第二十一课 基于扩展卡尔曼的传感器数据融合

1. 初步认识 SLAM 机器人

本节课程介绍介绍了 robot_pose_ekf 的概念及相关知识， EKF 包用于评估机器人的 3D 位姿，基于来自不同来源的位姿测量信息。它使用带有 6D（3D position and 3D orientation）模型信息的扩展卡尔曼滤波器来整合来自轮式里程计，IMU 传感器和视觉里程计等传感的数据信息。基本思路就是用松耦合方式融合不同传感器信息实现位姿估计。

Ekf 包的基本算法流程如下：也可以查看树莓派内的 ROS 工作空间中的源码。

以main函数开始：OdomEstimationNode的实例
OdomEstimation的实例
　创建系统模型和测量更新模型：OdomEstimation的构造函数
数据读取：OdomEstimationNode的构造函数
　传感器数据读取与滤波器坐标转换：odomCallback与imucallback
滤波器更新
　滤波器初始化：OdomEstimationNode::spin其①
　滤波器更新与消息发布：OdomEstimationNode::spin其②
结束

1>. OdomEstimationNode 实例化过程中声明了许多变量，比较重要的是它为系统、里程计、imu 等声明了概率密度函数（简写为 pdf）、模型，然后声明了滤波器为扩展卡尔曼滤波器（EKF）。

2>. OdomEstimation 的构造函数声明在 odom_estimation.h 中，公式在 odom_estimation.cpp 中。构造函数为各个传感器以及系统本身创造 pdf 以及模型。系统模型和传感器模型模型分为两种。其中系统模型（在概率机器人中也写做预测模型，以下统一为系统模型）中声明属于 robot_pose_ekf 包自身设定的概率密度函数。

```
    transformer_.lookupTransform("wheelodom", base_footprint_frame_, filter_time, odom_meas_);
    if (odom_initialized_){
  // 将绝对里程测量值转换至水平平面内的相对里程计测量值（坐标转换tf仍然由旋转和平移构成）
  Transform odom_rel_frame =  Transform(tf::createQuaternionFromYaw(filter_estimate_old_vec_(6)),
                      filter_estimate_old_.getOrigin()) * odom_meas_old_.inverse() * odom_meas_;
  ColumnVector odom_rel(6);
  // 将转换分解到x,y,z,Rx,Ry,Rz
  decomposeTransform(odom_rel_frame, odom_rel(1), odom_rel(2), odom_rel(3), odom_rel(4), odom_rel(5
  // 角度溢出的话就进行修正
  angleOverflowCorrect(odom_rel(6), filter_estimate_old_vec_(6));
  // 更新滤波器
  // 更新协方差矩阵
  odom_meas_pdf_->AdditiveNoiseSigmaSet(odom_covariance_ * pow(dt,2));

      ROS_DEBUG("Update filter with odom measurement %f %f %f %f %f %f",
                odom_rel(1), odom_rel(2), odom_rel(3), odom_rel(4), odom_rel(5), odom_rel(6));
      // 给里程计系统模型更新这次的测量值
  filter_->Update(odom_meas_model_, odom_rel);
```

2. HiBot 上的 EKF 配置

在 HiBot 上，我们用 EKF 包来融合编码器（odom）的测量和 IMU（imu_data）的测量，得到更加精确的机器人定位数据。在启动 launch 文件中基本的配置如下：

```
<!-- Robot pose ekf -->
<node pkg="robot_pose_ekf" type="robot_pose_ekf" name="robot_pose_ekf" output="screen">
  <param name="freq" value="30.0"/>
  <param name="sensor_timeout" value="1.0"/>
  <param name="odom_used" value="true"/>
  <param name="imu_used" value="true"/>
  <param name="vo_used" value="false"/>
  <remap from="/odom" to="/odom" />
  <remap from="/imu_data" to="/mobile_base/sensors/imu_data" />
</node>
```

其中主要配置了发布频率，传感器数据的延时容忍度，具体要使用那些传感器的数据来融合。还有使用传感器发布的话题名称，用于监听。同时 ekf 会发布融合之后的机器人位姿数据，话题名称如下代码片段定义/robot_pose_ekf/odom_combined:

```
ROS_INFO("Subscribing to robot_pose_ekf/odom_combined");
ekf_sub_ = node_.subscribe("/robot_pose_ekf/odom_combined", 10, &TestEKF::EKFCallback, (TestEKF*)this);
```

3. HiBot 上测试 EKF

1>. 将机器人放到上节课程新建地图的起点，打开机器人电源开关，等待大概 1 分钟后。Ubuntu 主机连接树莓派开放的 wifi。然后打开新的终端 ssh 连接到树莓派，运行 HiBot 启动节点：

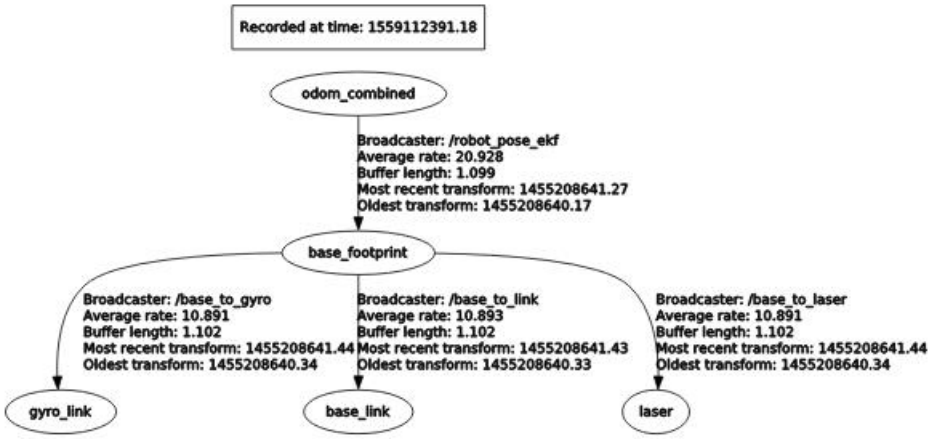
```
huanyu@ubuntu:~$ ssh huike@192.168.12.1
huike@huike-desktop:~$ roslaunch huanyu_robot_start Huanyu_robot_start.launch
[ INFO] [1455208732.734492355]: [ZHOUXUEWEI] Serial Port opened
[ INFO] [1455208732.745544095]: output frame: odom_combined
[ INFO] [1455208732.746011741]: base frame: base_footprint
[ INFO] [1455208734.751010952]: Initializing Odom sensor
[ INFO] [1455208734.751690729]: Initializing Imu sensor
[ INFO] [1455208734.800748993]: Odom sensor activated
[ INFO] [1455208734.801159557]: Imu sensor activated
[ INFO] [1455208734.820802883]: Kalman filter initialized with odom measurement
```

在上述终端看到如下输出，证明卡尔曼融合包已经启动运行。

```
Odom sensor activated
Imu sensor activated
Kalman filter initialized with odom measurement
```

2>. 下面我们来查看融合后的位姿数据和相关 tf 转换。

```
^Chuanyu@ubuntu:~$ rostopic echo /robot_pose_ekf/odom_combined
header:
  seq: 1615
  stamp:
    secs: 1455208595
    nsecs: 938891281
  frame_id: "odom_combined"
pose:
  pose:
    position:
      x: -0.0275320341067
      y: -0.00119769801801
      z: 0.0
    orientation:
      x: 0.0
      y: 0.0
      z: -0.0400005255409
      w: 0.999199658705
huanyu@ubuntu:~$ rosrun rqt_tf_tree rqt_tf_tree
```



从上面的 tf_tree 可以看出 odom_conbined->base_footprint 之间的 tf 转换是由 /robot_pose_ekf 提供的，也就是说机器人从上电时刻开始推演的轨迹是由 EKF 融合后提供的。