1.1 粒子滤波算法

1.1.1算法程序的简要说明

该算法程序是粒子滤波算法的一个应用，算法通过寻找一组在状态空间传播的随机样本来近似的表示概率密度函数，以样本均值替代积分运算，进而获得系统状态最小方差估计。

该算法对状态转移方程为

***其中，R 为方差为1，期望为0的满足正态分布的随机变量，即为带干扰的高斯白噪声；为在n时刻下的该系统的当前状态。***

观测方程为

的系统进行估计，并输出估计结果。

1.1.2 算法原理

该算法通过寻找一组在状态空间中传播的随机样本来近似的表示概率密度函数，以样本均值替代积分运算，进而获得系统状态最小方差估计的过程。

粒子滤波器能够从一系列含有杂讯或不完整的观测值中，估计出动态系统的内部状态。 在动态系统的分析中，需要两个模型，一个用来描述状态随时间的变化(系统模型) ，另一个用来描述每个状态下观测到的杂讯(观测模型) ，将这两个模型都用机率来表示。在许多情况下，每得到一个新的观测值时，都必须对系统做出一次估计，利用递回滤波器 ，能够有效地达到这样的目的。 递回滤波器会对得到的资料做连续处理，而非分批处理，因此不需要将完整的资料储存起来，也不需要在得到新的观测值时，将现有的资料重新做处理。 递回滤波器包含两个步骤：

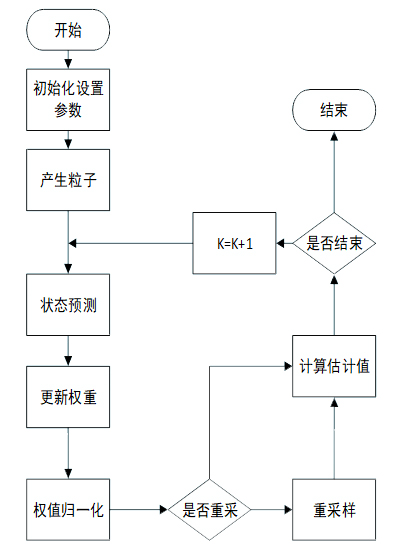
预测 ：利用系统模型，由前一个状态的资讯预测下一个状态的机率密度函数 。

更新 ：利用最新的观测值，修改预测出的机率密度函数。

藉由贝叶斯推论 (Baysian inference)，可以描述出状态空间的机率，并在得到新的观测值时，对系统做出更新，因而达成上述目的。

1.1.2 算法流程及说明:

粒子滤波算法流程如图1-1-1所示：



***图1-1-1 PID粒子滤波算法流程图***

该算法首先初始化参数并在初态周围随机产生模拟状态预测粒子，并参照状态观测方程更新归一化权重，此后重新采样进行下一个时间阶段的粒子滤波预测直到结束粒子滤波预测为止。

1.1.3输入数据

输入数据由状态转移方程为：

的系统产生，其中初始变量设置为10。

输入数据作为待预测系统的真实状态保存在xAll[]数组中，并输出到控制台

运行程序，输入数据保存至ParticleFilter.c源文件同目录的InitialData.dat文件中。

2.1.4输出数据

粒子滤波器的预测状态保存在xMeanParticle[]数组中，输出到控制台

运行程序，输出数据保存至ParticleFilter.c源文件同目录的FilterData.dat文件中。

1.2 卡尔曼滤波算法

1.2.1算法程序的简要说明

该算法程序是卡尔曼滤波算法的一个应用，以最小均方误差为最佳估计准则，采用信号与噪声的状态空间模型，利用前一时刻的估计值和当前时刻的观测值来更新对状态变量的估计，求出当前时刻的估计值。

该算法对状态转移方程为

***其中，R 为方差为1，期望为0的满足正态分布的随机变量，即为带干扰的高斯白噪声；为在n时刻下的该系统的当前状态。***

观测方程为

的系统进行估计，并输出估计结果。

1.2.2 算法原理

卡尔曼滤波器可分为两个部分：时间更新方程和测量更新方程。时间更新方程负责及时向前推算当前状态变量和误差协方差估计的值。测量更新方程负责反馈——它将先验估计和新的测量变量结合以构造改进的后验估计。

卡尔曼滤波的一个典型实例是从一组有限的，包含噪声的物体位置的观察序列（可能有偏差）预测出物体的位置的坐标及速度。

卡尔曼滤波的核心思想是递推求解, 在忽略之前数据量的前提下, 根据xk-1递推出xk。经典卡尔曼滤波算法步骤与结构如下:

首先设置初始量

再通过状态预测并推算误差协方差：

计算卡尔曼增益

计算误差

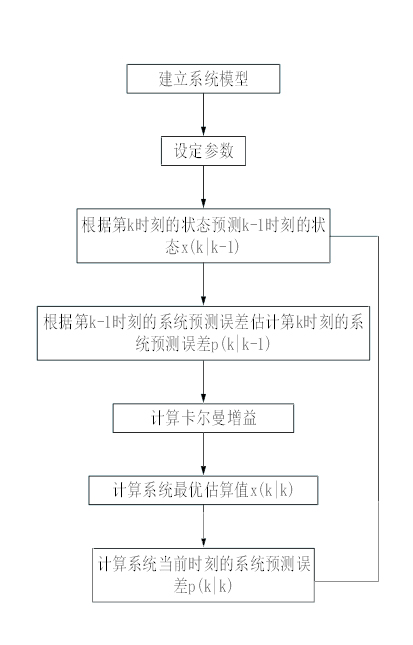
更新误差协方差

该算法程序采用的为以经典卡尔曼滤波为基础的扩展卡尔曼滤波算法，其仅将卡尔曼滤波里的矩阵A和H用公式函数f与h代替, 如下公式所示。

然后采用泰勒展开式的一阶逼近非线性系统，如公式下所示, 对其状态进行线性估计, 并通过更新系统协方差矩阵对系统状态进行递归更新。

1.2.3 算法流程及说明:

卡尔曼滤波器算法流程如图1-2-1所示：



***图1-2-1 卡尔曼算法流程图***

该算法首先在选定的预测系统的基础上初始化设定参数，此后对该系统加入高斯噪声进行系统模拟，并计算卡尔曼增益，此后更新预测状态。在新的时间阶段再重复进行状态预测的过程，直到结束。

1.2.4输入数据

输入数据由状态转移方程为：

的系统产生，其中初始变量设置为10。

输入数据作为待预测系统的真实状态保存在xAll[]数组中，并输出到控制台

运行程序，输入数据保存至KalmanFilter.c源文件同目录的InitialData.dat文件中。

1.2.5输出数据

卡尔曼滤波器的预测状态保存在xharAll[]数组中，输出到控制台

运行程序，输出数据保存至KalmanFilter.c源文件同目录的KalmanFilterFilterData.dat文件中。

1.5 视觉制导算法

1.5.1算法程序的简要说明

该算法程序是视觉制导算法的一个应用。该算法程序对所获取的图片执行制导功能，灰度化读取图片并将图片进行二值化并拟合出测试图片的道路中心线，最终获取偏移角度的制导信息。

该算法将偏移角度制导信息输出到控制台，并将每一行中心像素偏移角度保存至并Angle.dat文件中。

文件marginData.dat.包含了左侧边缘像素信息，右侧边缘像素信息，中心线像素信息；

使用该代码前，使用者应当在代码的第9行将：

Mat process\_img=imread("C:/Users/admin/Desktop/Test1.jpg", IMREAD\_GRAYSCALE);

中的.jpg中的图片文件修改成所需要进行制导图片Test1具体的路径地址.再运行visualNavigation.c++文件即可。

xx/Test1Angle.dat为Test1图片制导生成的Angle.dat文件供使用者参考

xx/Test1图片数据.dat为Test1图片制导生成的marginData.dat文件供使用者参考

xx/Test1制导轨迹.jpg为Test1图片制导生成中心线的图片，供使用者参考

xx/Test1制作导结果.jpg为Test1图片制导控制台的结果输出，包含了制导信息。

1.5.2算法原理

该算法是基于opencv对图片进行图像处理并拟合图像的中心线。该算法首先读取处理图片所有像素的灰度值，此后对整张图片的灰度求平均值。依据平均值作为划分区间，对图片进行二值化处理，高于平均值的像素灰度设置为255，低于平均值的像素灰度设置为0。此后对图片逐行查找边界，即为灰度值由0到255或灰度值由255到0的点作为左右边界，取中值即为制导图片的道路中心线。其次划分为两种制导模式，分别为以获得图片中央作为小车行驶的中心，以及道路中心线作为小车行驶的中心。在从下往上数50个像素处计算与中心纵向偏离的像素个数，依据像素个数判断小车偏离的角度，给出下一个时刻小车是否需要转向，向哪个方向转向以及偏转的角度。

1.5.3 算法流程及说明:

视觉制导算法流程如图1-5-1所示：

输入制导图片，图片高

计算整张图片灰度平均值

依据像素灰度值的大小进行二值化

查找两侧边界

两侧边界求均值得到中心线

获得偏离中心的像素

给出制导方向，偏移角度

结束

***图1-5-1 视觉制导算法流程图***

首先输入要进行制导的图片以及图片的高度，运行程序获得图片中道路的中心线。在从下往上数50个像素处计算与中心纵向偏离的像素个数，依据像素个数判断小车偏离的角度，给出下一个时刻小车是否需要转向，向哪个方向转向以及偏转的角度。

1.5.4输入数据

处理图片 Test1

1.5.5输出数据

制导图片道路的左侧边缘，保存在leftMargin的数组变量中

制导图片道路的右侧边缘，保存在rightMargin的数组变量中

制导图片道路的中心路径，保存在midPoints的数组变量中

最终将上述数据保存至marginData.dat在源文件同目录的文件当中。  
 以中心线起点为小车位置的偏离角度保存在变量deviAngle1中，输出到控制台。

以图片中心为小车起点的偏离角度保存在变量中deviAngle 中，输出到控制台。

以图片中心为起点，每一个高度的中心线偏离中心的角度保存在Angle.dat，在源文件同目录的文件当中。

1.7 PID控制算法

1.7.1算法程序的简要说明

该程序为pid算法的一个应用。程序对于所给定的不完全随机信号，锯齿波信号以及三角波信号进行了跟踪控制。在给定的目标信号的情况下，生成控制跟踪信号在任意时刻尽可能的向给定的目标信号逼近。

对于完全随机信号，每一个时刻生成的信号值为随机数且与此前的信号值无关联的信号，是无法实现跟踪控制的。因此在该算法中所生成的随机信号做了一个限制，即每一次生成随机信号时，与上一次信号的取值做做对比，如果相差大于一个给定的阈值则重新生成信号，直到与上一次生成的随机信号的差值小于阈值为止。这样的做法等价于后一个时间段的信号值是前一个时间段的信号值加上一个幅度微小的随机数。鉴于生成信号值时此前的生成的信号值已经成为了确定性事件，因此所生成的信号不是一个完全随机的信号，相邻的信号值之间存在着关联，故可以对其进行模拟控制。

为使用该程序，使用者应当将pidcontrol.c的源文件与pidcontrol.h的头文件放在同一目录当中，修改pidcontrol.h 的头文件中signalSource的值为1或2或3:

设置为1 则为三角波信号，设置为2为锯齿波信号，设置为3为不完全随机信号。直接运行程序即可。输入数据展示在控制台以及targetData.dat文件当中，输出结果分别在控制台以及pidcontrolData.dat文件当中，。

1.7.2 算法原理

PID中的每一个字母分别代表了比例(proportion)，积分(integration)和微分(differentiation),它在比例，积分和微分的三个方面上对所需求的变量进行稳定的控制，从而达到理想的目标状态，其控制公式如下所示：

***PID计算公式(连续情况)***

***PID计算公式（离散情况）***

在连续情况的PID中，变量err(t)代表控制变量的目标状态与当前状态在时间t的差值，kp为比例调节系数，积分调节，微分调节的控制系数。Ti为积分调节的积分时间，Td为微分调节的微分时间差(理想情况为无穷小)。U(t)为当前时间t下，对控制变量的增量。

将连续情况的PID进行推导即可得到离散情况的PID计算公式，其中err(t)与e(k)相对应，程序采用的是离散情况的PID进行控制计算。

比例控制:

比例控制取决于控制变量的目标状态与当前状态在时间t的差值err(t)，通过对控制变量添加kp\*err(t)的增量Up(t),可使控制变量状态向目标状态逼近。若是单独采用比例控制的方法，会产生余差的影响，若要克服余差则必须引用积分控制。

积分控制：

通过积分可将一段时间内的误差相加，即便是一个很小的误差，也能让积分响应缓慢增加。积分响应的目的在于将稳定状态的误差保持在0.稳定状态误差是控制变量在稳定时和目标状态的差值，当满足稳定条件时，会产生积分饱和的最终结果，可以使系统在进入稳态后无稳态误差

微分控制：

在微分控制中，控制器的输出与输入误差信号的微分（即误差的变化率）成正比关系。自动控制系统在克服误差的调节过程中可能会出现振荡甚至失稳。其原因是由于存在有较大惯性组件（环节）或有滞后（delay）组件，具有抑制误差的作用，其变化总是落后于误差的变化。解决的办法是使抑制误差的作用的变化“超前”，即在误差接近零时，抑制误差的作用就应该是零。这就是说，在控制器中仅引入“比例”项往往是不够的，比例项的作用仅是放大误差的幅值，而需要增加的是“微分项”，它能预测误差变化的趋势，这样，具有比例+微分的控制器，就能够提前使抑制误差的控制作用等于零，甚至为负值，从而避免了被控量的严重超调。所以对有较大惯性或滞后的被控对象，比例+微分（PD）控制器能改善系统在调节过程中的动态特性。

1.7.3 算法流程及说明:

PID控制算法流程如图1-7-1所示：

开始

输入目标信号并采样得到r(k)

计算目标信号r(k)与跟踪y(k)控制信号的偏差

计算控制增量U(k),

跟踪信号y(k-1)添加上U(k)作为y(k)的值

否

是否模拟完所有点

是

结束

***图1-7-1 PID控制算法流程图***

首先设定控制变量的初始状态，目标状态等一系列的参数，接着进入PID控制系统。当初始状态经过加上一系列的控制量之后与目标状态的差的绝对值小于一个稳定的阈值之后停止进行PID控制循环，此时控制的变量达到了目标状态。

1.7.4输入数据

程序用于进行跟踪控制的目标信号：

1. 锯齿波信号
2. 三角波信号
3. 随机信号

以下参数置于xx/xx/pidcontrol.h中，执行程序时需根据需求修改参数

采样时间ts：

为生成对应的目标信号，执行A/D模拟转换到离散数据的采样时间间隔，默认为0.01s

模拟跟踪时间simuLength：

为生成对应的目标信号，执行A/D模拟转换到离散数据的采样点，默认为3000个

信号选择参数 signalSource:

用于选择控制以上信号的具体哪一个信号，设置为1 则为三角波信号，设置为2为锯齿波信号，设置为3为随机信号。

输入信号离散化采样数据保存在xxxx/xxxx/targetData.dat的文件中，同时显示在控制台左侧

1.7.5输出数据

对于目标信号的控制跟踪信号，保存在xxx/xxx/pidcontrolData.dat的文件中，同时显示在控制台右侧，与目标信号做比较。