雷达项目算法文档

运动点聚类

身高计算和姿态识别

聚类之后的结果会传到MSS端，MSS端的task\_app.c文件中的MmwDemo\_appTask函数负责将数据读入并写入对应结构体传给前端，程序在这里进行身高的计算。

对于一个对象，传入的数据主要有以下属性：

typedef struct MmwDemo\_output\_message\_target\_t

{

/\*! @brief tracking ID \*/

uint32\_t tid;

/\*! @brief Detected target X coordinate, in m \*/

float posX;

/\*! @brief Detected target Y coordinate, in m \*/

float posY;

float posZ;

/\*! @brief Detected target X velocity, in m/s \*/

float velX;

/\*! @brief Detected target Y velocity, in m/s \*/

float velY;

float velZ;

/\*! @brief Detected target X acceleration, in m/s2 \*/

float accX;

/\*! @brief Detected target Y acceleration, in m/s2 \*/

float accY;

float accZ;

} MmwDemo\_output\_message\_target;

在这一步计算之后添加了一个身高字段和一个姿态字段，因为雷达板采用的是水平放置，所以posz数据加上雷达板的高度就可以认为是检测对象的高度。

因为环境干扰和雷达板检测运动点的原因，posz的数据有一定的误差，为了不让误差数据对结果造成太大的干扰，程序中设计了身高上限阈值和下限阈值，下限阈值用来避免出现0.9,1.2之类的“不正常”身高，上限阈值同样用来避免2.2,2.5之类的“不正常”身高（实际测试中发现这样的误差还是挺常见的）。

同样，如果一个对象在连续200帧以内都出现了，则可以“固定”该对象的身高，“固定”方法有多种，可以取平均或者取加权平均（越靠近帧底的权重越大），不过这些都额外开辟内存，因此暂时将最后的身高作为“固定”身高。

在身高“固定”之后，就可以开始计算姿态，姿态以实时身高为准，依据实时身高和固定身高的比例来计算姿态，如果比例大于0.85，则认为对象是坐着的，比例大于0.35且小于0.85则是坐着的，比例大于0.2则是躺着的，其余则是Unknown。

算法的前提假设如下：

1. 在室内的情况下，地面将是平坦的，即高度没有变化
2. 根据用户的需要，传感器可以安装在不同的高度
3. 在大多数情况下，传感器安装时没有向下倾斜
4. 对象不能只是出现，他们必须进入现场。当他们进入场景时，他们将走路或跑步（也就是处于运动的状态）。

算法的流程图如下：

