第 5 课 UKF 与粒子滤波实现

作者：范泽宣

对于卡尔曼的方法而言，最常用的也就是经典卡尔曼和扩展卡尔曼，卡尔曼应用的弊端也很明显，比如并不是所有测量系统的误差分布都是高斯的，而且在 EKF 在线性化的过程中引入了误差，甚至会造成算法发散，而且在一般情况下 Jacobian 矩阵是不易计算的，增加了算法的复杂度。

这节课我们从 UKF（无迹卡尔曼）说起，有了 UKF 我们就能很容易理解粒子滤波，UKF 是 S.Julier 等人提出的一种非线性滤波的方法，与 EKF 不同的是，它并不是像 EKF 一样对非线性方程 f 和 h 在估计点处做线性化逼近，而是用无迹变换在估计点附近确定采样点，用这些采样点表示的高斯密度近似状态的概率密度函数。在 CCTV 亚太机器人大赛中，笔者曾经尝试将 UKF 的方法引入基于陀螺仪码盘的全场定位算法之中，实际效果要好于线性滤波方法，这也充分说明了该方法解决复杂运动模型的通适性。

无迹变换的实现方法是在原状态分布中按某一规律选取一些采样点，使这些采样点的均值和协方差等于原状态分布的均值和协方差；将这些点代入非线性函数中，相应得到非线性函数值的点集，通过这些点集求取变换后的均值和协方差。

略微了解了 UKF 之后，我们先停一下，一会再实例介绍 UKF 的算法实现，先来看下什么是粒子滤波，请注意粒子滤波在思想上和上面介绍的 UKF 的相似性。

粒子滤波器是贝叶斯滤波器的一种非参数执行情况，我们来复习下什么是贝叶斯规则，贝叶斯公式如下：

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

分母是标准化常数，用于确保左边的后验概率其所有可能的值之和为 1。因此，我们通常可写成：

$$p(x) = \eta p(y|x)p(x)$$

在背景知识 e 给定的情况下，贝叶斯规则变成：

$$p(x|y,\mathrm{e}) = \frac{p(y|x,e)p(x|e)}{p(y|e)}$$

粒子滤波的关键思想采用一大群粒子来表示后验概率，其中每一个粒子代表了你的控制对象（如你的钢铁侠盔甲系统）可能存在的一种潜在状态，状态

假设的概率和可表示为：

$$\mathrm{p(x)} = \sum_{i=1}^{N} w_i \delta_{s[i]}(x) = 1$$

$\delta_{s[i]}$是在第 i 个样本的状态 s[i]下的狄拉克函数，样本集 s 可用于近似任意分布（钢铁侠盔甲可能的运动方向和位置）。粒子滤波利用的样本集合对多模态分布模型建模的能力和 UKF 很像，有没有发现他们宏观上的相似呢。

粒子滤波方法的实现过程是怎样的呢，1.采样；2.重要性加权；3.从采样。

一般来说，样本粒子的真实概率分布是未知的或者是一种不合适采样的形式。正是通过 2.重要性加权从不同于我们想要近似的分布中提出采样。

假设我们要计算 x（钢铁侠盔甲出现在区域 A 中的位置），定义函数 f 的期望如下：

$$E_p[f(x)] = \int p(x) f(x) dx$$

定义一个函数 B：如果参数为真，它的返回值为 1，否则为 0。我们可以通过一下表达式表示这个期望：

$$E_p[B \in A] = \int p(x) B(x \in A) dx = \int \frac{p(x)}{\pi(x)} \pi(x) B(x \in A)\, dx$$

其中$\pi(x)$是一个分布，满足：

$$p(x) > 0 \Longrightarrow \pi(x) > 0$$

因此可以定义权重 w 为：

$$\mathrm{w}(x) = \frac{p(x)}{\pi(x)}$$

权重 w 用于说明 p 和π之间的差异。从而推出：

$$E_p[B(x \in A)] = \int \pi(x) w(x) B(x \in A)\, dx = E_\pi[w(x) B(x \in A)]$$

让我们重新思考基于样本的表示并假设这个样本来自于π。通过计算落入区域 A 的所有粒子，可以通过累加样本来计算π在区间 A 上的积分，即：

$$\int_{\mathrm{A}} \pi(x) dx \approx \frac{1}{N} \sum_{i=1}^{N} B(s^{[i]} \in A)$$

此处笔者本着课程的实用工程原则，不再详述粒子滤波算法的推导公式，因为即使你看完了推导公式，程序也不一定会写，对相关数学感兴趣的同学推荐去看下各种书籍中介绍的推导方法，推荐下 M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp 的《A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking》这篇论文。

让我们来结合实例看下 python 实现粒子滤波的程序：

```
def particle_filter(motions, measurements, N=500): # I know it's tempting, but don't change N!
    # --------
    #
    # 首先生成粒子
```

```
        #
        p = []
        for i in range(N):
            r = robot()
            r.set_noise(bearing_noise, steering_noise, distance_noise) //此函数设置噪声，可随意编写
            p.append(r)//生成了你运动对象的 N 中可能的运动状态
        # --------
        #
        # Update particles
        #

        for t in range(len(motions)):

            # motion update (prediction)
            p2 = []
            for i in range(N):
                p2.append(p[i].move(motions[t]))//此处为机器人运动，可宏观理解，粒子滤波的具体实现在下面
            p = p2

            # measurement update
            w = []
            for i in range(N):
                w.append(p[i].measurement_prob(measurements[t]))

            #  粒子滤波实现，从采样过程
            p3 = []
            index = int(random.random() * N)
            beta = 0.0
            mw = max(w)
            for i in range(N):
                beta += random.random() * 2.0 * mw//重要性加权
                while beta > w[index]:
                    beta -= w[index]
                    index = (index + 1) % N
                p3.append(p[index])
            p = p3
        return get_position(p)
```

粒子滤波的程序实现就是如此简单，这个样本方法常常可以解决观察系统不易建立或关联已知数学模型的情况。我们再回来看下 UKF，对于不同时刻 k，由具有高斯白噪声 W(k)的随机变量 X 和具有高斯白噪声 V(k)的观测变量 Z 构成的非线性系统可以由以下公式描述：

$$\begin{cases} \mathrm{X}(K+1) = f(x(k), w(k)) \\ Z(k) = h(x(k), v(k)) \end{cases}$$

随机变量 X 在不同时刻 k 的无迹 Kalman 滤波算法基本步骤如下：

（1） 获取一组采样点（Sigma 点集）及其对应权值：

$$X^{(i)}(k|k) = [\hat{X}(k|k) \quad \hat{X}(k|k) + \sqrt{(n+\lambda)P(k|k)} \quad \hat{X}(k|k) - \sqrt{(n+\lambda)P(k|k)}\,]$$

（2） 计算 2n+1 个 Sigma 点集的一步预测：

$$X^{(i)}(k|k) = f[k, \hat{X}(k|k)]$$

（3） 计算系统状态量的预测和协方差矩阵，不同于粒子滤波的细节是这一步利用一组 Sigma 点的预测计算对点集的加权平均，得到系统状态的下一步预测：

$$\hat{X}(k+1|k) = \sum_{i=0}^{2n} w^{(i)} X^{(i)}(k+1|k)$$

$$P(k+1|k) = \sum_{i=0}^{2n} w^{(i)} \left[\hat{X}(k+1|k) - X^{(i)}(k+1|k)\right]\left[\hat{X}(k+1|k) - X^{(i)}(k+1|k)\right]^T$$

（4） 根据一步预测值，再次使用 UT 变换，产生新的 Sigma 点集：

$$X^{(i)}(k+1|k) = [\hat{X}(k+1|k) \quad \hat{X}(k+1|k) + \sqrt{(n+\lambda)P(k+1|k)} \quad \hat{X}(k+1|k) - \sqrt{(n+\lambda)P(k+1|k)}\,]$$

（5） 将由步骤（4）预测的 Sigma 点集代入观测方程，得到预测的观测量 *i=1*、*2*、*……2n+1*。

$$Z^{(i)}(k+1|k) = h[X^{(i)}(k+1|k)]$$

（6） 由步骤（5）得到 Sigma 点集的观测预测值，通过加权求和得到系统预测的均值及协方差：

$$\bar{\mathrm{Z}}(k+1|k) = \sum_{i=0}^{2n} w^{(i)} Z^{(i)}(k+1|k)$$

$$P_{Z_k Z_k} = \sum_{i=0}^{2n} w^{(i)} [\, Z^{(i)}(k+1|k) - \bar{Z}(k+1|k)][\, Z^{(i)}(k+1|k) - \bar{Z}(k+1|k)]^T + R$$

$$P_{X_k Z_k} = \sum_{i=0}^{2n} w^{(i)} [\, Z^{(i)}(k+1|k) - \bar{Z}(k+1|k)][\, Z^{(i)}(k+1|k) - \bar{Z}(k+1|k)]^T$$

（7） 计算 Kalman 增益矩阵：

$$K(k+1) = P_{Z_k Z_k} P_{X_k Z_k}$$

（8） 最后，计算系统的状态更新和协方差更新：

$$\hat{X}(k+1|k+1) = \hat{X}(k+1|k) + \mathrm{K}(k+1)[Z(k+1) - \hat{Z}(k+1|k)]$$

$$P(k+1|k+1) = P(k+1|k) - K(k+1)P_{Z_k Z_k} K^T(k+1)$$

看看以上的详细算法过程，和之前的几节课对比后，有没有发现 UKF 在非线性滤波时并不需要做泰勒级数的展开，而是在估计点附近进行 UT 变换，使获得的 Sigam 点集的均值和协方差与原统计特性匹配，再直接对这些 Sigma 点集进行非线性映射，以近似得到状态概率密度函数，上述算法的 matlab 实现如下：

```
for t=2:N
    X(:,t)=F*X(:,t-1)+G*sqrtm(Q)*randn(2,1);
end
for t=1:N
    Z(t)=Dist(X(:,t),Xstation)+w(t);
end
L=4;
alpha=1;
kalpha=0;
belta=2;
ramda=3-L;
for j=1:2*L+1
    Wm(j)=1/(2*(L+ramda));
    Wc(j)=1/(2*(L+ramda));
end
Wm(1)=ramda/(L+ramda);
Wc(1)=ramda/(L+ramda)+1-alpha^2+belta;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
Xukf=zeros(4,N);
Xukf(:,1)=X(:,1);
P0=eye(4);
for t=2:N
    xestimate= Xukf(:,t-1);
    P=P0;
    cho=(chol(P*(L+ramda)))';
    for k=1:L
        xgamaP1(:,k)=xestimate+cho(:,k);
        xgamaP2(:,k)=xestimate-cho(:,k);
    end
    Xsigma=[xestimate,xgamaP1,xgamaP2];
```

```
    Xsigmapre=F*Xsigma;
    Xpred=zeros(4,1);
    for k=1:2*L+1
        Xpred=Xpred+Wm(k)*Xsigmapre(:,k);
    end
    Ppred=zeros(4,4);
    for k=1:2*L+1
        Ppred=Ppred+Wc(k)*(Xsigmapre(:,k)-Xpred)*(Xsigmapre(:,k)-
Xpred)';
    end
    Ppred=Ppred+G*Q*G';
    chor=(chol((L+ramda)*Ppred))';
    for k=1:L
        XaugsigmaP1(:,k)=Xpred+chor(:,k);
        XaugsigmaP2(:,k)=Xpred-chor(:,k);
    end
    Xaugsigma=[Xpred XaugsigmaP1 XaugsigmaP2];
    for k=1:2*L+1
        Zsigmapre(1,k)=hfun(Xaugsigma(:,k),Xstation);
    end
    Zpred=0;
    for k=1:2*L+1
        Zpred=Zpred+Wm(k)*Zsigmapre(1,k);
    end
    Pzz=0;
    for k=1:2*L+1
        Pzz=Pzz+Wc(k)*(Zsigmapre(1,k)-Zpred)*(Zsigmapre(1,k)-Zpred)';
    end
    Pzz=Pzz+R;

    Pxz=zeros(4,1);
    for k=1:2*L+1
        Pxz=Pxz+Wc(k)*(Xaugsigma(:,k)-Xpred)*(Zsigmapre(1,k)-Zpred)';
    end
    K=Pxz*inv(Pzz);
    xestimate=Xpred+K*(Z(t)-Zpred);
    P=Ppred-K*Pzz*K';
    P0=P;
    Xukf(:,t)=xestimate;
end
for i=1:N
    Err_KalmanFilter(i)=Dist(X(:,i),Xukf(:,i));
end
```

卡尔曼滤波如何和粒子滤波结合使用呢，一般是粒子滤波结合卡尔曼涉及

到 slam 可以看 youtube 上 Cyrill Stachniss 教授的课程，有空，笔者将会翻译该课程为中文教程。

通过这 5 节课的内容相信你应该可以对卡尔曼滤波有了一个宏观及细致的了解了，也可以编写相应应用的卡尔曼程序了吧，不论怎样，卡尔曼滤波应用相当广泛，不论各行各业，只要应用传感器、单片机的应用都可以应用卡尔曼，希望这几节课对你的卡尔曼工程应用有所帮助，卡尔曼滤波课程就到这里吧，就到这里啦。

本文参考：1. EXTENDED KALMAN FILTER DESIGN USING BEARING AND TIME-TO-GO MEASUREMENT FOR A HOMING MISSILE GUIDANCE

2. 《卡尔曼滤波原理及应用》黄小平，王岩编著

3. Artificial Intelligence for Robotics Programming a Robotic Car

4. Cyrill Stachniss 《Robotic Mapping and Exploration》