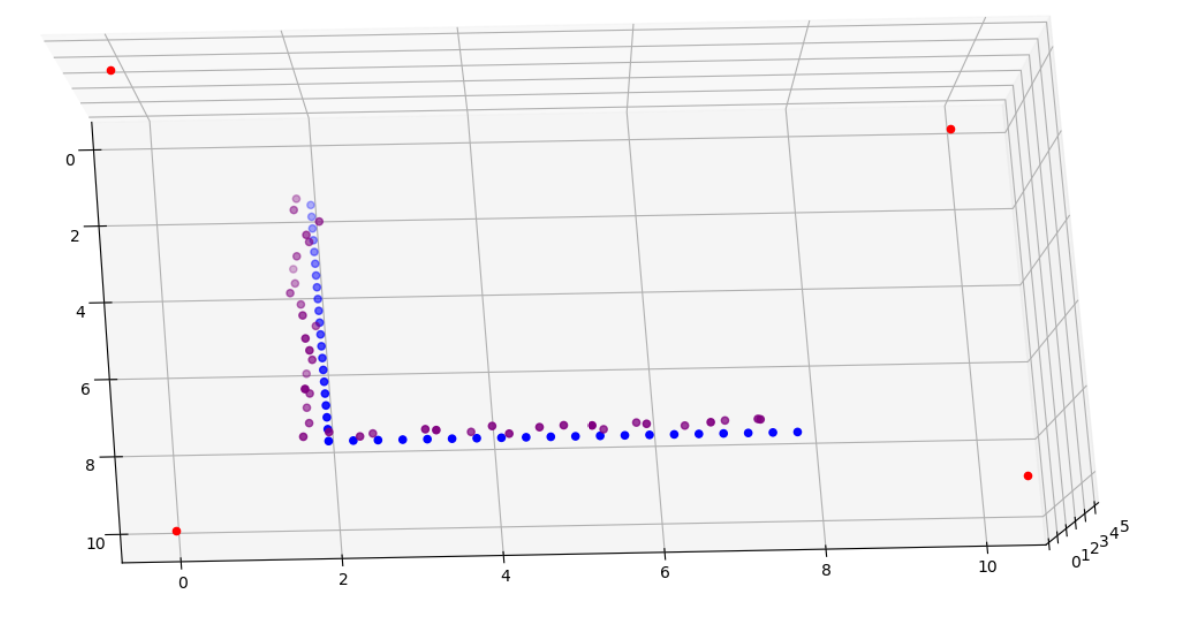
# Savitzky-Golay滤波

## 一、原理及Python仿真说明

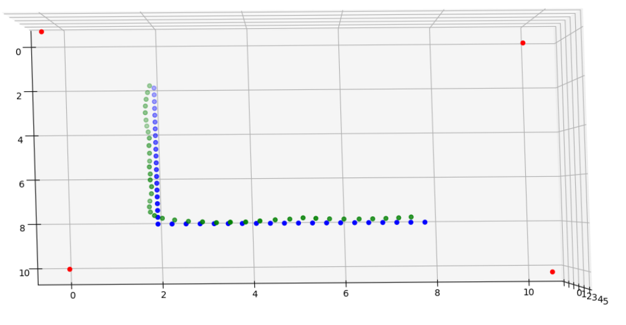
使用**Savitzky-Golay**算法对标签x y z坐标进行**实时的**位置滤波。基于标签的历史数据对当前运动状态进行预测，与UWB测量值融合，实现跟踪轨迹的平滑与抗噪。Savitzky-Golay原理见文件夹中论文。

**蓝色点**为标签的真实轨迹。**紫色点**为含测量噪声的LM算法定位结果。**绿色点**为对测量结果滤波之后的结果。滤波窗口为9， 阶次为2.。该参数可调，且必须调。这代表我们用匀加速运动描述9个采样时间内物体的运动状态。这应用了微分的思想来描述物体的运动。

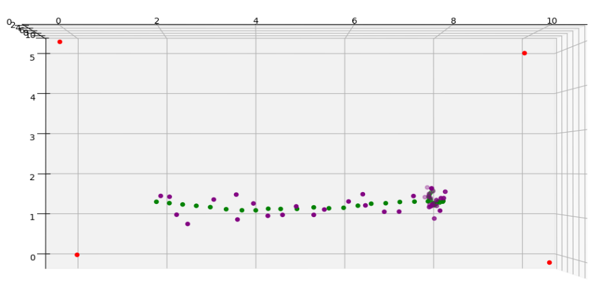
注意，在实际使用当中，若定位频率过低，相对较大的滤波窗口将无法较好地反映运动状态（需要理解微分思想）。此时滤波器将失去滤波的意义（即无法对噪声进行有效滤除），只是单纯地将测量点平滑而已。针对行人目标，我推荐标签的定位频率在20Hz以上。



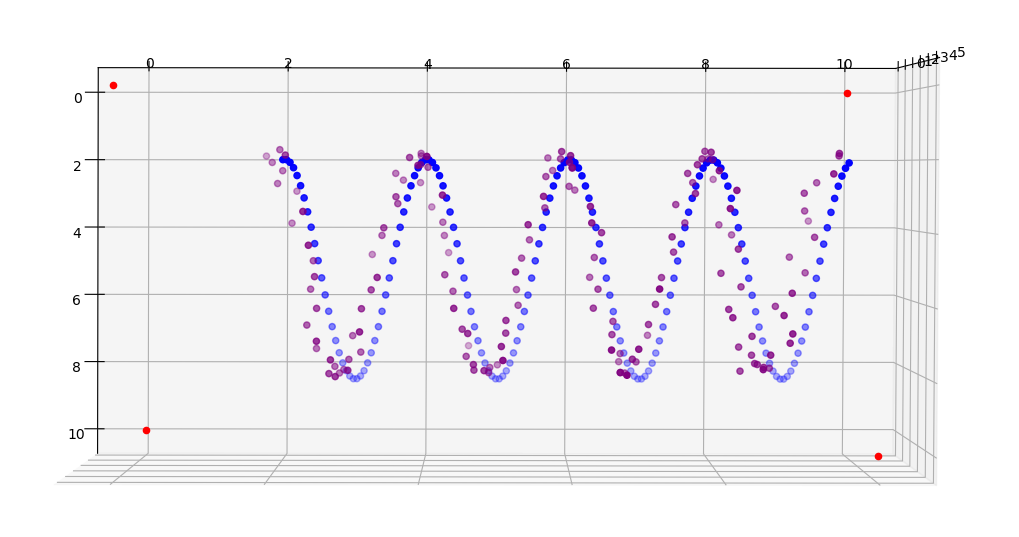
直角轨迹下测量值与真值对比（俯视图）



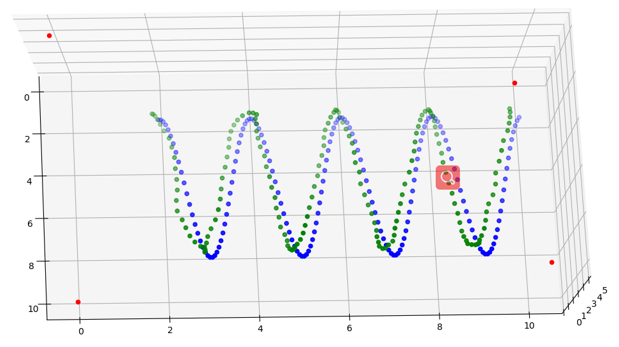
直角轨迹下滤波值与真值对比（俯视图）



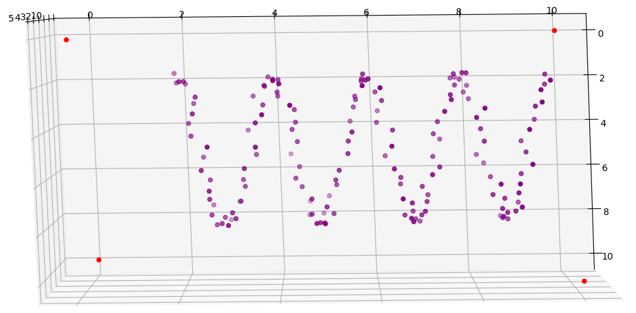
直角轨迹下滤波值与测量值对比（侧视图）



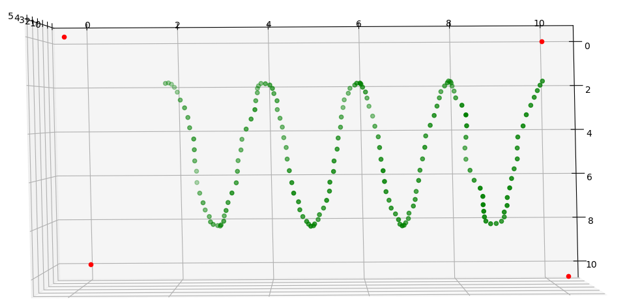
曲线轨迹下测量值与测量值对比（俯视图）



曲线轨迹下滤波值与测量值对比（俯视图）



曲线轨迹下测量值（俯视图）



曲线轨迹下滤波值（俯视图）

## 二、C++程序接口及参数说明

使用三个函数分别对XYZ三坐标信号进行滤波。这三个函数应放在Tag对象下。函数输入为实时的新信号与滤波的阶次，输出为滤波后的结果。

滤波器参数(m, t, n, d)说明：

m是滤波器窗口从中心到边缘的宽度。即滤波窗口整宽为2m+1。更大的滤波窗口会带来更平滑的滤波效果，但是会加大滤波的延迟和惯性。这个参数需要根据实际应用进行调整。

t是决定滤波结果赋值的参数。t=0时，滤波结果赋值给窗口中心。这意味着滤波会有m个采样间隔的延迟，但滤波效果会比较好。当t=m时，滤波器是实时的，几乎没有延迟，但代价是平滑的效果会稍弱。这个也是可以根据实际应用进行取舍。

n是多项式拟合时使用的最高阶次。n越大意味着滤波器对数据更加敏感，也意味着我们用多高阶次的运动方程来描述目标在一个滤波窗口内的运动。针对行人目标，我建议该参数在1-2之间调整。XY方向可以使用2阶，Z方向使用1阶。

d是微分阶次。建议保持为0。

## 三、注意事项（重要）

Savitzky-Golay滤波假定采样间隔是相同的。由于目前TDOA模式下的标签会有休眠模式，静态下定位频率很小，导致“由动转静”的过程中采样间隔变化较大，会使滤波器生成不好的结果。所以需要引入一个机制：

每次开始计算定位都要记录下时戳，用来衡量两次定位采样的间隔。在下一次定位中，如果过了1.3倍的（上一次）采样间隔时间，仍然没有定位信号到达，则认为是标签“由动转静”了，需要将上一次的定位结果作为此次定位结果喂给滤波器。同样需要记下时戳。

这种喂虚拟定位结果给滤波器的操作最多执行2m+1次。2m+1为滤波窗口总宽。