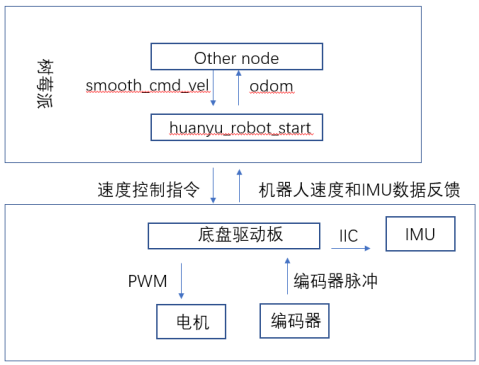
第十五课 里程计和 IMU 数据的发布

1.机器人启动节点学习

机器人启动节点源码包位于/home/huike/robot_ws/src/路径下，包名是 huanyu_robot_start，主要源码在包内的 src 目录下。这个包是非常重要的，它是 HiBot 平台接入 ROS 最重要的一个包。里程计主要功能是读取底盘驱动板的数据，发布 odom 里程计、和 imu 的等话题。在移动机器人建图和导航过程中，提供相对准确的里程计信息非常关键。

整个移动机器人的控制结构如下图所示，其中 huanyu_robot_start 节点将订阅的 smooth_cmd_vel 话题数据通过串口通信接口发送给下位机（嵌入式 stm32 控制板）。下位机中根据机器人运动学公式进行解算，将机器人速度转换为每个轮子的速度，然后通过电机驱动板控制电机转动。电机驱动板对电机转速进行闭环控制（PID 控制），并统计单位时间内接收到的编码器脉冲数，计算出轮子转速。上报到 ROS 中。



单独使用里程计来估计小车位置姿态的效果不是特别好，因为轮子打滑等情况在实际中很难避免。因此可以使用 IMU 或其它传感器来同时进行测量，如果使用 robot_pose_ekf（扩展卡尔曼滤波）对 imu 和里程计的数据进行融合，可以得到更准确的机器人位姿信息。

2.odom 话题

ROS 使用坐标变换包 tf 来求解机器人的位置以及将传感器数据与静态地图关联起来，然而 tf 软件库主要负责以 tf 坐标变换树的形式管理机器人的各种坐标系之间的关系，关于 tf 的详细教程会在相关课程中介绍。 由于 tf 中没有提供机器人的速度信息，因此，导航功能包要求里程计信息源在 ROS 上发布位姿的变换信息以及包含速度信息的里程计 nav_msgs/Odometry 消息。

下面具体看一下 nav_msgs/Odometry 消息格式：

```
huanyu@ubuntu:~$ rostopic type /odom
nav_msgs/Odometry
huanyu@ubuntu:~$ rosmsg show nav_msgs/Odometry
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
string child_frame_id
geometry_msgs/PoseWithCovariance pose
  geometry_msgs/Pose pose
    geometry_msgs/Point position
      float64 x
      float64 y
      float64 z
```

```
    geometry_msgs/Quaternion orientation
      float64 x
      float64 y
      float64 z
      float64 w
  float64[36] covariance
geometry_msgs/TwistWithCovariance twist
  geometry_msgs/Twist twist
    geometry_msgs/Vector3 linear
      float64 x
      float64 y
      float64 z
    geometry_msgs/Vector3 angular
      float64 x
      float64 y
      float64 z
  float64[36] covariance
```

上面的消息中主要包含：消息头、父子坐标系名称、odom 轨迹推演的位姿（包含航向四元数）、位姿协方差、线性速度、角速度、速度协方差。在代码中如何计算和发布这些字段的数据呢？下面我们一一查看代码。

```
  std_msgs::Float32 power_msgs;
  power_msgs.data = this->Power_valtage;
  power_pub.publish(power_msgs);


  //next, we'll publish the odometry message over ROS
  nav_msgs::Odometry odom;
  odom.header.stamp = ros::Time::now();;
  odom.header.frame_id = "odom";

  //set the position
  geometry_msgs::Quaternion odom_quat = tf::createQuaternionMsgFromYaw(th);
  odom.pose.pose.position.x = x;
  odom.pose.pose.position.y = y;
  odom.pose.pose.position.z = 0.0;
  odom.pose.pose.orientation = odom_quat;

  //set the velocity
  odom.child_frame_id = this->robot_frame_id;
  odom.twist.twist.linear.x =  this->vx;
  odom.twist.twist.linear.y =  0.0;
  odom.twist.twist.angular.z = this->vth;

  if(this->vx == 0)
  {
      memcpy(&odom.pose.covariance, odom_pose_covariance2, sizeof(odom_pose_covariance2));
      memcpy(&odom.twist.covariance, odom_twist_covariance2, sizeof(odom_twist_covariance2));
  }
  else
  {
      memcpy(&odom.pose.covariance, odom_pose_covariance, sizeof(odom_pose_covariance));
      memcpy(&odom.twist.covariance, odom_twist_covariance, sizeof(odom_twist_covariance));
  }
  odom_pub.publish(odom); //publish the message
```

在上面代码中可以看到位姿协方差和速度协方差有不同的权值矩阵表示，父坐标系是"odom"（机器人初始位置），子坐标系是"base_footprint"（机器人运动学中心）。vx、vy、vth 是机器人分别在 x、y、z 方向上的瞬时速度（右手笛卡尔坐标系）。x、y、odom_quat 是在时间段 Dt 内进行运动学轨迹推演的结果，具体推算过程如下：

```
  /* Calculation tf and odom */
  double delta_x = (vx * cos(th) - vy * sin(th)) * dt;
  double delta_y = (vx * sin(th) + vy * cos(th)) * dt;
  double delta_th = vth * dt;
  x += delta_x;
  y += delta_y;
  th += delta_th;
```

Vx、vy、vth 是由底盘驱动器进行运动学解析后，由串口传送到树莓派的，这些内容在之前的课程中已经做了详细的介绍。经过上面代码段的计算和发布，清晰的明白在机器人上电后，机器人的运动轨迹一直在进行计算推演，最终会以话题的方式发布到 ROS 堆里。但是由于以上速度完全由编码器的时间片累计得到，如果机器人出现打滑和其他意外情况，机器人的位姿推演就出现了误差。该如何解决这种问题，下面

就开始讲解 IMU 姿态传感器，包括姿态融合和话题的发布。

3. IMU 话题

在 HiBot 平台上使用的是 MPU6050，它包含加速度计和陀螺仪，可以时刻得到三轴重力加速度数据和三轴角速度数据。但是由于加速度计的动态性很差，而陀螺仪的短期角速度积分又有着很高的信任度，所以在姿态求解过程中二者相辅相成，就可以得到很好的位姿信息。在 HiBot 启动包中使用 Mahony 算法进行姿态求解。以下是姿态求解的代码片段。

```
// Compute feedback only if accelerometer measurement valid (avoids NaN in accelerometer normalisation)
if(!((ax == 0.0f) && (ay == 0.0f) && (az == 0.0f))) {

    // 首先把加速度计采集到的值(三维向量)转化为单位向量，即向量除以模
    recipNorm = invSqrt(ax * ax + ay * ay + az * az);
    ax *= recipNorm;
    ay *= recipNorm;
    az *= recipNorm;

    // 把四元数换算成方向余弦中的第三行的三个元素
    halfvx = q1 * q3 - q0 * q2;
    halfvy = q0 * q1 + q2 * q3;
    halfvz = q0 * q0 - 0.5f + q3 * q3;

    // Error is sum of cross product between estimated and measured direction of gravity
    halfex = (ay * halfvz - az * halfvy);
    halfey = (az * halfvx - ax * halfvz);
    halfez = (ax * halfvy - ay * halfvx);

    // Compute and apply integral feedback if enabled
    if(twoKi > 0.0f) {
        integralFBx += twoKi * halfex * (1.0f / sampleFreq);    // integral error scaled by Ki
        integralFBy += twoKi * halfey * (1.0f / sampleFreq);
        integralFBz += twoKi * halfez * (1.0f / sampleFreq);
        gx += integralFBx;              // apply integral feedback
        gy += integralFBy;
        gz += integralFBz;
    }
```

详细的推到和求解过程请查看代码学习，姿态求解算法最终得到机器人的姿态四元数

```
  Mpu6050.orientation.w = q0;
  Mpu6050.orientation.x = q1;
  Mpu6050.orientation.y = q2;
  Mpu6050.orientation.z = q3;
```

有了姿态四元数我们便可以发布 IMU 的话题。在发布之前首先需要了解 IMU 话题的消息格式。

```
huanyu@ubuntu:~$ rostopic type /mobile_base/sensors/imu_data
sensor_msgs/Imu
huanyu@ubuntu:~$ rosmsg show sensor_msgs/Imu
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
geometry_msgs/Quaternion orientation
  float64 x
  float64 y
  float64 z
  float64 w
float64[9] orientation_covariance
geometry_msgs/Vector3 angular_velocity
  float64 x
  float64 y
  float64 z
float64[9] angular_velocity_covariance
geometry_msgs/Vector3 linear_acceleration
  float64 x
  float64 y
  float64 z
float64[9] linear_acceleration_covariance
```

在 HiBot 平台中，IMU 话题的名称是/mobile_base/sensors/imu_data，上面输出的消

息格式中：包含消息头、四元数、角速度、加速度和各自数据的协方差。IMU 模块安装在底盘驱动板上，我们在 STM32 代码中已经通过 IIC 协议读取到了加速度和角速度数据，并且通过串口发送到树莓派的工作空间中，然后通过 Mahony 求解了姿态四元数。接下来就是 IMU 话题的发布工作了。

```
void Huanyu_start_object::publisherImuSensor()
{
    sensor_msgs::Imu ImuSensor;

    ImuSensor.header.stamp = ros::Time::now();
    ImuSensor.header.frame_id = "gyro_link";

    ImuSensor.orientation.x = 0.0;
    ImuSensor.orientation.y = 0.0;
    ImuSensor.orientation.z = Mpu6050.orientation.z;
    ImuSensor.orientation.w = Mpu6050.orientation.w;

    ImuSensor.orientation_covariance[0] = 1e6;
    ImuSensor.orientation_covariance[4] = 1e6;
    ImuSensor.orientation_covariance[8] = 1e-6;

    ImuSensor.angular_velocity.x = 0.0;
    ImuSensor.angular_velocity.y = 0.0;
    ImuSensor.angular_velocity.z = Mpu6050.angular_velocity.z;

    ImuSensor.angular_velocity_covariance[0] = 1e6;
    ImuSensor.angular_velocity_covariance[4] = 1e6;
    ImuSensor.angular_velocity_covariance[8] = 1e-6;

    ImuSensor.linear_acceleration.x = 0;
    ImuSensor.linear_acceleration.y = 0;
    ImuSensor.linear_acceleration.z = 0;

    imu_pub.publish(ImuSensor);
}
```

以上代码段发布了 IMU 话题，由于机器人在 2D 平面内移动，不会再 y 和 z 的轴向上产生旋转速度，所以角速度和四元数都赋值为零，同时加速度数据在现阶段不需要，所以也赋值为零。协方差矩阵也赋值为以上定值。

4. 运行和验证

Ubuntu 主机通过 ssh 远程登陆到树莓派，然后运行启动文件

```
huanyu@ubuntu:~$ ssh huike@192.168.12.1
huike@192.168.12.1's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.9.80-v7+ armv7l)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

262 packages can be updated.
2 updates are security updates.

New release '18.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon May 20 20:51:27 2019 from 192.168.12.123
huike@huike-desktop:~$ roslaunch huanyu_robot_start Huanyu_robot_start.launch
```

看到以下输出证明运行成功。

```
[ INFO] [1558369644.592127720]: [ZHOUXUEWEI] Serial Port opened
[ INFO] [1558369644.594050623]: output frame: odom_combined
[ INFO] [1558369644.594570463]: base frame: base_footprint
[ INFO] [1558369646.610354994]: Initializing Odom sensor
[ INFO] [1558369646.611031604]: Initializing Imu sensor
[ INFO] [1558369646.659040890]: Odom sensor activated
[ INFO] [1558369646.659857291]: Imu sensor activated
[ INFO] [1558369646.663041955]: Kalman filter initialized with odom measurement
```

这个时候我们可以在任何终端查看话题和消息数据

```
huanyu@ubuntu:~$ rostopic list
/amcl_pose
/cmd_vel
/mobile_base/sensors/imu_data
/mobile_base/sensors/imu_data_raw
/nodelet_manager/bond
/odom
```

同时也可以查看相应话题下的消息数据

```
huanyu@ubuntu:~$ rostopic echo /odom
```

```
header:
  seq: 4206
  stamp:
    secs: 1558369856
    nsecs: 600601388
  frame_id: "odom"
child_frame_id: "base_footprint"
pose:
  pose:
    position:
      x: 0.0240537448216
      y: 0.000421786698409
      z: 0.0
    orientation:
      x: 0.0
      y: 0.0
      z: -0.0165345906268
      w: 0.999863294312
  covariance: [1e-09, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.001, 1e-09, 0.0, 0.0, 0.0,
 0.0, 0.0, 1000000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1000000.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 1000000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1e-09]
twist:
  twist:
    linear:
      x: 0.0
      y: 0.0
      z: 0.0
    angular:
      x: 0.0
      y: 0.0
      z: 0.0
  covariance: [1e-09, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.001, 1e-09, 0.0, 0.0, 0.0,
 0.0, 0.0, 1000000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1000000.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 1000000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1e-09]
```

5. 总结

本节课程主要学习了 odom 里程计推演的过程、IMU 姿态融合、odom 和 imu_data_raw 话题的发布。同时在机器人上验证了发布数据的过程。在学习过程中，要明白 odom 话题是编码器运算数据的运动学轨迹推到，imu 话题是姿态传感器的姿态融合结果。但是为什么二者提供的都是机器人的姿态数据呢？那该选择那个数据用于导航和建图呢？在上文中也有提到，里程计无法解决车体打滑和其他意外情况。如果能将二者的数据通过扩展卡尔曼融合，会给机器人提供更加精确的位姿数据。所以后续课程中会学习如何将二者的数据进行卡尔曼融合。