文章编号: 1001-0920(2019)03-0561-06 DOI: 10.13195/j.kzyjc.2017.1135

基于KLD的蝙蝠算法优化自适应粒子滤波

滕 飞†, 薛 磊, 李修和

(国防科技大学 电子对抗学院，合肥 230037)

摘 要: 针对粒子滤波存在计算效率低和因粒子贫化导致的计算精度下降问题,基于KLD(Kullback Leibler distance)采样和蝙蝠算法,提出一种可动态调整粒子规模的自适应粒子滤波算法. 首先,在重要性采样中利用KLD采样动态调整粒子规模;然后,使用蝙蝠算法定向优化粒子集,并在迭代更新中使蝙蝠算法和KLD采样相互作用,从而达到同时提升计算精度和计算效率的目的. 实验结果验证了所提出算法的可行性和有效性.

关键词: 粒子滤波；蝙蝠算法；KLD采样；粒子多样性；状态估计；非线性非高斯

中图分类号: TP273 **文献标志码:** A

Adaptive particle filter with bat optimization based on KLD sampling

TENG Fei†, XUE Lei, LI Xiu-he

(Electronic Countermeasures Institute，National University of Defense Technology，Hefei 230037，China)

Abstract: In order to solve the problem of particle impoverishment and low computational efficiency of a particle filter, an adaptive particle filter which can dynamic adjust the particle set size based on Kullback Leibler distance (KLD) is presented in this paper. Firstly, KLD sampling is used to dynamiclly adjust the particle set size in importance sampling. Then, the bat algorithm is used to optimaize the particle set purposefully, and it interacts with KLD sampling in iterative updates, so as to achieve the purpose of enhancing calculation accuracy and efficiency. The simulation results show the feasibility and effectiveness of the proposed algorithm.

Keywords: particle filter; bat algorithm; KLD sampling; particle diversity; state estimation; non-linear and non-Gaussian

0 引 言

粒子滤波算法是一种基于Monte Carlo理论的滤波算法. 将复杂积分转化为加权求和运算,可有效地估计非线性动态模型的隐状态. 目前,粒子滤波及其智能优化算法广泛地应用于目标跟踪、自动控制及故障检测等领域.

粒子滤波算法主要由重要性采样和重采样两部分组成. 其中,重要性采样产生实现目标状态估计的加权粒子集. 但随着迭代步数递增,粒子集的权重逐渐偏向少数粒子,使权值方差逐步增大,大部分粒子变为无效粒子,整个粒子集的有效性下降,即产生粒子退化现象. 为了解决粒子退化的问题,Gordon等[1]提出了重采样算法,其算法核心思想是复制大权值粒子并删除小权值粒子. 虽然该算法解决了粒子退化问题,但造成粒子多样性匮乏[2],即粒子贫化问题,粒子贫化会严重影响粒子滤波算法的性能. 因此,针对粒子贫化问题的改进算法一直是研究的热点[3].

目前,基于群体智能优化思想的粒子滤波算法,作为一种解决粒子贫化问题的新思路,得到了国内外学者广泛研究. 其中,Fang等[4]提出了一种基于粒子群算法的粒子滤波算法,利用粒子群算法的定向寻优能力改进粒子集的分布. 在此基础上,又出现了许多改进算法. Qiu等[5]提出了粒子群算法和蚁群算法的融合寻优算法,实现了粒子集间的信息共享,增强了算法的全局寻优能力. Klamargias等[6]提出了对大权重粒子多次重复采样的粒子群粒子滤波算法,使粒子向高似然区域逼近,从而提高了算法的计算精度. 萤火虫算法作为近几年较为热门的智能优化算法,是由剑桥大学的Yang教授[7]于2010年提出的. 在此基础上,Tian等[8]提出了基于萤火虫算法优化的粒子滤波,通过亮度和吸引度动态优化粒子集,从而解决了重采样带来的粒子贫化问题.

收稿日期: 2017-08-26；**修回日期:** 2017-10-27.

基金项目: 武器装备预研重点基金项目(9140A33020112JB39085).

责任编委: 陈家伟.

作者简介: 滕飞(1990−), 男, 博士生, 从事多目标跟踪、随机集理论及其应用的研究；薛磊(1962−), 男, 教授, 博士生导师, 从事电信技术等研究.

†通讯作者. E-mail: tfmmt@foxmail.com.

随着萤火虫算法的提出,Yang教授[9-11]又提出了蝙蝠算法. 相对于萤火虫算法由亮度和吸引度双因子驱动,蝙蝠算法由搜索频率、脉冲强度和脉冲频率三因子驱动,自适应性更强,并且采用全局寻优和局部寻优双寻优策略,使算法的收敛速度更快,摆脱局部最优的能力更强,因此,利用该算法改进粒子滤波是一条新的思路. Chen[12]于2017年提出了基于蝙蝠算法的粒子滤波算法(Bat algorithm optimized particle filter, BAPF),并通过仿真实验验证了该算法优于其他群体智能优化粒子滤波算法. 但是,基于群体优化算法的粒子滤波,均将寻优粒子数设为先验固定值. 在实际使用中,为保证算法的性能,通常使用较大的粒子采样规模,这样会导致算法效率大幅下降,并且,由于粒子滤波的加权求和步骤,过多的冗余粒子也会导致计算精度下降.

针对粒子滤波因粒子贫化而导致的计算精度下降问题和采用群体智能优化后带来的计算效率下降问题,基于KLD采样算法和蝙蝠寻优算法,本文提出一种基于KLD的蝙蝠算法优化自适应粒子滤波算法(KLD sampling and bat algorithm optimized particle filter, KBPF). 该算法利用KLD采样动态删除冗余粒子,获得最优粒子数,再利用蝙蝠算法对新生最优粒子集定向寻优,使两种算法在迭代中相互作用,最终达到同时提升计算精度和计算效率的目的. 仿真实验的结果验证了本文算法的有效性.

1 粒子滤波算法分析

在非线性动态模型隐状态估计问题中,首先,针对目标问题构建动态模型,该模型由状态转移方程和状态观测方程组成,即

$$x_k = f(x_{k-1}) + N_{k-1}, \tag{1}$$

$$z_k = h(x_k) + R_k. \tag{2}$$

其中: $f(x_{k-1})$ 和 $h(x_k)$ 均为非线性方程, k 为估计步数, N_{k-1} 为状态转移噪声, R_k 为状态观测噪声.

粒子滤波算法基于Monte Carlo[13]学派的思想,首先,通过带权值的随机粒子集合 $\{x^i_{0:k}, w^i_{1:k}\}_N$,近似表示基于观测的后验概率密度函数 $p(x_{0:k}|z_{1:k})$,即

$$p(x_{0:k}|z_{1:k}) \approx \sum_{i=1}^{N} w^i_k \delta(x_{0:k} - x^i_{0:k}). \tag{3}$$

其中: k 为迭代步数, i 为第 i 个粒子, N 为粒子总数. 然后,通过后验概率密度函数估计目标状态的期望 $\mathrm{E}(x_k)$,即

$$\hat{x}_k = \mathrm{E}(x_k) = \int x_k p(x_{0:k}|z_{1:k}) \mathrm{d}x_{0:k}. \tag{4}$$

由于观测方程和状态方程的非线性,无法对 $p(x_{0:k}|z_{1:k})$ 直接采样,引入便于采样的重要性分布 $q(x_{0:k})$,则有

$$\mathrm{E}(x_k) = \int x_k \frac{p(x_{0:k}|z_{1:k})}{q(x_{0:k})} q(x_{0:k}) \mathrm{d}x_{0:k}. \tag{5}$$

其中: $\dfrac{p(x_{0:k}|z_{1:k})}{q(x_{0:k})}$ 可视为对重要性分布函数 $q(x_{0:k})$ 的加权,权值用 $w_{0:k}$ 表示,即

$$w_{0:k} = \frac{p(x_{0:k}|z_{1:k})}{q(x_{0:k})}. \tag{6}$$

状态转移过程中, k 步的目标状态只与 $k-1$ 步相关,因此,可将目标隐状态 $\{x_{0:k}\}$ 视为Markov Chain. 对式(6)进行推导,可得

$$w_k = w_{k-1} \frac{p(z_k|x_k)p(x_k|x_{k-1})}{q(x_k|x_{k-1})}. \tag{7}$$

其中: $p(z_k|x_k)$ 为似然分布, $p(x_k|x_{k-1})$ 为状态转移分布,由动态模型得到. 令重要性分布 $q(x_k|x_{k-1}) = p(x_k|x_{k-1})$. 最终得到加权粒子集 $\{x^i_k, w^i_k\}_N$ 的更新过程,即

$$x^i_k \sim q(x^i_k|x^i_{k-1}) = p(x^i_k|x^i_{k-1}), \tag{8}$$

$$w^i_k = w^i_{k-1} p(z_k|x^i_k). \tag{9}$$

对权值 w^i_k 归一化后,代入式(3)即可获得对后验概率密度的近似.

在加权粒子集的更新过程中,会使少数粒子的权值趋向于1,而大部分粒子的权值趋向于0[14]. 这导致大部分粒子的有效性趋向于0,即产生粒子退化现象. 针对该问题,Gordon等[1]提出了重采样算法.

重采样算法将所有粒子的权值统一变为 $1/N$,使退化问题得到了缓解. 但为了保留粒子集所代表的分布特性,重采样算法会复制大权值粒子和删除小权值粒子. 因而,导致重复粒子增多,即出现粒子贫化现象[15]. 由于粒子滤波基于Monte Carlo采样思想进行设计,采样多样性会有所下降,从而导致算法的整体性能下降. 因此,需要对重要性采样后的粒子进行针对性调整,以保持粒子的多样性.

2 KLD采样

KLD采样是一种利用KLD自适应调整采样规模的算法. 将其引入粒子滤波,当粒子集中时,估计确定性较高,需要较少粒子;当粒子分散时,估计不确定性高,需要较多粒子.

KLD是用来描述两个分布之间差距的度量,定义为

$$d(p,q) = \sum_x p(x) \log\left(\frac{p(x)}{q(x)}\right). \tag{10}$$

利用KLD调整粒子采样规模,需要采样的最优粒子数满足:真实后验分布与估计后验分布之间的

KLD小于τ的概率为$1-\delta$. 假设N个粒子从m个离散独立的子空间中采样得到,m个离散子空间中的采样粒子数表示为$B=[b_1,b_2,\cdots,b_m]$.其中:B满足多项式分布$B\sim M(N,P_b),P_b=[p_1,p_2,\cdots,p_m]$. 得到待估计概率为$\hat{P}=B/N$. 将其代入式(10)得到$d(\hat{P},P_b)$. 对于$\hat{P}=B/N$,采样的粒子数量越大,越接近真实的后验分布,因此,利用该参数来衡量满足条件的最低采样规模需求. 以此推得采样规模满足[16]

$$N_{\mathrm{KLD}}=\frac{1}{\tau}\chi^2_{m-1,1-\delta}. \tag{11}$$

为方便计算,利用Wilson-Hilferty转换可得式(11)的近似

$$N_{\mathrm{KLD}}=\frac{m-1}{2\tau}\left\{1-\frac{2}{9(m-1)}+\sqrt{\frac{2}{9(1-m)}}z_{1-\delta}\right\}^3. \tag{12}$$

其中:$z_{1-\delta}$表示标准正态分布的上$1-\delta$分位值;参数采用Dieter Fox给出的先验参数,$\tau=0.15,\delta=0.01$. 由式(12)可以看出,粒子规模与误差界限成反比,符合粒子滤波的特性.

3 KBPF算法

3.1 KBPF算法设计原理

本文算法的核心目标是,在利用蝙蝠算法优化粒子分布和提升粒子多样性的同时,利用KLD采样减弱群体智能算法对计算效率的负面影响,最终实现对算法效率和精度的双重提升.

KLD作为衡量分布之间差距的度量,符合粒子滤波对后验分布估计的应用场景. 可利用KLD作为判据,在重要性采样中,将粒子集过于集中的冗余粒子删除,降低粒子规模,实现计算效率的提升. 在粒子滤波中,每一个粒子都代表部分关于分布的信息,所以KLD采样删除冗余粒子的同时也会删除一部分信息. 由于KLD采样的实质是将过于拥挤的粒子剔除,在粒子滤波中,过于集中的粒子互相之间对分布描述的信息差很小,这里删除的粒子对粒子集描述后验分布的影响很小.

蝙蝠算法是一种模仿蝙蝠追踪目标行为的仿生学算法. 该算法主要模拟蝙蝠利用声呐探测跟踪目标的策略,实现智能寻优. 蝙蝠算法优化粒子滤波时,其粒子的运动由全局寻优和局部寻优两种策略引导,将粒子集中的每个粒子视为一只在空间中寻找最优位置的"蝙蝠". 全局寻优时,每个粒子x_k^i以搜索脉冲频率f_i指导飞行速度v_k^i,调整位置. 全局搜索初期,粒子以较快的速度飞行以提升搜索效率;当粒子接近最优解时,减慢飞行速度以保证搜索精度. 局部寻优时,通过脉冲强度A_k^i和脉冲频率r_k^i进行控制,在全局寻优结果$x_k^{\prime i}$的基础上得到第k步最终的搜索结果x_k^i. 局部搜索初期,采用高脉冲强度和低脉冲频率以保证搜索效率;随着粒子接近"目标",减小脉冲强度,提高脉冲频率,以保证搜索的精度. 整个寻优策略随着粒子集的分布变化自适应调整. 粒子集由于全局寻优的作用,整体向全局最优点移动,又因局部寻优的作用,使得粒子集不过度集中于全局最优点,最终使得整个粒子集的分布更加合理.

本文提出的算法KBPF将KLD采样和蝙蝠算法融合进粒子滤波,并使这两种算法在迭代中相互作用. 第1步,对$k-1$步所得加权粒子集$\{\hat{x}_{k-1}^i,\hat{w}_{k-1}^i\}_N$执行重要性采样,同时利用KLD采样删除过于集中的冗余粒子,得到缩小规模后新生粒子集$\{x_k^i,w_k^i\}_{N_{\mathrm{KLD}}}$. 其中,新生粒子集由于删除了过于集中的粒子,在保留原有分布状态的前提下,粒子间距增大,使得下步蝙蝠算法优化时,获得分布更优的初始粒子集,使算法不易陷入局部最优,同时获得更高的计算效率. 第2步,利用蝙蝠算法优化$\{x_k^i,w_k^i\}_{N_{\mathrm{KLD}}}$的粒子分布,使整个粒子集向最优粒子飞行,最终利用蝙蝠算法优化的粒子集$\{x_k^{*i},w_k^{*i}\}_{N_{\mathrm{KLD}}}$得到对目标状态的估计. 蝙蝠算法优化后,粒子整体分布合理,并向最优粒子集中,因此,在$k+1$步的KLD采样时,将获得更好的性能. 第3步,对$\{x_k^{*i},w_k^{*i}\}_{N_{\mathrm{KLD}}}$进行重采样,采样$N$个粒子,得到$k+1$步的初始粒子集$\{\hat{x}_k^i,\hat{w}_k^i\}_N$. 由于蝙蝠算法优化了KLD采样后的粒子集,粒子向最优点集中,使得重采样后保存的非重复粒子增多,提升了粒子多样性.

综上,本文设计的KBPF算法将KLD采样和蝙蝠算法更强的自适应性与随机性融合进粒子滤波中,并使两种算法在迭代中相互作用,提高粒子多样性,有效压缩粒子规模,达到同时提升算法计算效率和精度的目的.

3.2 KBPF算法设计

KBPF算法分为3部分,具体实现如下.

1) 自适应调整粒子规模.

为防止粒子规模过小,影响计算,预设粒子规模下限为$N_{\min}=30$.

i) 执行重要性采样,通过式(8)和(9)得到第k步迭代加权粒子集$\{x_k^i,w_k^i\}_N$;

ii) 判断粒子x_k^i是否落入空子集b_m中,当粒子x_k^i落入空子集后,该子集状态变为非空,更新非空子集数$m=m+1$;

iii) 设动态粒子规模为n,当$n>N_{\min}$时,将m代

入式(12),更新N_{KLD};

iv) 当$n < N_{\min}$时更新动态粒子规模$n = n + 1$;

v) 当$n = N_{\mathrm{KLD}}$时跳出循环,得到新生粒子集$\{x_k^i, w_k^i\}_{N_{\mathrm{KLD}}}$.

对于KBPF算法,在重要性采样初期,由于空子集较多,大部分新生粒子大概率落入空子集中,m增长较快,使N_{KLD}更新快. 随着空子集变少,m增长放缓. 当$n = N_{\mathrm{KLD}}$时,跳出采样,得到新生粒子集合$\{x_k^i, w_k^i\}_{N_{\mathrm{KLD}}}$,采样规模为$N_{\mathrm{KLD}}$.

2) 全局寻优.

对新生粒子集$\{x_k^i, w_k^i\}_{N_{\mathrm{KLD}}}$进行全局定向寻优. 首先,需预设目标函数作为判断状态点优劣的依据. 粒子滤波中,可通过当前粒子在似然函数中的分布状态来判断其优劣,粒子所处区域似然值越高则越优. 因此,基于粒子滤波的似然分布$p(z_k|x_k)$,得到如下目标函数公式:

$$I = \mathrm{e}^{[-\frac{1}{2R_k}(z_k - \hat{z}_k^i)^2]}. \tag{13}$$

在预设范围$[f_{\min}, f_{\max}]$内随机生成搜索脉冲频率f_i,即

$$f_i = f_{\min} + (f_{\max} - f_{\min}) \times \beta, \tag{14}$$

其中$\beta \in [0, 1]$为均匀分布随机数. 利用目标函数(13)计算$\{x_k^i, w_k^i\}_{N_{\mathrm{KLD}}}$,得到当前粒子集合中的全局最优粒子$x_k^{\mathrm{best}}$. 模仿蝙蝠追踪目标的特性,使粒子向最优粒子飞行. 寻优初期,粒子散布于整个空间中,间距较大,所以采用较快的飞行速度,保证粒子向最优解的收敛速度;当粒子集合逐渐向最优解收敛时,逐渐降低粒子的飞行速度,以保证寻优末期的精度,从而实现自适应的速度调整. 综上,得到飞行速度更新公式

$$v_k^i = v_k^i + (x_k^{\mathrm{best}} - x_k^i) \times f_i. \tag{15}$$

利用自适应飞行速度v_k^i调整当前粒子的位置,使粒子集向最优粒子移动,有

$$x_k^{'i} = x_k^i + v_k^i. \tag{16}$$

3) 局部寻优.

为了防止粒子过度集中,并增加粒子运动的随机性,在全局寻优后,对粒子集进行局部寻优操作.

当$C \leqslant r_k^i$时,直接保存式(16)的新生粒子,即

$$x_k^i = x_k^{'i}. \tag{17}$$

当$C > r_k^i$时,利用脉冲强度在最优解附近随机得到新生粒子

$$x_k^i = x_k^{\mathrm{best}} + \varepsilon A_k^i. \tag{18}$$

其中:C和ε为均匀分布随机数,$C \in [0, 1], \varepsilon \in [-1, 1]$. 为了提升局部搜索能力,模仿蝙蝠的声呐探测机制,在算法中引入脉冲强度A_k^i和脉冲频率r_k^i两个动态参数. 脉冲频率决定了是否接受全局寻优解的概率,脉冲强度决定了当不接受全局寻优解时新粒子的偏移距离. 在粒子向最优粒子逼近的过程中,每只“蝙蝠”逐渐增加发射脉冲的频率,以提升接受寻优结果的概率,即

$$r_k^i = r_k^0[1 - \mathrm{e}^{-\gamma(k-1)}]; \tag{19}$$

同时减小发射脉冲的强度,防止在靠近最优解时产生不必要的过度偏移,即

$$A_k^i = \alpha A_{k-1}^i. \tag{20}$$

其中预设参数$\alpha = \gamma = 0.9$.

4 算法流程

Step 1: 由前步计算得到$k-1$步的加权粒子集$\{x_{k-1}^i, w_{k-1}^i\}_N$,其中$N$为初始粒子数;

Step 2: 开始第k步计算,利用式(8)和(9)更新粒子状态和权值,得到加权粒子集$\{x_k^i, w_k^i\}_N$;

Step 3: 循环执行ii)$\sim$iv),更新N_{KLD};

Step 4: 当$n = N_{\mathrm{KLD}}$时跳出循环,得到新生粒子集$\{x_k^i, w_k^i\}_{N_{\mathrm{KLD}}}$;

Step 5: 全局寻优,顺序执行式(14)$\sim$(16),得到粒子的更新位置和飞行速度$\{x_k^{'i}, v_k^i\}_{N_{\mathrm{KLD}}}$;

Step 6: 局部寻优,顺序执行式(17)$\sim$(20),得到一步寻优最终的粒子集合$\{x_k^i\}_{N_{\mathrm{KLD}}}$;

Step 7: 利用目标函数(13)更新全局最优位置x_k^{best};

Step 8: 循环执行Step5$\sim$Step7,当达到预设寻优代数N_{bat}或x_k^{best}符合精度阈值η时跳出循环,得到新生粒子集合$\{x_k^{*i}\}_{N_{\mathrm{KLD}}}$;

Step 9: 将新生粒子集合$\{x_k^{*i}\}_N$代入式(9)更新权值,对权值采用归一化操作,得到新生加权粒子集$\{x_{0:k}^{*i}, w_{1:k}^{*i}\}_N$;

Step 10: 利用式(3)和(4)加权求和,输出k步的隐状态估计$\hat{x}_k$;

Step 11: 对$\{x_k^{*i}, w_k^{*i}\}_{N_{\mathrm{KLD}}}$执行重采样操作,采样$N$个粒子,得到$k+1$步的初始粒子集$\{\hat{x}_k^i, \hat{w}_k^i\}_N$;

Step 12: 对$k+1$步隐状态进行估计,代入$\{\hat{x}_k^i, \hat{w}_k^i\}_N$,循环执行Step 1$\sim$Step 12.

5 仿真实验

为了验证本文算法的性能,实验采用强非线性单变量非静态增长模型[12]. 其状态转移方程和状态观测方程如下:

状态转移模型

$$x_k = 0.5x_{k-1} + \frac{25x_{k-1}}{1+x_{k-1}^2} + 8\cos(1.2(k-1)) + N_{k-1}; \tag{21}$$

状态观测模型

$$z_k = \frac{x_k^2}{20} + R_k. \tag{22}$$

其中: 设转移噪声为$N_k \sim N(0,10)$, 观测噪声为$R_k \sim N(0,2)$.

为了有效说明本算法的特性, 选取标准粒子滤波SIRPF算法和蝙蝠粒子滤波BAPF算法作为对比.

参数初始化: 隐状态初始位置$x_0 = 0.1$, 初始化粒子集$x_s \sim N(0.3,8)$; 粒子规模$N = 1\,000$, 估计总步数为$T = 75$; KLD先验参数$\tau = 0.15$和$\delta = 0.01$; 全局搜索频率范围$f_{\min} = 0, f_{\max} = 1.7$; 局部寻优参数$\alpha = \gamma = 0.9, A_0 = 0.25, r_0 = 0.5$; 最大寻优代数$N_{\text{bat}} = 20$.

首先, 检验本文算法对粒子多样性的提升作用. 由于KBPF算法的粒子规模动态改变, 采用绝对粒子多样性无法准确实时地衡量其性能, 对此提出新的度量参数: 粒子多样性比例参数N_S. 在粒子滤波的一步估计中, 对估计目标状态的加权粒子集进行系统重采样, 非重复粒子占总粒子数的比例越高, 说明粒子多样性越高, 其中$N_S \in [0,100]$. 对3种算法在各时刻的粒子多样性比例进行仿真, 得到结果如图1所示.

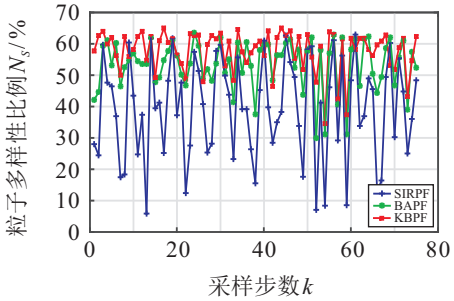

图1　粒子多样性比例

由图1可见: 本文提出的算法, 使得粒子多样性比采用SIRPF算法有明显的提升; 对比BAPF算法, 粒子多样性的比例略有提升. 说明KLD采样虽然使粒子总数变少, 但在动态粒子总数的背景下, KLD采样删除了冗余粒子的操作, 使蝙蝠算法具有更好的优化性能.

然后, 检验本文算法对粒子规模的动态调整能力. 为了验证蝙蝠算法优化粒子集作用于下步迭代的KLD采样, 可提升KLD采样的性能, 引入原始KLD采样粒子滤波作为对照组, 得到粒子规模动态曲线如图2所示.

由图2可见, BAPF和SIRPF粒子规模不变, 由于

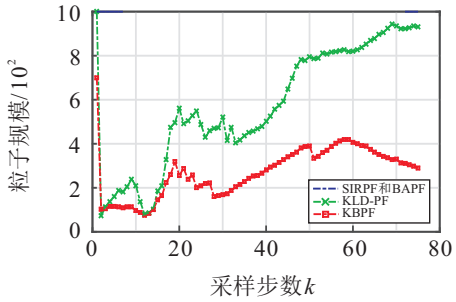

图2　粒子规模

KLD采样删除了冗余粒子, KBPF算法的粒子规模显著减小. 对比KBPF和KLD采样粒子滤波可见, KBPF粒子规模更小, 控制粒子规模的稳定性更好. 由于蝙蝠算法在每步迭代中合理优化粒子分布, 粒子相对集中于最优点, 使每步KLD采样均可有效删除冗余粒子, 稳定控制粒子规模. 该实验结果验证了KBPF可有效缩减粒子规模, 并通过蝙蝠算法优化保持稳定, 达到有效提升计算效率的目的.

最后, 验证本文算法对估计精度的提升效果. 图3为3种算法对目标隐状态的跟踪轨迹, 可见本文算法的跟踪精度最高.

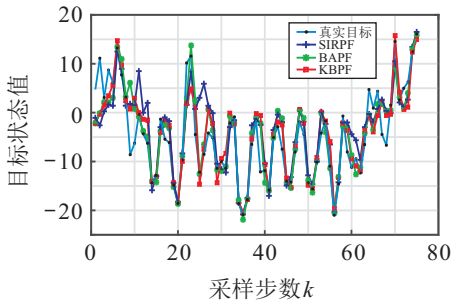

图3　目标状态跟踪轨迹

图4为3种算法估计结果的RMSE, 可以看出, 本文算法的RMSE最低.

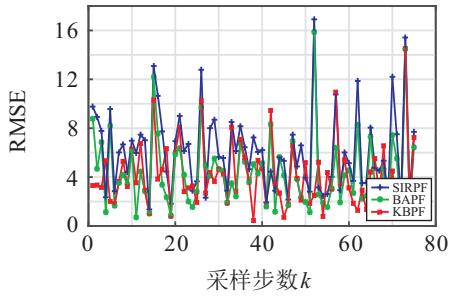

图4　3种算法的RMSE

为了精确反映算法对计算效率和精度的作用, 对3种算法进行500次蒙特卡洛仿真. 得到当粒子规模分别为100、500和1 000时, 算法的RMSE均值和运算时间, 其中运算时间为一次蒙特卡洛仿真中各算法单次迭代的运算时间, 如表1所示(N表示粒子数, 对于本文算法则表示最大粒子数).

表1 实验结果对比

粒子数 N	RMSE			运算时间/s		
	SIRPF	BAPF	KBPF	SIRPF	BAPF	KBPF
100	5.1329	3.5930	3.6187	0.1393	0.1784	0.1672
500	4.4933	3.3946	3.1149	0.3545	0.4598	0.3963
1 000	4.1784	3.1856	2.9863	0.6307	1.0345	0.6521

分析表1可知:在计算精度方面,对比原始SIRPF算法,BAPF和KBPF算法的估计精度均有提升;对比KBPF算法和BAPF算法,当粒子数取较小值100时,本文算法的计算精度与其基本持平,当粒子规模增大至500和1 000时,计算精度优于BAPF. 该结果验证了本文算法可通过KLD采样,进一步提升BAPF的计算精度. 计算效率方面,对比原始SIRPF,由于KBPF和BAPF算法均引入了全局寻优策略,运算时间均有一定程度的增加;对比BAPF,本文算法由于可自适应动态调整粒子数,使得运算时间明显减少,表明该算法可有效提升群体智能优化粒子滤波的计算效率.

6 结 论

本文针对粒子滤波中存在的粒子贫化问题以及采用群体智能优化后带来的计算效率下降问题,提出了一种基于KLD的蝙蝠算法优化自适应粒子滤波算法. 本文算法首先利用KLD采样自适应调整粒子规模,删除冗余粒子,优化蝙蝠算法初始粒子集,再利用蝙蝠算法定向优化调整后的粒子集,提升粒子多样性比例. 由于粒子滤波算法的迭代性,蝙蝠算法使粒子分布更合理且相对集中于最优点,作用于下步迭代的KLD采样,使粒子规模稳定控制在较小的范围,从而达到提升粒子滤波计算精度和计算效率的目的. 由于KBPF算法可自适应地实现粒子间的信息交互并能有效压缩粒子规模,进一步与人工智能结合,在视觉跟踪领域将有更好的发展和应用.

参考文献(References)

[1] Gordon N J, Salmond D J, Smith A F N. Novel approach to nonlinear/non-Gaussian Bayesian state estimation[J]. IEE Proc of Radar and Signal Processing, 1993, 140(2): 107-113.

[2] Li T C, Sun S D, Sattar T P, et al. Fight sample degeneracy and impoverishment in particle filters: A review of intelligent approaches[J]. Expert Systems with Applications, 2014, 41(8): 3944-3954.

[3] Li T C. Resampling methods for particle filtering: Classification implementation and strategies[J]. IEEE Signal Processing Magazine, 2015, 32(3): 72-86.

[4] Fang Z, Tong G F, Xu X H. Particle swarm optimized particle filter[J]. Control and Decision, 2007, 22(3): 273-277.

[5] Qiu X N, Liu S R, Lv Q. Particle filter algorithm based on information-shared mechanism and its application to visual tracking[J]. Control Theory & Applications, 2010, 27(12): 1724-1730.

[6] Klamargias A D, Parsopoulos K E. Particle filtering with particle swarm optimization in systems with multiplicative noise[C]. Proc of the 10th Annual Conf on Genetic and Evolutionary Computation. New York: 2008: 57-62.

[7] Yang X S. Firefly algorithm, stochastic test functions and design optimisation[J]. Int J of Bio-Inspired Computation, 2010, 2(2): 78-84.

[8] Tian M C, Bo Y M, Chen Z M, et al. Firefly algorithm intelligence optimized particle filter[J], Acta Automatica Sinica, 2016, 42(1): 89-97.

[9] Yang X S. A new metaheuristic bat-inspired algorithm[C]. Int Workshop on Nature Inspired Cooperative Strategies for Optimization. Granada, 2010: 65-74.

[10] Yang X S. Bat algorithm: A novel approach for global engineering optimization[J]. Engineering Computations, 2012, 29(5): 464-483.

[11] Yang X S. Bat algorithm: Literature review and applications[J]. Bio-Inspired Computation, 2013, 5(3): 141-149.

[12] Chen Z M. Intelligent particle filter based on bat algorithm[J]. Acta Physica Sinica, 2017, 66(5): (050502-1)-(050502-10).

[13] Doucet A. Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator[J]. Biometrika, 2015, 102(2): 295-313.

[14] Doucet A, Johansen A M. A tutorial on particle filtering and smoothing: Fifteen years later[J]. Handbook of Nonlinear Filtering, 2009, 12(3): 656-704.

[15] Mohammadi A, Asif A. Distributed consensus innovation particle filtering for bearing/range tracking with communication constraints[J]. IEEE Trans on Signal Processing, 2015, 63(3): 620-635.

[16] Fox D. Adapting the sample size in particle filters through KLD-sampling[J]. Int J of Robotics Research, 2003, 22(12): 985-1003.

(责任编辑:李君玲)