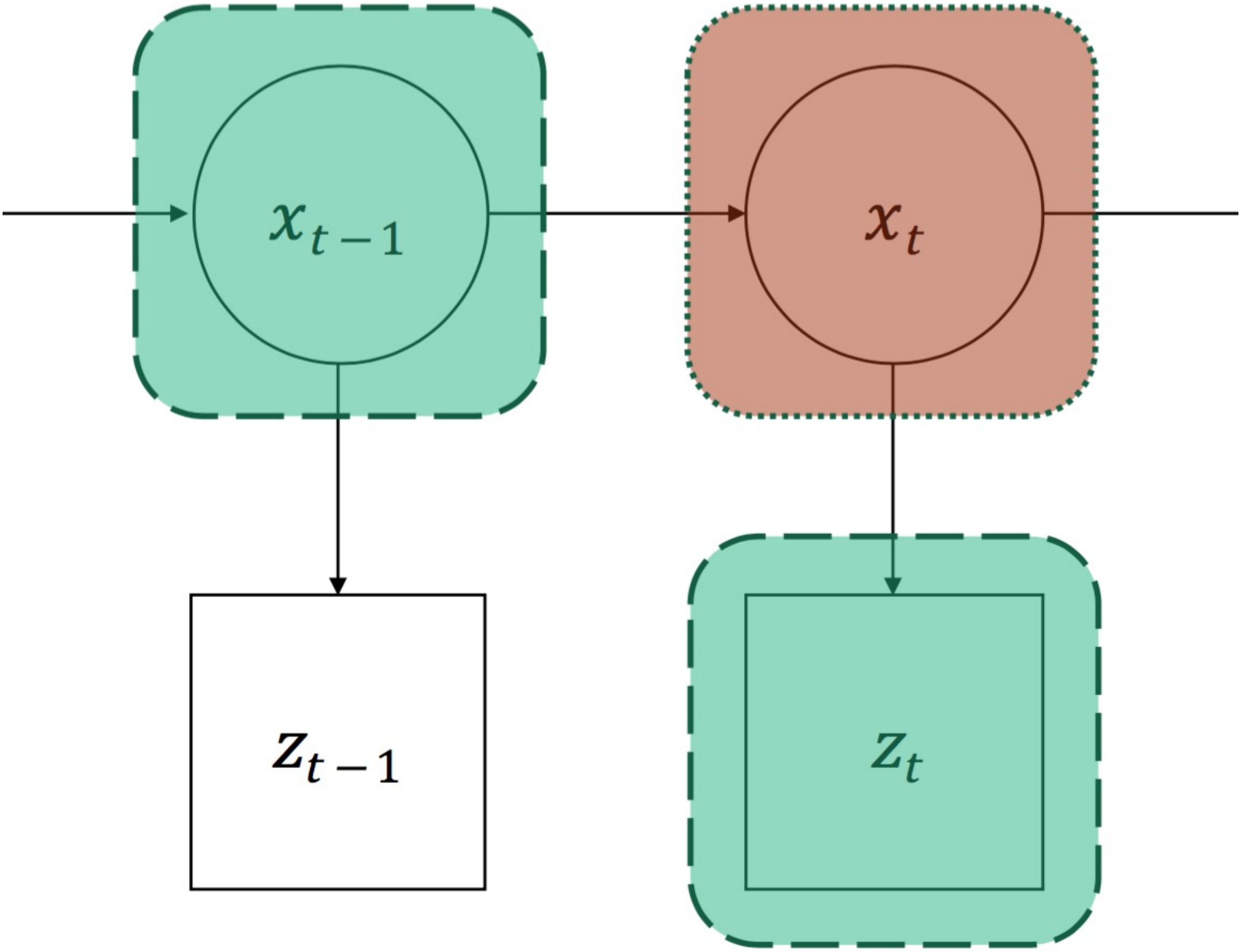


目标追踪之卡尔曼滤波

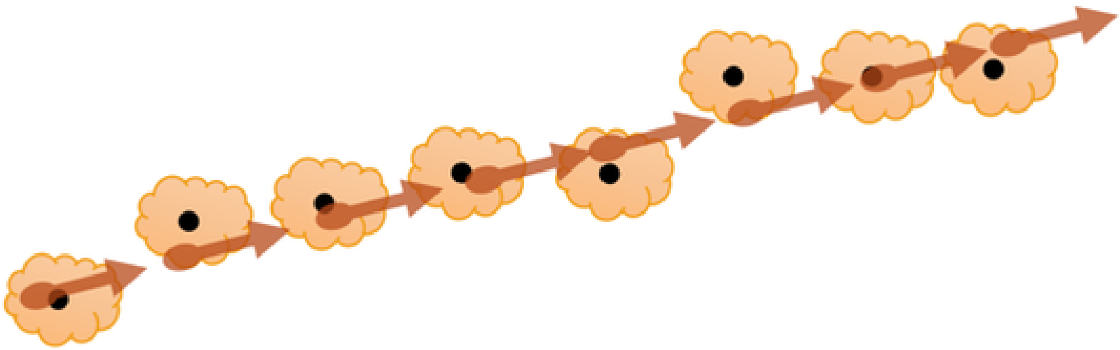
ChingKitWong · 9 个月前

最近在看Coursera的robotic learning，发现挺有意思的。这里算是做一下week 2的用卡尔曼滤波来做机器人目标追踪的笔记，紧接着上一篇的《从高斯分布、机器人误差、EM算法到小球检测》。

这篇小文章主要有两个内容

卡尔曼滤波建模

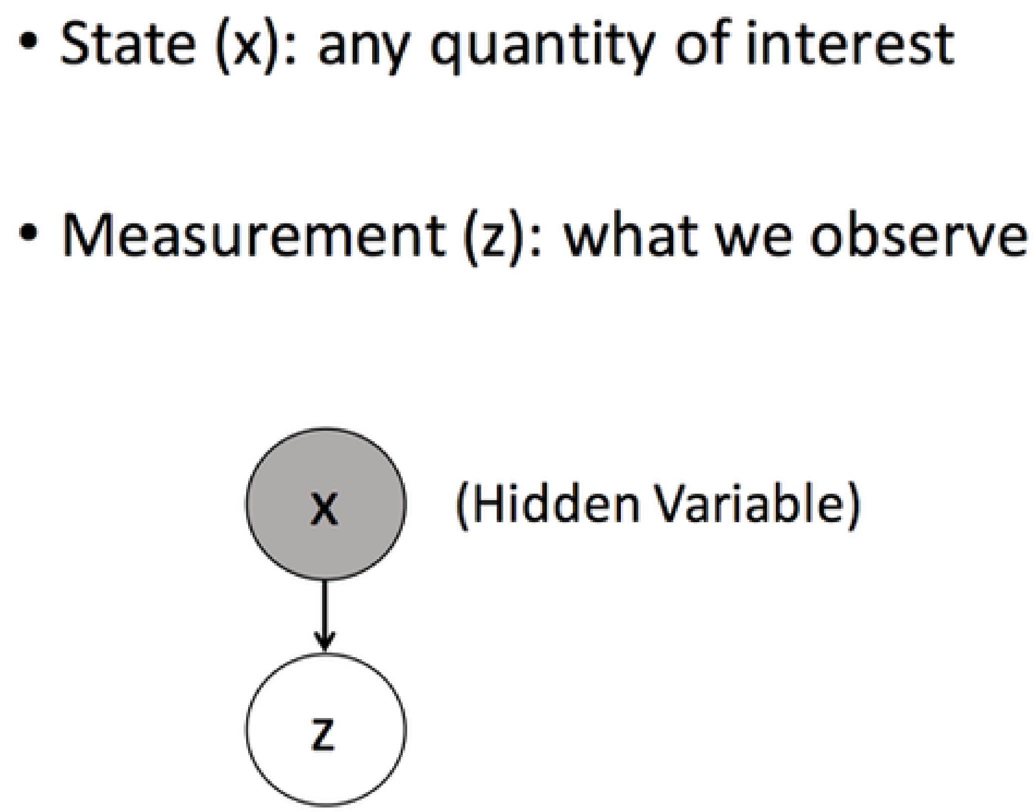
滤波，简要来说就是如何过滤掉收到干扰的信号。卡尔曼滤波所做的是从一堆观测数据中，去估计出真实数据样子。举一个简单的例子，就说机器人的目标追踪。人感知这个世界靠的是眼睛，机器人感知这个世界靠的是传感器。机器人通过传感器测量物体位置的时候，传感器可能带来测量误差。例如，上一篇利用高斯分布来检测小球的文章，我们很有可能由于遮挡、模型的精度不够导致我们测量的小球中心的位置会有些许误差。例如下图中，我们测量的可能是带有噪声的黑点，那么真正的轨迹究竟是如何的呢？



我们把我们所关心的真实的小球当前的状态称之为状态（State）x，传感器测量得到的小球位置称之为测量值（Measurement）z。其中，x为隐含变量。

由于看的公开课学的kalman filter，所以文章中用到的数学符号大部分跟随课件。

另外是Coursera这门课上讲课的那个小哥的ppt总是感觉写的不太严谨，所以这里我另外自己推导了一下。



所有的模型最重要的开始就是它的假设，卡尔曼滤波的第一个假设是当前的状态只与前一个状态相关。用数学表达是

$$p(x_t|x_{t-1},...,x_0)=p(x_t|x_{t-1})$$

接下来是卡尔曼滤波的第二个假设，由于我们对真实的状态并不知道，所以我们估计的状态实际上都有一定的不确定性，而这种不确定性我们用高斯分布的均值以及方差来表示。对于上一个状态 x_{t-1} ，假设我们估计其均值为 $\hat{x}_{t-1}$ 方差为 $\hat{P}_{t-1}$ 。则有

$$p(x_{t-1})\sim N(\hat{x}_{t-1},\hat{P}_{t-1}) \quad (1)$$

根据离散动力系统的构造，我们能得到 x_t 与 x_{t-1} 的关系

$$x_t=Ax_{t-1}+\epsilon \quad (2)$$

其中A为系统的状态转移矩阵， ϵ 代表系统误差并服从高斯分布。在我们的小球追踪实验中，x为小球的位置以及速度，A矩阵则是根据位移以及速度的关系构造。

$$x=[v \quad \frac{dv}{dt}],A=\begin{pmatrix}1 & dt\\0 & 1\end{pmatrix}$$

$p(x_t|x_{t-1}) \sim N(Ax_{t-1}, \Sigma_m)$ (3)

那么我们可以根据公式(1)和公式(3)，计算出 $p(x_t)$ 的边缘概率分布。

其边缘概率分布服从

$p(x_t) \sim N(A\hat{x}_{t-1}, A\hat{P}_{t-1}A^T + \Sigma_m)$ （5）

懒得推导可以根据PRML这书的这一页纸直接得到结论，需要详细推导的话可以看看书（PRML第二章）。

> **Marginal and Conditional Gaussians**
>
> Given a marginal Gaussian distribution for $\mathbf{x}$ and a conditional Gaussian distribution for $\mathbf{y}$ given $\mathbf{x}$ in the form
>
> $$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) \tag{2.113}$$
> $$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}) \tag{2.114}$$
>
> the marginal distribution of $\mathbf{y}$ and the conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ are given by
>
> $$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}}) \tag{2.115}$$
> $$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}\{\mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}\}, \boldsymbol{\Sigma}) \tag{2.116}$$
>
> where
>
> $$\boldsymbol{\Sigma} = (\boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1}. \tag{2.117}$$

接下来是对观测值建模，观测值 z_t 与 x_t 的关系，则为

$z_t = Cx_t + \epsilon$ (6)

这里C是从真实值到观察值的状态转移矩阵，ϵ 为观测误差并服从 $N(0, \Sigma_0)$ 的高斯分布。则同理，$p(z_t|x_t)$ 服从 Cx_t 为中心，Σ_0 为方差的高斯分布。

$p(z_t|x_t) \sim N(Cx_t, \Sigma_0)$ (7)

我们的目标是希望**最大化后验概率**。

$$\hat{x}_t = \underset{x_t}{\operatorname{argmax}}\ p(x_t|z_t)$$

说人话的话，就是一开始当我们还没观测到结果，我们就有 x_t 在此时刻的状态大概是怎么样的情况，也就是先验，如今当我们观测到了结果 z_t，我们需要如何修正 x_t 的概率分布呢？我们需要求

$$P = A\hat{P}_{t-1}A^T + \Sigma_m$$
$$R = \Sigma_0$$

根据贝叶斯公式（或者直接套用上面PRML的图），我们可以直接得出后验概率为

$$p(x_t|z_t) \sim N((P^{-1} + C^TR^{-1}C)^{-1}(C^TR^{-1}z_t + P^{-1}A\hat{x}_{t-1}), (P^{-1} + C^TR^{-1}C)^{-1})$$

那么估计值 $\hat{x}_t$ 应该为高斯分布的均值

$$\hat{x}_t = \underset{x_t}{\operatorname{argmax}}\ p(x_t|z_t) = (P^{-1} + C^TR^{-1}C)^{-1}(C^TR^{-1}z_t + P^{-1}A\hat{x}_{t-1})$$

而估计值的方差则为

$$\hat{P}_t = (P^{-1} + C^TR^{-1}C)^{-1}$$

为了表达上的优美，我们将 $\hat{x}_t$ 的公式做一下变换

先给出一个**矩阵逆定理**

$$(C^TR^{-1}C + P^{-1})^{-1} = P - PC^T(R + CPC^T)^{-1}CP$$

我们令 $K = PC^T(R + CPC^T)^{-1}$

那么

$$\hat{x}_t = (C^TR^{-1}C + P^{-1})^{-1}(C^TR^{-1}z_t + P^{-1}A\hat{x}_{t-1})$$
$$\hat{x}_t = (P - KCP)(C^TR^{-1}z_t + P^{-1}A\hat{x}_{t-1})$$
$$\hat{x}_t = A\hat{x}_{t-1} + K(z_t - CA\hat{x}_{t-1})$$

我们可以这么理解上面这一条公式， $A\hat{x}_{t-1}$ 应该为t时刻中，没有观察值预测到的状态。$z_t - CA\hat{x}_{t-1}$ 为预测的观察值与实际观察值的差距。然后通过一个 K 来调整最终的 $\hat{x}_t$ 估计值。所以 K 在卡尔曼滤波中，有一个特殊的意义，叫做**卡尔曼增益。**

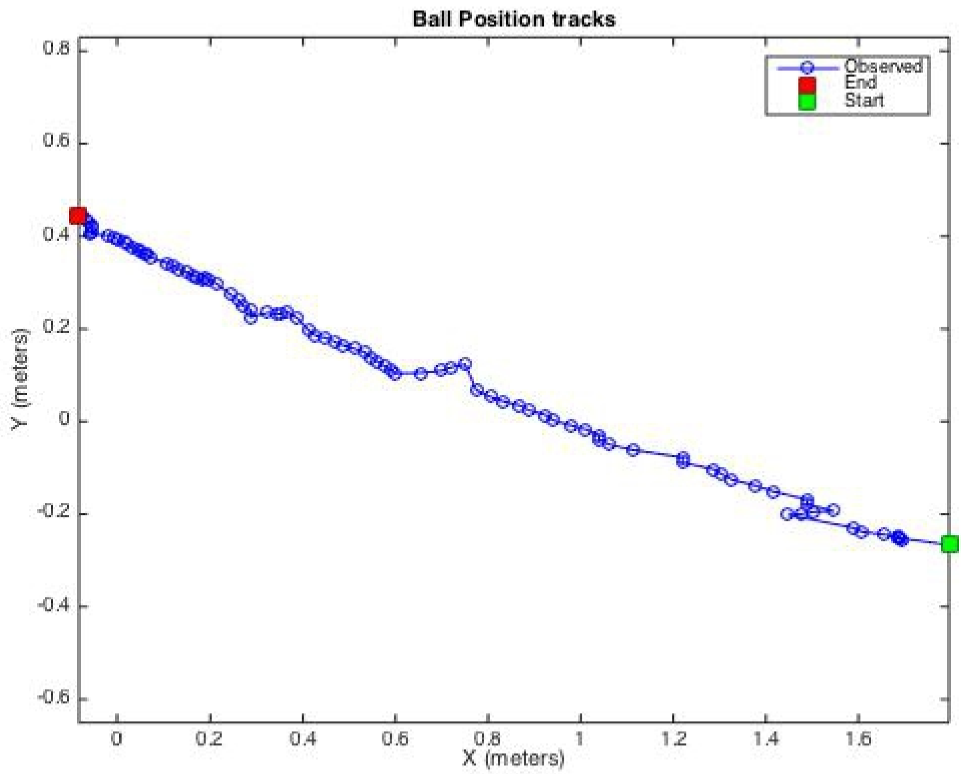
整个卡尔曼滤波的算法基本如下：

1. 设定初始状态的均值和方差 $\hat{x}_0, \hat{P}_0, \Sigma_m, \Sigma_0$ ， $t = 1$
2. 接受下一个观测值 z_t
3. 计算方差 P, R ，以及卡尔曼增益 K

5. $t = t + 1$，回到2

实战：小球追踪实验

这次的仿真实验给出了小球每一帧的运动轨迹，我们需要在每一个时刻中预测出小球在10帧之后位置，具体的要求见Requirement。



小球的轨迹如上图所示，为了做对比，我们引入一种最朴素的算法作为估计。对于每一时刻的 x_t，我们估计它的速度为

$$v_t = \frac{x_t - x_{t-1}}{t}, t = 1$$

那么十帧之后，它的位置估计为

$$\hat{x_t} = x_t + 10v_t$$

我们把这种方法叫做Naive Prediction。

另外的一个方法，就是我们的卡尔曼滤波。首先，我们假定**x轴与y轴是相互独立的**。卡尔曼滤波中的状态我们设定为x轴，y轴坐标，以及x轴方向与y轴方向的速度。

$$x_t = [x, y, vx, vy]$$

$$A = \begin{pmatrix} 0, 1, 0, t \end{pmatrix}, t = 1$$

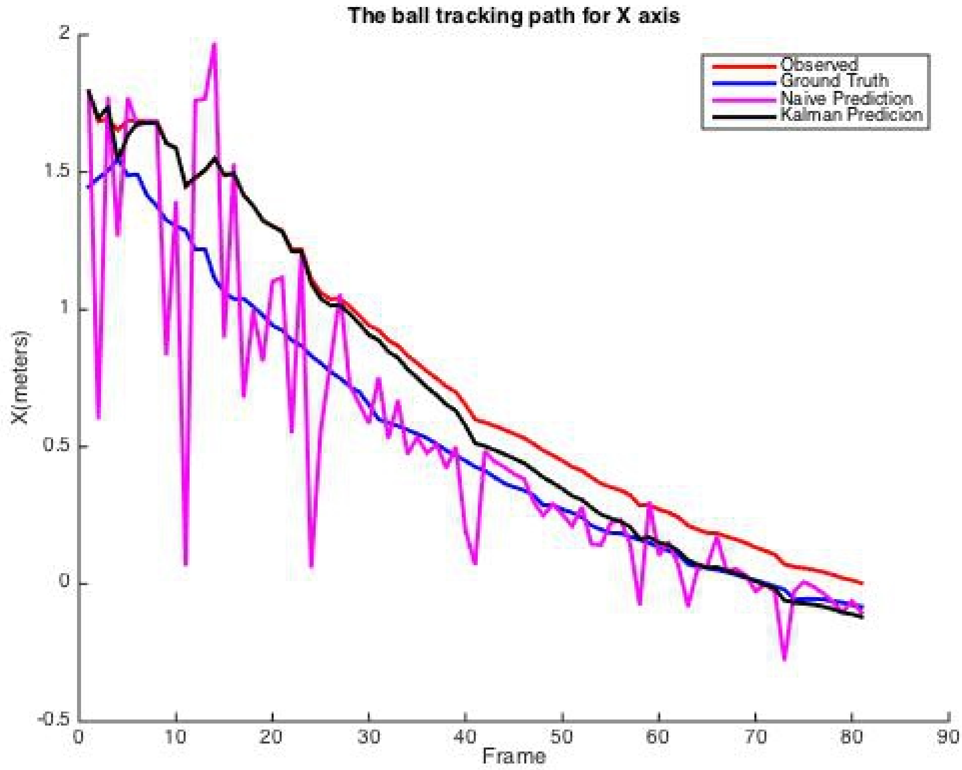
我们假定观测的 z_t 实际上是状态 x_t 在10帧之前的位置。则 C 矩阵为

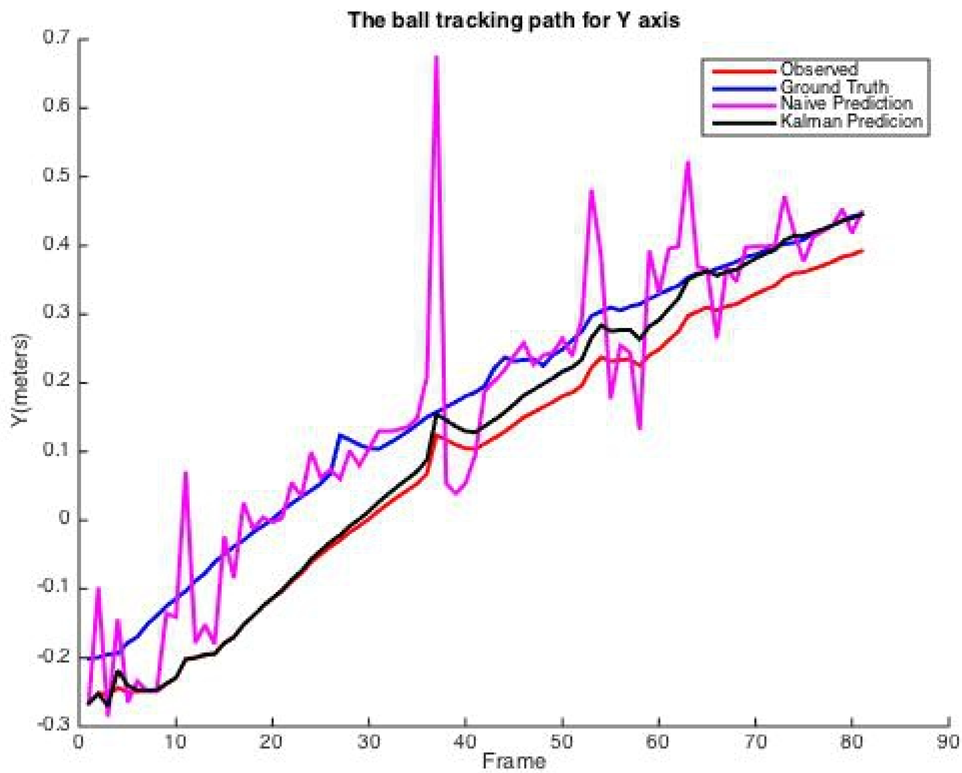
$$C = \begin{pmatrix} 1 & 0 & -10t & 0 \\ 0 & 1 & 0 & -10t \end{pmatrix}, t = 1$$

建立好动力系统后，接下来是对卡尔曼滤波的参数进行初始化，由于一开始我们并不知道初始状态的位置，所以我们可以把第一个观测值当作我们初始状态，然后加上一个**较大的初始方差**。剩下的系统误差的方差以及观测误差的方差则需要自己慢慢的调试了。具体的matlab代码如下

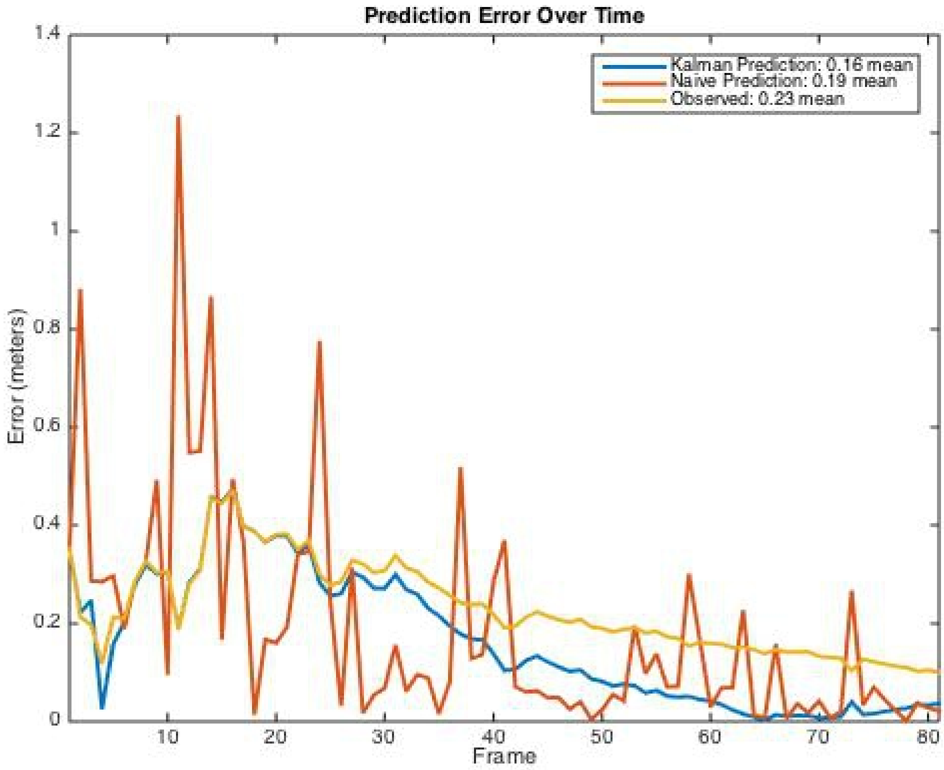
```
function [ predictx, predicty, state, param ] = kalmanFilter( t, x, y, state, param, previous_t
%UNTITLED Summary of this function goes here
%   Four dimensional state: position_x, position_y, velocity_x, velocity_y

    %% Place parameters like covaraince, etc. here:
    sigm =  eye(4);
    sigo =  1e-3*eye(2);
    dt = t- previous_t;
    A = [1 0 dt 0 ;0 1 0 dt;0 0 1 0;0 0 0 1];
    C = [1 0 -10* dt 0;0 1 0 -10*dt];
    % Check if the first time running this function
    if previous_t<0
        state = [x,y,0,0];
        param.P = 10 * sigm;
        param.R =  sigo;
        predictx = x;
        predicty = y;
        return;
    end
    param.R = sigo;
    P = A*param.P*A'+ sigm;
    K = P*C'/(param.R+C*P*C');
    state = A*state'+ K*([x,y]'-C*A*state');
    state = state';
    param.P = P - K*C*P;
    predictx = state(1);
    predicty = state(2);
    return
end
```

由于预测的10帧后的轨迹会与上图的轨迹重合，为了更好地可视化出我们的目标，我们另外将x轴





上图是x轴以及y轴的追踪效果，其中红线代表每一时刻的观测值，蓝线是10帧之后我们需要预测



这里算是告一段落了，这篇文章只是粗略地介绍一下卡尔曼滤波算法，以及做了一个小仿真实验。在我个人看来，卡尔曼滤波算法中，有很多人为设定的东西，例如A、C两个状态转移矩阵，还有不同的方差设定。要能准确的调节这些参数，我觉得最重要的一点就是理解算法的本身，理解每一个假设还有求解中间的过程。这也是我之所以比较多的写中间卡尔曼滤波推导过程的原因，最好还是自己动手做一做。欢迎讨论~

===

写于2016年8月23日

补充一下$K=PC^T(R+CPC^T)^{-1}=(C^TR^{-1}C+P^{-1})^{-1}C^TR^{-1}$的推导，主要还是参考《Probabilistic Robotics》里面的推导公式。只是照搬书上的公式，套回来这里用的数学符号。

$$
\begin{aligned}
K &= (C^TR^{-1}C+P^{-1})^{-1}C^TR^{-1} \\
&= (C^TR^{-1}C+P^{-1})^{-1}C^TR^{-1}(CPC^T+R)(CPC^T+R)^{-1} \\
&= (C^TR^{-1}C+P^{-1})^{-1}(C^TR^{-1}CPC^T+C^TR^{-1}R)(CPC^T+R)^{-1} \\
&= (C^TR^{-1}C+P^{-1})^{-1}(C^TR^{-1}CPC^T+C^T)(CPC^T+R)^{-1} \\
&= (C^TR^{-1}C+P^{-1})^{-1}(C^TR^{-1}CPC^T+P^{-1}PC^T)(CPC^T+R)^{-1} \\
&= (C^TR^{-1}C+P^{-1})^{-1}(C^TR^{-1}C+P^{-1})PC^T(CPC^T+R)^{-1} \\
&= PC^T(CPC^T+R)^{-1}
\end{aligned}
$$

赞赏

2 人赞赏

卡尔曼滤波 Kalman Filter 机器人 自动控制

收藏 分享 举报

67

...

29 条评论

写下你的评论

郑华滨

66666，拜读一下

9 个月前

江山

倒车位置不准？来一发卡尔曼滤波

9 个月前

雪落语城

楼主，可以了解一下你下面那几周作业么？或者你写在哪里没

9 个月前

知 首发于
当机器人遇上了学习

写文章

https://www.coursera.org/learn/robotics-learning/home/week/2
，我自己做的作业答案就是上面这一份。至于画图之类的，公开课里面的代码有给例子。

9 个月前

拉维必达索尔

一直觉得K没有什么意义，这个卡尔曼增益完全是为了简化公式而单拉出来的。您对此怎么看？

9 个月前

ChingKitWong（作者） 回复 **拉维必达索尔** 查看对话

按照公式推理来说，他确实只是做了个公式简化的效果。但我觉得卡尔曼增益这个概念提供了另外一种理解这个算法的方式。

9 个月前

江山 回复 **雪落语城** 查看对话

前三周的都在专栏里面有详细讲解呀~

9 个月前

白毅

mark一下，开完会看看

9 个月前

徐磊

有个问题想问下。。。想要学习卡尔曼滤波，只求能熟练应用，不求推导，但本科生的数学储备不够，是应该直接找那种应用类的书看还是补充数学知识（想自己做陀螺仪数据融合，网上代码一堆，但我想弄懂这玩意）

9 个月前

ChingKitWong（作者） 回复 **徐磊** 查看对话

因为我本身不是做这个的，我只是碰巧上了公开课，应用书籍方面做不了太多推荐。至于数学

上的话，基本的高等数学和概率统计稍微看一下应该就没问题了。

9 个月前 1 赞

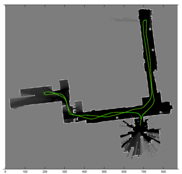
文章被以下专栏收录

当机器人遇上了学习

机器人学习（Robot Learning）

进入专栏

推荐阅读



蒙特卡罗定位（Particle Filter Localization）

江山 · 8 个月前　　发表于 当机器人遇上了学习

占据栅格地图（Occupancy Grid Map）

江山 · 9 个月前　　发表于 当机器人遇上了学习

运动牙套的物种起源：库里叼着它可不止为了耍酷

懒熊体育 · 20 天前 · 编辑精选　　发表于 懒熊体育

Monster Farm 开发日志（三） 体素游戏美术心得

Lea Liu · 2 个月前 · 编辑精选　　发表于 纽大的游戏故事