# 基于分段RTS平滑的凸组合航迹融合算法

陈金广[1,2] 高新波[1]
(西安电子科技大学电子工程学院影像系统实验室 西安 710071)[1]
(西安工程大学计算机科学学院 西安 710048)[2]

**摘 要** 针对凸组合航迹融合算法在过程噪声不为零的情况下性能下降的问题,引入了RTS平滑算法来提高融合性能。由于传统的RTS平滑算法是得到全部滤波结果之后才执行逆向平滑过程,造成输出延迟,为此,提出了分段RTS平滑算法,一方面可以提高航迹融合性能,另一方面能够保证融合过程中的实时性。在融合过程中,针对局部节点有无额外计算能力的不同情况,结合实施平滑步骤的时机,提出了基于分段RTS平滑的先平滑再融合和先融合再平滑两种改进的凸组合航迹融合方法。这两种方法在不同过程噪声水平下,性能表现都超过凸组合融合算法和最优融合算法。仿真结果表明了新算法的有效性和优越性。

**关键词** 数据融合,跟踪,航迹融合,分段RTS平滑,滤波
**中图法分类号** TP391 **文献标识码** A

## Convex Combination Track-to-track Fusion Algorithms Based on Piecewise RTS Smoothing

CHEN Jin-guang[1,2] GAO Xin-bo[1]
(VIPS Lab, School of Electronic Engineering, Xidian University, Xi'an 710071, China)[1]
(School of Computer Science, Xi'an Polytechnic University, Xi'an 710048, China)[2]

**Abstract** Aimed at the problem that the performance of convex combination track-to-track fusion algorithm degrades in the case that the process noise of dynamic system does not equal zero, RTS smoother was introduced to improve the performance. The traditional RTS smoothing algorithm can be work when all the filtering data are obtained, and the output is delayed largely. Piecewise RTS smoothing algorithm was presented to solve the problem. On the one hand, track-to-track performance can be enhanced; on the other hand, real-time ability can be remained. Furthermore, aimed at the calculation ability of local node owns or not, Smoothing First and Fusing Next (SFFN) and Fusing First and Smoothing Next (FFSN) algorithms based on the piecewise RTS smoothing algorithm were presented according to the chance of implementing the smoothing process. The new methods exceed the performance of the traditional convex combination fusion algorithm and the optimal fusion algorithm in a wide range of process noise. Simulation results show the new algorithms' validity and superiority.

**Keywords** Data fusion, Tracking, Track-to-track fusion, Piecewise RTS smoothing, Filtering

多传感器信息融合系统的结构可以分为集中式和分布式。集中式融合需要把所有局部传感器量测数据都传送到中心节点,网络传输负载大,对中心节点处理能力要求高。在分布式融合系统中,每个传感器均有自己的处理器,在各自节点进行预处理,然后将结果送到中心节点,进行融合处理。由于融合中心的主要任务是对各局部航迹进行融合,因此这种融合结构也称为航迹融合[1]。航迹融合网络负载小,对中心节点处理能力要求不高,所以航迹融合算法历来是人们研究的热点。

最易于实现的航迹融合算法是凸组合融合算法[2],该算法不考虑局部航迹之间的相关性,直接进行线性加权平均,涉及到的参量少,算法简单,运算量小。其它的航迹融合算法还有Bar-Shalom-Campo融合算法、最优分布式估计融合、协方差交叉法以及联邦滤波器融合算法等[1-3]。也有学者采用模糊逻辑的方法进行航迹融合[4,5],并且取得了较好的整体融合性能。但是,这些算法涉及到的参量多,运算量大。文献[6]在不考虑融合实时性的前提下,应用RTS平滑算法,取得了较好的滤波效果。Willsky等人提出了多尺度自回归估计融合算法[8,9],该算法综合了Kalman滤波和RTS平滑算法,提高了多尺度估计融合性能。本文考虑到RTS(Rauch-Tung-Striebel smoother, RTS)平滑算法[7]对噪声的抑制效果要比单纯的Kalman滤波结果好,因此,可以将RTS平滑算法应用到融合过程中,以提高航迹融合性能。

RTS平滑算法的逆向平滑过程是从最后一个滤波结果反向递推向前平滑,也就是说在获得了整个目标航迹跟踪过程之后,才使用RTS平滑算法对系统航迹进行平滑,平滑步骤严重滞后,融合结果的实时性难以得到保证。为了避免输出结果大幅度延迟,本文提出了分段RTS平滑算法。该算法

到稿日期:2009-10-20 返修日期:2009-12-15 本文受国家自然科学基金(60832005,60702061)资助。
陈金广(1977-),男,博士生,讲师,主要研究方向为信息融合、目标跟踪,E-mail:xacjg@163.com;高新波(1972-),男,博士,教授,博士生导师,主要研究方向为影像处理、分析和理解、模式识别和机器学习等。

思想是对需要平滑的时间段内的滤波结果进行分段，然后逐段逆向平滑，由于分段长度可大可小，从而在不同程度上避免融合过程中应用RTS平滑造成的实时性欠佳问题。仿真结果表明，分段平滑算法的分段长度对整个航迹融合结果的误差的影响不大，而且，都比传统航迹融合算法效果要好。

将分段RTS平滑过程应用到凸组合航迹融合过程中，能够较好地克服由于过程噪声使得凸组合航迹融合算法性能降低的问题。在融合过程中，针对局部传感器有无额外计算能力，结合实施分段平滑过程的时机，本文提出了基于分段RTS平滑的先平滑再融合和先融合再平滑两种改进的凸组合航迹融合方法。仿真结果表明，这两种方法航迹融合性能高于凸组合航迹融合算法和最优航迹融合算法。

对于应用于非线性情形的扩展Kalman滤波器、Unscented Kalman滤波器和粒子滤波器，文献[10-12]也提出了相应的平滑算法，从而也可以套用本文思路针对多传感器非线性系统采用各种非线性平滑算法进行航迹融合。

## 1 RTS平滑算法与航迹融合算法

平滑器是用在滤波过程之后的处理，利用全部过去、现在和未来的观测量来计算出一个理想的估计值[13]。平滑问题比预测和滤波问题复杂，但是估计精度高。由于平滑器算法要根据滤波结果运算，因此，好的滤波结果才会有好的平滑结果。

### 1.1 RTS平滑算法

RTS平滑(Rauch-Tung-Striebel smoother, RTS)[7]属于固定区间平滑，是最简单易用的一类固定区间平滑器。

考虑如下线性系统

$$\begin{cases} x_k = F_{k,k-1}x_{k-1} + v_{k-1}, k=1,2,\cdots,N \\ z_k = H_k x_k + w_k \end{cases} \tag{1}$$

其中，$x_k$ 为系统状态向量；$z_k$ 为观测向量；$v_k$ 为过程噪声；$w_k$ 为量测噪声；$v_k$ 和 $w_k$ 均为零均值高斯白噪声，方差分别为 $Q$ 和 $R$。初始状态 $x_0$ 和初始状态协方差 $P_0$ 已知，初始状态、过程噪声、量测噪声三者互不相关。

RTS平滑算法包括两个过程，正向滤波和逆向平滑。对式(1)描述的线性动态系统，正向滤波过程可以采用标准Kalman滤波算法，滤波公式为

$$\begin{cases} \hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - H_k \hat{x}_{k|k-1}) \\ P_{k|k} = (I - K_k H_k)P_{k|k-1} \end{cases} \tag{2}$$

其中，$K_k$ 为Kalman增益。

对于逆向平滑过程，平滑公式为

$$\begin{cases} \hat{x}^s_{k|N} = \hat{x}_{k|k} + K^s_k(\hat{x}^s_{k+1|N} - x_{k+1|k}) \\ P^s_{k|N} = P_{k|k} + K^s_k(P^s_{k+1|N} - P_{k+1|k})(K^s_k)^{\mathrm{T}} \end{cases} \tag{3}$$

其中，$k=0,1,2,\cdots,N$，$\hat{x}^s_{k|N}$ 和 $P^s_{k|N}$ 分别是RTS平滑算法平滑后的状态向量及其协方差；$\hat{x}_{k|k}$ 和 $P_{k|k}$ 是Kalman滤波后的估计值及其协方差；$K^s_k$ 是平滑增益，公式为

$$K^s_k = P_{k|k}F_{k+1,k}P^{-1}_{k+1|k} \tag{4}$$

$x_{k+1|k}$ 和 $P_{k+1|k}$ 是一步预测值及其协方差。

从逆向平滑过程可以看出，RTS平滑利用滤波结果进行计算，平滑效果依赖于前向滤波效果。而且逆向平滑是从最后一个滤波结果开始逐个向前进行平滑，因此，要对时间段[0 $N$]内的观测值进行平滑，必须首先得到 $N$ 个滤波结果。如果 $N$ 比较大，逆向平滑过程就会严重滞后，系统反应迟钝，从而限制了RTS平滑算法在实时性要求比较高的航迹融合过程中的应用。

### 1.2 凸组合航迹融合

凸组合航迹融合算法(Convex Combination track-to-track Fusion, CCF)[2]不考虑各传感器局部估计误差之间的相关性。当局部航迹都是传感器航迹并且不存在过程噪声，各传感器在初始时刻的估计误差也不相关时，凸组合融合算法是最优的。然而该算法运算量小，需要传送到中心节点的信息量就少。假设 $M$ 个传感器的估计值及其协方差为 $\{\hat{x}^i, P^i\}_{i=1}^M$，则融合方程为

$$\hat{x} = (\sum_{i=1}^M (P^i)^{-1})^{-1} \sum_{i=1}^M (P^i)^{-1}\hat{x}^i, P^{-1} = \sum_{i=1}^M (P^i)^{-1} \tag{5}$$

### 1.3 最优航迹融合

最优航迹融合算法(Optimal track-to-track Fusion, OF)[3]考虑了局部估计误差之间的相关性，对航迹的过程噪声有较好的抑制性。该算法的性能和集中式扩维融合算法性能相同，所以可以作为衡量分布式航迹融合算法性能的基准算法。该算法的融合方程为

$$\begin{cases} \hat{x} = P((P^i)^{-1}\hat{x}^i + (P^j)^{-1}\hat{x}^j - (\bar{P}^i)^{-1}\bar{x}^i - (\bar{P}^j)^{-1}\bar{x}^j + (\bar{P})^{-1}\bar{x}) \\ P = ((P^i)^{-1} + (P^j)^{-1} - (\bar{P}^i)^{-1} - (\bar{P}^j)^{-1} + \bar{P}^{-1})^{-1} \end{cases} \tag{6}$$

其中，$(\bar{x}^i, \bar{P}^i)$ 和 $(\bar{x}^j, \bar{P}^j)$ 分别表示传感器 $i, j$ 的一步预测值及其协方差；$(\bar{x}, \bar{P})$ 表示融合后的一步预测值及其协方差。

由于最优航迹融合算法已经考虑了过程噪声等因素对融合结果的影响，因此，如果将分段平滑算法应用到最优航迹融合算法过程中，融合性能不但没有提高，反而变差，这种现象是在实验过程中发现的。但是在凸组合算法的基础上实施分段平滑，就能取得较好的实验效果。

## 2 基于分段RTS平滑的凸组合航迹融合方法

### 2.1 分段RTS平滑算法

假设某一动态系统的总时长为 $k=1,2,\cdots,N$。分段RTS平滑算法就是对 $N$ 个观测值分段进行前向滤波和逆向平滑，假设分段长度为 $L$，且 $1 \leqslant L \leqslant N$，则分段RTS平滑算法步骤如下。

Step 1 输出变量清空 $x_s=[]$，$P_s=[]$；

Step 2 局部滤波结果变量清空 $M=[]$，$P=[]$；分段平滑计数器清零 $count=0$；

Step 3 如果对目标的跟踪没有结束，则继续向下处理，否则输出结果 $x_s$，$P_s$，整个分段RTS平滑算法结束；

Step 4 接收当前滤波结果并保存在变量 $M$，$P$ 中，并采用计数器 $count$ 计数。如果 $count==L$，针对局部滤波结果变量 $M$，$P$ 采用传统RTS平滑算法进行逆平滑过程，平滑结果保存在输出变量 $x_s$，$P_s$ 中；转到Step 2。

算法中，$\hat{x}$，$P$ 既可以是局部节点的滤波状态及方差阵，也可以是中央节点对各个局部节点滤波结果融合之后的状态值及方差阵；$x_s$ 和 $P_s$ 表示分段RTS平滑之后的状态值及方差阵。

如果定义分段RTS平滑算法的响应时间为从数据采样开始到获得第一个平滑结果为止的时间，则分段RTS平滑算法的响应时间为

$$t_r=(t_{sampling}+t_{filtering}+t_{smoothing})L \tag{7}$$

其中，$t_{sampling}$ 表示平均采样时间；$t_{filtering}$ 表示平均滤波时间；$t_{smoothing}$ 表示平均逆向平滑时间；$L$ 表示 RTS 平滑算法分段的长度。当 $L\geqslant N$ 时，分段 RTS 平滑算法退化为传统 RTS 平滑算法；当 $L<N$ 时，分段 RTS 平滑算法的响应时间比不分段的 RTS 平滑算法的响应时间小。

取得 $L$ 个滤波数据后，就立刻进行逆向平滑过程，不需要等待后续数据的到来，从而大大降低了逆向平滑的滞后时间。分段长度 $L$ 取值越小，融合算法的实时性就越好。

### 2.2 基于分段 RTS 平滑的凸组合航迹融合方法

在凸组合航迹融合过程中应用分段 RTS 平滑算法，针对局部节点有无额外计算能力，结合实施逆向平滑过程的时机，将其分为两种：(1)先平滑再融合(Smoothing First and Fusing Next，SFFN)。也就是先对局部航迹分段 RTS 逆向平滑，然后再利用凸组合算法进行融合。(2)先融合再平滑(Fusing First and Smoothing Next，FFSN)。也就是先采用凸组合算法融合局部航迹，再对融合后的航迹进行分段 RTS 逆向平滑。这两种方法的融合过程如图 1(a)和图 1(b)所示。在工程实践过程中，如果局部节点有额外的计算能力，就将逆向平滑过程放在局部节点处理，即采用 SFFN；如果局部节点没有额外的计算能力，就将逆向平滑过程放在中心节点处理，即采用 FFSN。由于在中心节点实施逆向平滑过程使用的数据是融合后的，经过的处理比在局部节点多，因此理论上来讲，SFFN 比 FFSN 效果好。仿真结果与这一结论一致。

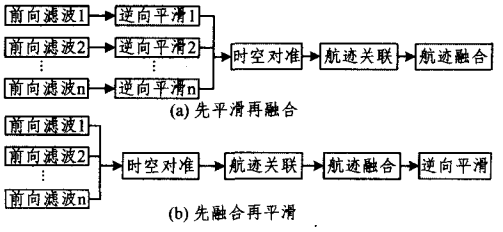


图 1 基于 RTS 平滑的航迹融合过程

假定各个局部节点数据采样是同步的，并且忽略数据在分布式传感器网络中的传输时间，那么航迹融合系统的响应时间可以采用式(8)估算。

$$\begin{cases} t_{r1}=(t_{sampling}+t_{filtering}+t_{smoothing})L+t_{fusion} \\ t_{r2}=(t_{sampling}+t_{filtering}+t_{fusion}+t_{smoothing})L \\ t_{r3}=(t_{sampling}+t_{filtering}+t_{fusion}) \end{cases} \tag{8}$$

其中，$t_{r1}$，$t_{r2}$，$t_{r3}$ 分别表示 SFFN，FFSN 和 CCF 3 种算法的响应时间；$t_{sampling}$ 表示平均采样时间；$t_{filtering}$ 表示平均滤波时间；$t_{fusion}$ 表示平均融合时间；$t_{smoothing}$ 表示平均逆向平滑时间；$L$ 表示 RTS 平滑算法分段的长度。当 $L\geqslant N$ 时，分段 RTS 平滑算法退化为常规 RTS 平滑算法，故此时前两种算法响应时间性能并未得到改善；当 $1\leqslant L<<N$ 时，尤其是当 $L<<N$，分段 RTS 算法响应时间大大缩短。由以上分析可知，当分段长度选取得比较小时，分段平滑算法可以有效地缩短融合结果响应时间，从而确保航迹融合的实时性。

在使用 SFFN 和 FFSN 算法过程中，如果分段 RTS 平滑算法分段长度过长，则融合结果滞后，无法满足融合实时性要求；如果分段长度过短，则融合性能有可能达不到最优。这就需要在实时性和精确性之间做折中。由于最优分段长度难于提前估算，因此使用过程中只能凭借经验选取合适的分段长度。本文第 1 组实验和第 3 组实验选取的分段长度值为 1，试验中发现，虽然所选取的分段长度并不是最优的，但是同样能够大幅度提高航迹融合性能。

## 3 仿真实验及结果分析

状态变量为 $x=[\xi\ \ \dot{\xi}]^T$，其离散状态方程和量测方程为

$$\begin{cases} x_{k+1}=\begin{bmatrix}1 & T\\ 0 & 1\end{bmatrix}x_k+\begin{bmatrix}T^2/2\\ T\end{bmatrix}v_k \quad k=0,1,\cdots,99 \\ z_k^i=[1\quad 0]x_k+w_k^i \quad k=1,\cdots,100, \end{cases} \tag{9}$$

其中，$i=1,2$ 表示传感器 1 和传感器 2；过程噪声 $Q$ 和量测噪声 $R^i$ 是相互独立的零均值高斯白噪声；并且过程噪声、量测噪声和状态初始值互不相关。实验中 $x_0=[0\quad 10]^T$，$P_0=[10\quad 1;1\quad 10]$，$R^1=5$，$R^2=15$，$T=1$，仿真步数 $N=100$ 步。

实验 1(基于分段 RTS 的凸组合航迹融合算法性能分析)

采用分段长度 $L=1$、过程噪声 $Q=10$ 的基于分段 RTS 平滑的凸组合融合算法进行 1000 次 Monte Carlo 仿真实验，对比 SFFN，FFSM，CCF 和 OF 4 种融合算法的均方根误差(Root Mean Square Error，RMSE)。

实验结果如图 2(a)和图 2(b)所示。结果表明，在融合过程中应用分段长度为 1 的 RTS 平滑过程之后，融合效果改善显著，状态向量两个分量的误差显著降低。所提出的两种方法比 CCF 和 OF 算法的性能都要好。

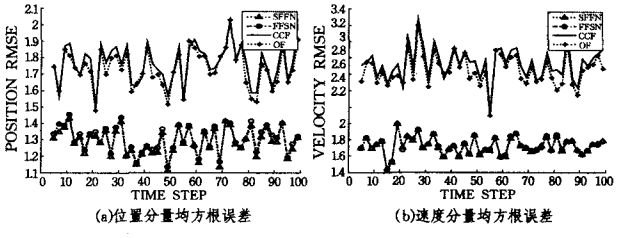


图 2 融合算法性能对比

实验 2(分段 RTS 平滑算法分段长度对本文算法影响分析)

过程噪声 $Q=10$，依次改变 RTS 平滑算法的平滑尺寸 $L=1,2,3,\cdots,N$，分别采用不同的分段长度 Monte Carlo 仿真 100 次，计算不同平滑算法尺寸仿真结果的均方根误差。当使用平滑尺寸参数 $k$ 进行仿真时，计算第 $k$ 个均方根误差如下

$$\sqrt{\frac{1}{Runs*N}\sum_{i=1}^{Runs}\sum_{j=1}^{N}(\hat{x}_j^i-x_j^i)(\hat{x}_j^i-x_j^i)^T} \tag{10}$$

实验结果如图 3(a)和图 3(b)所示，横坐标表示分段的长度。实验结果表明，分段 RTS 平滑算法的分段长度对融合结果影响不大，随着分段长度的增加，融合误差趋向常数。而且，不管对分段 RTS 平滑算法的分段长度如何选择，基于分段 RTS 航迹融合方法的效果总是优于 CCF 和 OF。

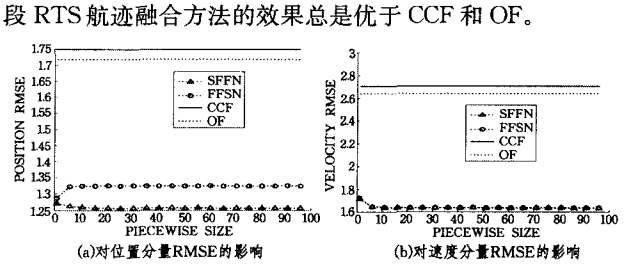


图 3 分段 RTS 平滑算法分段长度对本文算法的影响

实验 3(过程噪声水平对本文算法影响分析)

设定 RTS 平滑算法的平滑尺寸 $L=1$，采用不同数量级的过程噪声分别进行 300 次 Monte Carlo 仿真实验，过程噪

声数值依次设定为 $Q=10^{-4},10^{-3},10^{-2},10^{-1},10^{0},10^{1},10^{2},10^{3},10^{4}$。对比 SFFN,FFSN,CCF 和 OF 4 种融合算法在不同过程噪声水平下的均方根误差。当给定一个过程噪声参数值时,其仿真结果的均方根误差定义与式(10)相同。

实验结果如图 4(a)和图 4(b)所示,当过程噪声范围设定为 $Q<10^{4}$ 时,本文算法融合性能优于 CCF 和 OF,当过程噪声水平高于 $10^{4}$ 时,本文算法性能开始退化。也就是说,当过程噪声数量级别和各个局部传感器量测噪声级别相差 1000 倍以上时,新算法性能开始退化。但是此时过程噪声相对于量测噪声来说,其取值级别已经不再具有应用价值。

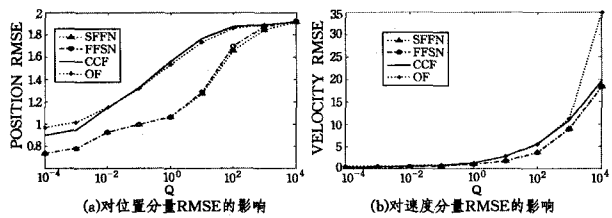


图 4 本文算法对过程噪声 Q 敏感程度分析

**结束语** 在航迹融合过程中,采用分段 RTS 平滑过程的凸组合航迹融合算法能够明显地改善融合性能,且能够兼顾融合系统的实时性。本文的贡献在于:提出了分段 RTS 算法,并将其应用在航迹融合过程中;针对局部节点有无额外计算能力,提出了两种基于分段 RTS 平滑的凸组合航迹融合方法来应对不同的情况,从而改善了融合效果。

本文采用的 RTS 算法是经典的基于 Kalman 滤波条件下的平滑算法。实际上,新的滤波算法和相应的 RTS 平滑算法也具有相同特征。这些平滑算法包括 UKF RTS 平滑算法以及粒子滤波 RTS 平滑算法等。对新型滤波平滑算法在航迹融合过程中的应用值得进一步研究。

## 参 考 文 献


[1] 韩崇昭,朱洪艳,段战胜. 多源信息融合[M]. 北京:清华大学出版社,2006

[2] Chong C Y, Mori S, Chang K C, et al. Architectures and algorithms for track association and fusion [J]. IEEE Transactions on Aerospace and Electronic Systems, 2000, 15(1): 5-13

[3] Chang K C, Saha R K, Bar-Shalom Y. On optimal track-to-track algorithm [J]. IEEE Transactions on Aerospace and Electronic Systems, 1997, 33(4): 1271-1276

[4] Chen Xiaohui, Zhang Yang. A fuzzy fusion in multitarget tracking with multisensor [C]// International Conference on Computational Intelligence and Security. Harbin, China, 2007: 152-156

[5] Tafti A D, Sadati N. Novel adaptive Kalman filtering and fuzzy track fusion approach for real time applications [C]// 3rd IEEE Conference on Industrial Electronics and Applications. Singapore, 2008: 120-125

[6] 潘泉,张磊,崔培玲,等. 动态多尺度系统估计理论与应用[M]. 北京:科学出版社,2007

[7] 杨艳娟,金志华,田蔚风,等. RTS 平滑算法在捷联惯性导航系统初始对准精度事后评估中的应用 [J]. 上海交通大学学报,2004,38(10):1743-1747

[8] Chou K C, Willsky A S, Benveniste A. Multiscale recursive estimation, data fusion, and regularization [J]. IEEE Transactions on Automatic Control, 1994, 39(3): 464-478

[9] Sarkka S. Unscented Rauch-Tung-Striebel smoother [J]. IEEE Transactions on Automatic Control, 2008, 53(3): 845-849

[10] Rauch H E, Tung F, Striebel C T. Maximum likelihood estimates of linear dynamic systems [J]. Journal of American Institute of Aeronautics and Astronautics, 1965, 3(8): 1445-1450

[11] Godsill S J, Doucet A, West M. Monte Carlo smoothing for nonlinear time series [J]. Journal of the American Statistical Association, 2004, 99(465): 156-168

[12] Klaas M, Briers M, de Freitas N, et al. Fast particle smoothing: if I had a million particles [C]// Proceedings of ICML Pittsburgh. PA, 2006: 481-488

[13] Bar-Shalom Y, Li X R, Kirubarajan T. Estimation with Applications to Tracking and Navigation [M]. New York: Wiley-Interscience, 2001


(上接第 145 页)

编码错误,提高编程人员的效率。

当然,我们只是做了一些初步的工作,该模型在许多方面还有待于进一步完善。首先,对集群节点间的任务划分和分配模式尚需要进一步研究,以便实现让任务的划分自动化和动态化,从而最大程度地减轻程序设计人员的负担,并使系统达到动态负载平衡;其次,更多的设计模式也需要被逐步地发掘,同时需要对它们做进一步的调试和优化,以使现有的设计模式模板库更加丰富,让用户具有更大的选择空间和获得更好的系统性能。

## 参 考 文 献


[1] Dagum L, Menon R. OpenMP: an industry standard API for shared-memory programming [J]. IEEE Computational Science & Engineering, 1998, 5(1): 46-55

[2] Gropp W, Lusk E, Skjellum A. Using MPI: portable parallel programming with the message-passing interface[M]. 2rd ed. Cambridge, Mass: MIT Press, 1999

[3] 王文义,董绍静. 基于并行程序效率和通用性的实践与研究[J]. 计算机科学,2009,36(6):290-293

[4] Newton P, Browne J C. The CODE 2.0 graphical parallel programming language[C]// Proceedings of the 6th ACM International Conference on Supercomputing. New York: ACM, 1992: 167-177

[5] Szafron D, Schaeffer J. An experiment to measure the usability of parallel programming systems[J]. Concurrency: Practice and Experience, 1996, 8(2): 146-166

[6] Macdonald S, Anvik J, Bromling S, et al. From patterns to frameworks to parallel programs[J]. Parallel Computing, 2002, 28(12): 1663-1683

[7] Reinders J. Intel Threading Building Blocks [M]. Sebastopol CA: O' Reilly, 2007

[8] Wu Huabei. Design-pattern based parallel programming model and system implementation[C]// Proceedings of the 4th International Conference on Wireless Communications, Networking and Mobile Computing. Piscataway: IEEE, 2008, 11(3): 1-5

[9] Yu Ce, Sun Jizhou, Wu Huabei, et al. EasyPAB: An Extensible IDE Framework for Parallel Applications[C]// Proceedings of Advanced Parallel Process Technologies. LNCS4847 Germany Heidelberg: Springer, 2007: 666-673