1] = _q(1); ctrl_state.q[2] = _q(2); ctrl_state.q[3] = _q(3); </pre>	<p>将获取的经过修正的陀螺仪角速度 <code>rates</code> 赋值给三个欧拉角速度。</p> <p>获取 <code>n</code> 系的重力加速度的每个元素的值。</p> <p>复制偏移量 <code>memcpy ( *dest, *src, size_t n)</code>. 该函数的作用是源 <code>source</code> 中的每一个值（从第 0 个数到第 <code>n-1</code> 个）依次传送到目标 <code>dest</code>。这里就是将陀螺仪的偏移 <code>_gyro_bias</code> 赋给角速度偏移量 <code>att.rate_offsets</code></p> <p>用四元数生成 <math>3 \times 3</math> 的旋转矩阵 <code>R</code></p> $R = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$ $C_b^n = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_0q_2 + q_1q_3) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$ <p>例如 <math>c_{21} = 2(q_1q_2 + q_0q_3)</math></p> <p>复制姿态四元数</p> <p>用于状态控制的姿态四元数 在姿态控制 <code>attitude_control</code> 中会调用此四元数 代码位置： <code>PX4/Firmware/src/.../AttitudeControl.cpp</code></p>
---	--

corr 包含磁力计修正、加速度计修正、(vision、mocap 修正)、gyro(陀螺仪)中测量到的

角速度偏转量，且因为 corr 为 update 函数中定义的变量，所以每次进入 update 函数中时会刷新 corr 变量的数据；\_rate 也会刷新其中的数据，含义为三个姿态角的角速度（修正后）；

\_q 为外部定义的变量，在这个函数中只会+=不会重新赋值，如果计算出现错误会返回上一次计算出的\_q。

下表是关于上表中用到的运动加速度\_pos\_accel 和磁力计偏移\_mag\_decl 获取的补充

<pre>bool gpos_updated; orb_check(_global_pos_sub, &amp;gpos_updated);  if (gpos_updated) {     orb_copy(ORB_ID(vehicle_global_position), _global_pos_sub,             &amp;_gpos);      if (_mag_decl_auto &amp;&amp;..... ){         update_mag_declination(math::radians             (get_mag_declination(_gpos.lat, _gpos.lon))); }     }</pre>	<p>int orb_check(int handle, bool *updated)</p> <p>功能：订阅者可以用来检查一个主题在发布者上一次更新数据后，有没有订阅者调用过orb_copy 来接收、处理过；</p> <p>说明：如果主题在被公告前就有人订阅，那么这个 API 将返回not-updated 直到主题被公告。可以不用 poll，只用这个函数实现数据的获取。</p> <p>参数：</p> <p>handle:主题句柄；</p> <p>updated:如果当最后一次更新的数据被获取了，检测到并设置updated 为ture;</p> <p>返回值：</p> <p>OK 表示检测成功；错误返回 ERROR;否则则有根据的去设置 errno;</p> <p>飞机位置已更新</p> <p>orb_copy(*meta, int handle, void *buffer )</p> <p>从订阅的主题 vehicle_global_position 中获取飞行器位置信息数据保存到_gpos 中。</p> <p>*_global_pos_sub 是订阅主题返回的句柄</p> <p>首先检测是否配置了自动磁偏角获取，如果配置了则用当前的经纬度（longitude and latitude）通过 get_mag_declination(_gpos.lat,_gpos.lon)函数获取当前位置的磁偏角角度值。</p> <p>然后使用 radians(degree)函数将角度转化为弧度，返回</p> $\frac{degree}{180} \pi$ <p>最后将磁偏移量进行更新 update_mag_declination(float new_declination ),</p> <p>接下来对于这个函数进行一下解释：</p> <p>若参数未初始化或者旋转极小，</p> <p>_mag_decl = new_declination;</p>
---	--



<pre> if(_acc_comp &amp;&amp; .....){     if(gpos_updated) {          Vector&lt;3&gt;vel(_gpos.vel_n,_gpos.vel_e,_gpos.vel_d);  float vel_dt = (_gpos.timestamp -                 _vel_prev_t) / 1000000.0f;  _pos_acc = _q.conjugate_inversed(             (vel - _vel_prev) / vel_dt); }  _vel_prev_t = _gpos.timestamp; _vel_prev = vel; } </pre>	<p>否则立刻旋转现在得估计阵以避免陀螺仪偏移增长。</p> <p>然后重复上面的<b>补偿磁力计偏移</b>:</p> <p>首先检测是否设置了自动设置磁偏移。 如果位置数据信息是真的</p> <p>定义速度向量 <math>vel=[_n, _e, _d]^T</math>, 分别代表 <math>[v_x, v_y, v_z]^T</math> 计算过程在函数 <code>position_estimator_inav_main.cpp</code> 中。</p> <p>若速度更新过, 即当前时间与上次记录时间不同, 则得到时间差 <math>vel\_dt</math>=当前时间 <code>timestamp</code> - 之前时间 <code>prev_t</code>. 这里时间都会除以 1,000,000 是因为程序中的测得的硬件 <code>hrt_absolute_time()</code> 实时时间都是以 us 为单位的, 进行速度或者加速度的运算时间单位为 s。因此有这样一个换算过程。</p> <p>计算加速度速度之差除时间</p> $a = \frac{(\text{当前速度} vel - \text{之前速度} v_{prev})}{\text{时间间隔} vel\_dt}$ <p>并将其从 n 系转换到 b 系。</p> <p>进行速度、时间更新。</p>
--	--

大致结构到此结束, 还有不尽完善的地方后期会进行增改。