卡尔曼(Kalman)滤波算法原理、C语言实现及实际应用

全栈程序员栈长 • 2022年6月13日 上午6:16 • 未分类

卡尔曼(Kalman)滤波算法原理、C语言实现及实际应用文章目录卡尔曼滤波一、滤波效果展示二、简介三、组成1.预测状态方程（1）目的：（2）方程：（3）备注2.预测协方差方程（1）目的（2）方程（3）备注3.卡尔曼增益方程（1）目的（2）方程（3）备注4.跟新最优值方程（卡尔曼滤波的输出）（1）目的（2）方程（3）备注5.更新协方差方程（1）目的（2）方程（3）备注四、C程序代码实现1.参数列表2.代码实现（一维数据滤波）五、发送波形到...

大家好，又见面了，我是你们的朋友全栈君。

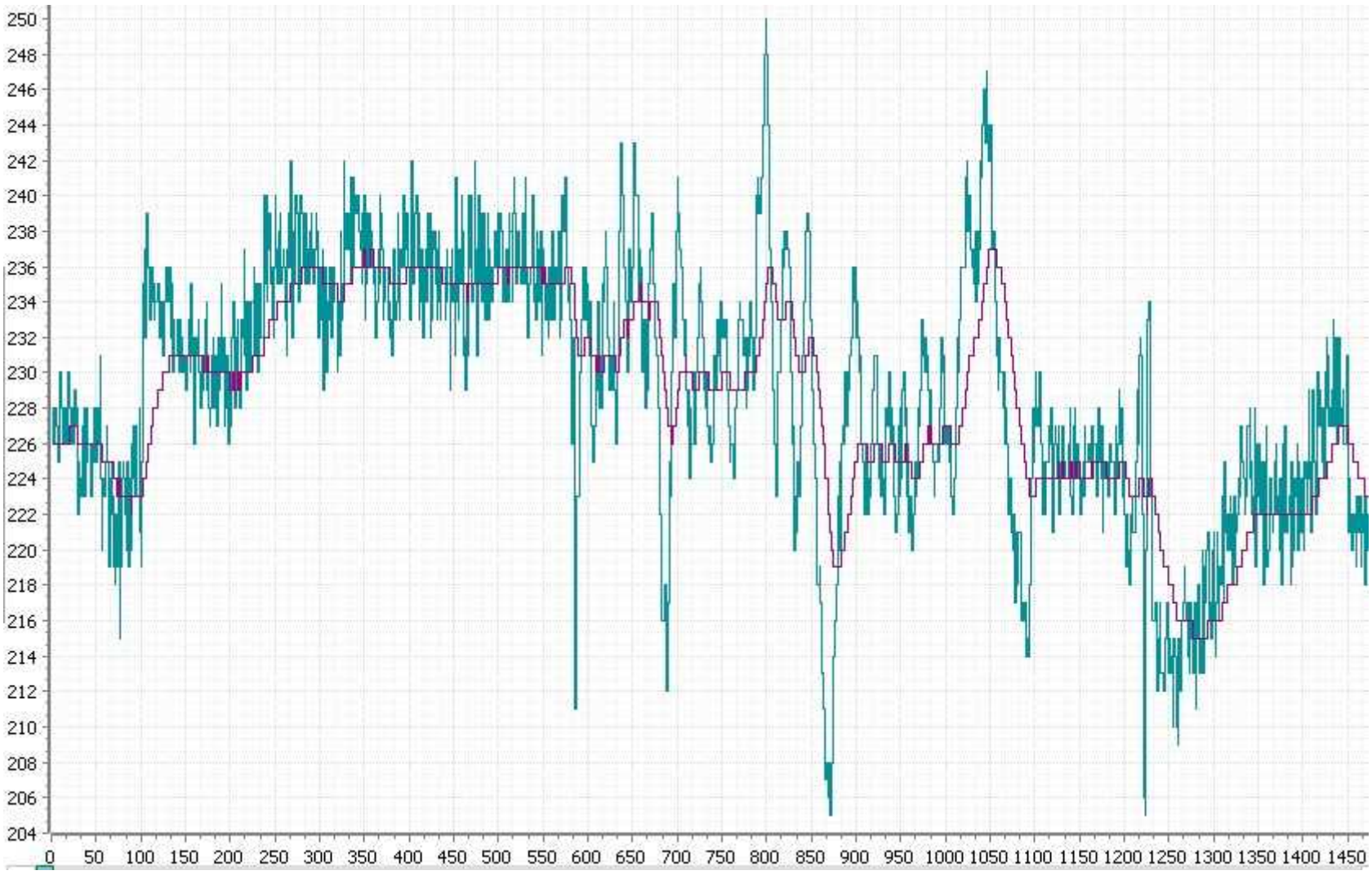
文章目录

卡尔曼滤波

一、滤波效果展示

蓝色的波形是实际测得的数据，红色的波形是经 Kalman 滤波后的数据波形。

注：这里是实际应用激光测距传感器（TOF）vl53l0x 测得的距离数据。



二、简介

采用递归的方法解决线性滤波问题，只需要当前的测量值和前一个采样周期的估计值就能进行状态估计，需要的存储空间小，每一步的计算量小。

三、组成

1. 预测状态方程

（1）目的：

由 **系统状态变量k-1时刻的最优值** 和 **系统输入** 计算出k时刻的 **系统预测值。**

（2）方程：

$$X_{k|k-1} = F_k X_{k-1|k-1} + B_k u_k$$

- $X_{k|k-1}$是利用$k-1$时刻预测的当前状态结果
- $X_{k-1|k-1}$是$k-1$时刻最优值
- F_k是作用在$X_{k-1|k-1}$状态下的变换矩阵，它是算法对状态变量进行预测的依据
- B_k是作用在控制量上的变换矩阵，在大多数实际情况下并没有控制增益
- u_k是当前状态的控制增益，一般没有这个变量，可以设为0

（3）备注

①. X k-1|k-1 为k-1时刻的输出。

②. 当X为一维数据时，Fk的值是1。

③. **一维数据下（uk=0时）： 系统预测值 = 系统状态变量k-1时刻的最优值。**

2. 预测协方差方程

（1）目的

根据 **k-1时刻的系统协方差** 预测 **k时刻系统协方差。**

（2）方程

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k$$

- $P_{k|k-1}$是k时刻系统协方差矩阵；
- $P_{k-1|k-1}$是$k-1$时刻系统协方差矩阵；
- Q_k是系统过程噪声的协方差。协方差矩阵只要确定了一开始的P_0，后面都可以递推出来，而且初始协方差矩阵P_0只要不是为0，它的取值对滤波效果影响很小，都能很快收敛。

（3）备注

①. 当X为一维数据时，Fk的值是1。

3. 卡尔曼增益方程

（1）目的

根据（k时刻） **协方差矩阵的预测值** 计算 **卡尔曼增益。**

（2）方程

$$K_k = \frac{P_{k|k-1} H_k^T}{H_k P_{k|k-1} H_k^T + R_k}$$

- K_k是卡尔曼增益，是滤波的中间结果；
- H_k是对象的预测矩阵；
- R_k是对象测量噪声的协方差矩阵，它是一个数值，作为已知条件输入滤波器。注意，这个值过大过小都会使滤波效果变差，且R_k取值越小收敛越快，所以可以通过实验手段寻找合适的R_k值再利用它进行真实的滤波。

（3）备注

①. 当 Pk|k-1 为一个一维矩阵时，Hk 是1。

4. 跟新最优值方程（卡尔曼滤波的输出）

（1）目的

根据 **状态变量的预测值** 和 **系统测量值** 计算出 **k时刻状态变量的最优值。**

（2）方程

$$X_{k|k} = X_{k|k-1} + K_k(Z_k - H_k X_{k|k-1})$$

- $X_{k|k}$是k时刻状态变量最优估计值；
- Z_k是对象的测量值。

（3）备注

①. 当 Pk|k-1 为一个一维矩阵时，Hk 是1。

5. 更新协方差方程

（1）目的

为了求 **k时刻的协方差矩阵。**（为得到k+1时刻的卡尔曼输出值做准备）

（2）方程

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}$$

- $P_{k|k}$为k时刻协方差矩阵；
- I为单位矩阵。

作用：用于下一次卡尔曼滤波

（3）备注

①. 当 Pk|k-1 为一个一维矩阵时，Hk 是1。

四、C 程序代码实现

1. 参数列表

序号	类型	名称	描述
1	float	LastP	上次估算协方差
2	float	Now_P	当前估算协方差
3	float	out	卡尔曼滤波器输出值（最优值）
4	float	Kg	卡尔曼增益
5	float	Q	过程噪声协方差
6	float	R	观测噪声协方差

2. 代码实现（一维数据滤波）

实际参数是参照别人已经选好的参数，不过也可以自己改变参数，去观察波形的效果，体会每个参数对于滤波效果的影响，这里不详细介绍。

```
 //1. 结构体类型定义
typedef struct
{

float LastP;//上次估算协方差 初始化值为0.02
float Now_P;//当前估算协方差 初始化值为0
float out;//卡尔曼滤波器输出 初始化值为0
float Kg;//卡尔曼增益 初始化值为0
float Q;//过程噪声协方差 初始化值为0.001
float R;//观测噪声协方差 初始化值为0.543
}KFP；//Kalman Filter parameter
//2. 以高度为例 定义卡尔曼结构体并初始化参数
KFP KFP_height={
0.02,0,0,0,0.001,0.543};
/** *卡尔曼滤波器 *@param KFP *kfp 卡尔曼结构体参数 * float input 需要滤波的参数的测量值（即传感器的采集值） *@return 滤波后的参数（最优值） */
float kalmanFilter(KFP *kfp,float input)
{

//预测协方差方程：k时刻系统估算协方差 = k-1时刻的系统协方差 + 过程噪声协方差
kfp->Now_P = kfp->LastP + kfp->Q;
//卡尔曼增益方程：卡尔曼增益 = k时刻系统估算协方差 / （k时刻系统估算协方差 + 观测噪声协方差）
kfp->Kg = kfp->Now_P / (kfp->NOw_P + kfp->R);
//更新最优值方程：k时刻状态变量的最优值 = 状态变量的预测值 + 卡尔曼增益 * （测量值 - 状态变量的预测值）
kfp->out = kfp->out + kfp->Kg * (input -kfp->out);//因为这一次的预测值就是上一次的输出值
//更新协方差方程: 本次的系统协方差付给 kfp->LastP 威下一次运算准备。
kfp->LastP = (1-kfp->Kg) * kfp->Now_P;
return kfp->out；
}
/** *调用卡尔曼滤波器 实践 */
int height;
int kalman_height=0;
kalman_height = kalmanFilter(&KFP_height,(float)height);
```

五、发送波形到上位机显示

这里使用的是匿名的上位机 V65 版本，具体如何使用可以参考茶大的博客，并且茶大博客里面有上位机的下载地址。茶大博客地址：https://blog.csdn.net/wangjt1988/article/details/83684188

注：文章方程截图及参数来源于中科浩电。

Author : Beyonderwei
Email : Beyonderwei@163.com
Website : http://beyonderwei.com
WeChat：