GPU-Accelerated Particle Filter Algorithm for Multi-Speaker Tracking and Its Applications (Postprint)

Authors: Cao Jie, Huang Kaijie, Wang Jinhua

Date: 2018-05-02T00:00:00+00:00

Abstract

To address the issue of particle divergence in particle filter-based multi-speaker tracking, which results in low multi-target tracking accuracy, a multi-source localization method based on parallel particle filtering and GPU-based K-means clustering is proposed. The method first analyzes that in the implementation of multi-target tracking using particle filters, the data association process incurs significant computational overhead, and particles tend to diverge when multiple targets are present. To address the problems of high computational complexity and particle divergence, a method combining parallel particle filtering with K-means clustering is proposed. Experimental results indicate that as the number of particles and targets increases, the computational complexity grows exponentially and particle divergence becomes severe. The particle filter-based multi-speaker tracking method utilizing GPU-based K-means clustering demonstrates a more convergent particle set and higher tracking accuracy compared to conventional particle filter tracking methods.

Full Text

Preamble

Title: GPU-Accelerated Particle Filter Multi-Speaker Tracking Algorithm and Its Application

Authors: Cao Jiea,b, Huang Kaijieb, Wang Jinhuab

Affiliations: aCollege of Computer & Communication; bCollege of Electrical & Information Engineering, Lanzhou University of Technology, Lanzhou 730050, China

Abstract: To address the problem of low multi-target tracking accuracy caused by particle divergence in particle filter-based multi-speaker tracking, this paper

proposes a multi-source localization method combining parallel particle filtering with GPU-based K-means clustering. The method first analyzes the data association process in particle filter multi-target tracking, which generates substantial computational load and suffers from gradual particle divergence as multiple targets emerge. To tackle these issues of high computational complexity and particle divergence, a parallel particle filter and K-means clustering approach is proposed. Experiments demonstrate that as the number of particles and targets increases, computational load grows exponentially and particle scattering becomes severe. The proposed GPU-based K-means clustering particle filter multi-speaker tracking method achieves more convergent particle sets and higher tracking accuracy compared to traditional particle filter tracking methods.

Keywords: GPU; particle filter; K-means; multi-target tracking

0 Introduction

Particle filtering methods [**?**] are suitable for non-Gaussian, non-linear system models, which better approximate real-world scenarios. By using time delay estimation as observations, the method directly obtains sound source positions after filtering. The speaker motion process [**?**, **?**] exhibits strong randomness, and appropriate models are required to describe this motion. Researchers have proposed using dynamic space models to characterize speaker movement and applying filtering theory [**?**, **?**] to solve speaker tracking problems. Dynamic space models provide prior information about speaker motion patterns, and leveraging such prior information helps improve speaker localization accuracy.

In speaker localization and tracking, Kalman filtering methods [**?**, **?**] address system state estimation under Gaussian linear models; Extended Kalman filtering methods [**?**, **?**] perform non-linear filtering using time delay estimation as observations to solve state estimation under Gaussian non-linear models; Unscented Kalman filtering methods [**?**] address state estimation under Gaussian non-linear models. Since real-world scenarios are typically non-linear and non-Gaussian, particle filtering is most widely applied in speaker localization and tracking.

Subsequent research has targeted improvements to particle filtering for speaker tracking along three directions: first, extraction of localization features [**?**]; second, algorithmic improvements to particle filtering itself [**?**], addressing limitations and applying them to speaker tracking; and third, adoption of more appropriate dynamic space models to describe multi-speaker motion processes. However, traditional particle filtering for multi-target tracking employs multiple particle filters, with each filter tracking one target and using data association methods to match different measurements to corresponding targets. This approach makes each target tracking process independent, but the association process generates substantial computational load, and particle filtering suffers from particle divergence and low tracking accuracy when tracking multiple targets.

To address the computational complexity issue, this paper introduces parallel strategies to implement particle filtering in parallel. Furthermore, to tackle particle divergence that occurs when multiple targets appear in the tracking area, this paper employs a GPU-based K-means clustering method [**?**] to solve this problem. Experimental results demonstrate that the proposed algorithm effectively improves hardware resource utilization in GPUs, achieves stronger particle convergence, and delivers higher tracking accuracy in multi-target tracking compared to the original particle filter tracking algorithm.

1 Speaker Motion Model

The multi-speaker motion model represents position information changes of moving speakers in smart meeting room environments, with the Langevin model being widely adopted. This model integrates characteristics of various random motions and can effectively describe speaker movement in sound source localization and tracking problems.

The sound source motion equation in the X-axis direction is:

$$x_t = x_{t-1} + \dot{x}_{t-1}T + \frac{1}{2}a_{xT}^2 \tag{1}$$

$$\dot{x}_t = \alpha\dot{x}_{t-1} + \beta_x v_{t-1} \tag{2}$$

where x_t and $\dot{x}_t$ represent position and velocity in the X-axis direction, respectively; y_t and $\dot{y}_t$ represent position and velocity in the Y-axis direction; v_{t-1} is a normally distributed random variable; T is the time interval between frames; L_s is the frame length; f_s is the sampling frequency; α is a constant; and β is the steady-state root-mean-square velocity, where $\beta = 10^{-1}$ in this paper.

2 Multi-Target Model

Based on changes in target motion state and quantity, target motion models include four types: target movement, target disappearance, target generation, and target derivation, as shown in Figure 1 [Figure 1: see original paper].

2.1 Multi-Speaker Tracking Method Based on Particle Filter and K-Means Clustering

Traditional particle filtering for multi-target tracking employs multiple particle filters, with each filter tracking one target. Data association methods match different measurements to corresponding targets to obtain each target' s features, making each target tracking process independent. However, the association process generates substantial computational load, and traditional particle filtering suffers from particle divergence as the number of targets increases. To address particle divergence and computational complexity, this paper introduces parallel strategies to implement particle filtering and K-means clustering algorithms

in parallel, thereby reducing computational complexity and improving tracking accuracy.

2.2 GPU-Based K-Means Clustering Algorithm

The core idea of the K-means clustering algorithm is to first select k objects from n given objects as initial cluster centers, then calculate the similarity between remaining objects and these k centers, and subsequently recompute the cluster centers. The basic algorithm flow is as follows:

a) Randomly select k objects from particle set X as initial cluster centers $c_1, c_2, ..., c_k$.

b) Assign each object x_i $(i = 1, 2, 3, ..., n)$ to the nearest cluster center based on distance. The distance between x_i and cluster center c_j is defined as $||x_i - c_j||$.

c) Recalculate new cluster centers for each cluster:

$$c_j = \frac{1}{N_j} \sum_{x \in S_j} x$$

where N_j is the number of objects in cluster S_j.

d) Repeat steps b) and c) until the k cluster centers no longer change, i.e., the criterion function converges.

The main idea of GPU-based K-means is data parallelism, where computationally intensive portions of the traditional K-means clustering algorithm are executed on the GPU device to achieve performance improvement. The data object assignment and K-means center point calculation are performed multiple times, with different data being independent. These operations can be separated into two kernel functions composed of numerous threads, invoked on the host side and executed on the device side. During task allocation, the host is responsible for placing k objects into the space represented by the objects to be clustered, rearranging all data objects, and controlling the iteration process, while the device side performs data-parallel intensive computations. After data object assignment, cluster labels for each data point are obtained for cluster center calculation. This paper downloads cluster labels from the device to the host, where the host rearranges all data and calculates the number of data objects contained in each cluster.

Algorithm 1: GPU-Based K-Means Clustering Algorithm

```
1 if threadId=0 then
2   Initialize cluster centers
3 end if
4 Synchronize threads
5 repeat
6   for all data points do
```

```
7     Assign data points to nearest cluster
8   end for
9   Synchronize threads
10  if threadId=0 then
11    for all clusters do
12      Update cluster centers
13    end for
14    for all clusters do
15      Check convergence
16    end for
17    if convergence then
18      signal threads to terminate
19    end if
20  end if
21 until convergence
```

The GPU-based parallel K-means clustering algorithm flow is shown in Figure 2 [Figure 2: see original paper]. During task allocation, the host places k objects into the space represented by the objects to be clustered, rearranges all data objects, and controls the iteration process, while the device performs data-parallel intensive computations. After data object assignment, cluster labels for each data point are obtained for cluster center calculation. The simplest approach would be to read all data objects and determine cluster centers for data points. However, extensive conditional statements are unsuitable for the GPU processor model. Therefore, this paper adds a process that involves downloading cluster labels from the device to the host, where the host rearranges all data and calculates the number of data objects contained in each cluster.

For data storage, data objects and center points are stored on the device side using dynamic arrays. This paper stores all parameters of the K-means clustering algorithm in global memory because constant memory and texture memory are read-only and have only 64KB capacity, which is insufficient to accommodate all K-means clustering parameters.

Since the GPU is a shared-memory processor architecture where processors are called threads and follow a master-slave model, each thread is limited between 0 and $t-1$, where t is the total number of threads. Typically, thread 0 is considered the master thread, and all other threads are slave threads (though any thread can be designated as master). Threads share a portion of memory containing the data point set, the current center point set, and compute the distance matrix A between all internal centers and sort matrix A to obtain the index permutation matrix B. Each thread also has three additional memory spaces for internal data use, assuming closed architecture access is feasible.

The mapping to GPU programming is described as follows: First, the master thread initializes all cluster centers as in the serial algorithm. Then the data is partitioned into subsets. This is merely the assignment stage, where each thread performs processing for its assigned data partition. The assignment for

each data point x_i is stored in an m-dimensional vector group. This eliminates simultaneous writes during cluster updates and simplifies the recording process.

After the assignment stage, synchronization between threads is required to ensure that cluster center update stage data is ready for use. The center point update stage can then be executed through reduction operations. However, for simplicity, this paper assumes the master thread executes this stage sequentially. The master thread iterates not through all center points but through partial computations of new center points for all assignments. An m-dimensional vector C is updated in each iteration, with each element c_j storing the number of data points assigned to cluster j. The next loop iterates through all center points, processing C to obtain the final center points.

Convergence determination is also made by the master thread, which checks whether the last stage has changed in the clustering algorithm to determine convergence. Once convergence conditions are met, slave threads are stalled by the master thread.

2.3 Localization Function

This paper utilizes a circular microphone array combined with a steerable power response method to extract multi-speaker azimuth angles during localization feature extraction:

$$\theta = \arg\max_{\omega,\tau} \frac{|\sum_{i=1}^{M}\sum_{j=1}^{M} R_{ij}(\tau)e^{-j\omega(\theta_i-\theta_j)}|}{\sum_{i=1}^{M}\sum_{j=1}^{M} R_{ij}(0)}$$

By placing two circular microphone arrays and using them to extract multi-speaker localization features (Figure 3 [Figure 3: see original paper]), where the coordinate origin O is located at the midpoint between the two circular arrays with center coordinates O_1 and O_2, and the sound source position is $S(x_s, y_s)$, the azimuth angles of the sound source relative to the two circular arrays are:

$$\theta_1 = \arccos\left(\frac{(O_1 - S)\cdot(x_s - O_1)}{\|O_1 - S\|\cdot\|x_s - O_1\|}\right)$$

$$\theta_2 = \arccos\left(\frac{(O_2 - S)\cdot(x_s - O_2)}{\|O_2 - S\|\cdot\|x_s - O_2\|}\right)$$

2.4 Multi-Speaker Tracking Algorithm Based on Particle Filter

This section employs a parallel particle filter algorithm to track multi-speaker states, using the Langevin model as the motion model and a Gaussian likelihood function. The steps for combining parallel particle filtering with parallel K-means clustering to achieve multi-source localization and tracking are as follows:

a) Sample particles $\{x_0^i, i = 1, ..., N\}$ from the initial distribution, with corresponding weights $w_0^i = 1/N$.

b) Predict the latest time particle position distribution according to the Langevin motion model.

c) Estimate each particle' s azimuth angle to compute the likelihood function. Azimuth angles are calculated using equations (6)-(7), then the corresponding likelihood function is computed.

d) Use the likelihood function results as particle weights and normalize the weights:

$$w_k^i = \frac{w_k^i}{\sum_{i=1}^{N} w_k^i}$$

e) Perform resampling using the normalized weights.

f) For multi-speaker tracking, use parallel K-means clustering to determine which particles originate from the same tracking target. The number of speakers is known during parallel K-means clustering. After clustering, the position estimate for speaker p is:

$$\hat{x}_p = \sum_{i=1}^{N_p} w_k^i x_k^i$$

where $\hat{x}_p$ is the position estimate for speaker p, x_k^i is the i-th particle corresponding to speaker p, and N_p is the number of particles.

3 Experimental Results and Analysis

The experimental platform uses a desktop computer with GTX960 graphics card and Visual Studio 2013 software. Hardware parameters are shown in Table 2 .

Table 2: Experimental Hardware Parameters

Component	Specification
GPU	GTX960
CPU	i5-4460
Frequency	3.2 GHz

The purpose of introducing parallel K-means clustering in particle filter-based multi-source tracking is to perform cluster analysis on resampled particles, grouping particles belonging to the same target. Figure 5 [Figure 5: see original paper] illustrates the clustering process of the parallel K-means clustering algorithm.

Figure 5 shows particle distribution changes at different stages during parallel execution: (a1) particle distribution before resampling, (b1) particle distribution after resampling, (c1) results after K-means clustering; (a2) particle distribution before resampling, (b2) particle distribution after resampling, (c2) results after parallel K-means clustering.

As shown in Figure 5, particles are relatively dispersed before resampling. After resampling, low-weight particles are replaced by high-weight particles, resulting in more concentrated distribution. Although the particle distribution clearly reveals each particle' s target 归属, the particle filter algorithm itself cannot classify these particles. Figure 5(c) demonstrates the classification results from parallel K-means clustering, where particles belonging to different targets are clearly distinguished. Compared with standard K-means clustering, parallel K-means clustering exhibits more distinct differentiation.

To validate the proposed method' s effectiveness for multi-speaker localization and tracking in smart meeting room environments, the experimental setup is shown in Figure 6 [Figure 6: see original paper]. The smart meeting room has noise and reverberation settings of $SNR = 20dB$ and $T_{60} = 150ms$, with two circular microphone arrays placed in the room. Three speakers speak simultaneously with trajectories forming right-angle, arc, and straight-line patterns, each speaking for 20 seconds.

Figure 7 [Figure 7: see original paper] shows the tracking results of the particle filter + clustering algorithm: (a) tracking results of particle filter + K-means algorithm, (b) tracking results of parallel particle filter + clustering algorithm.

Figures 8 [Figure 8: see original paper] and 9 [Figure 9: see original paper] present the tracking results on X-axis and Y-axis, respectively. Table 3 compares the tracking errors of the two algorithms in the X direction by selecting positions from the first 10 time instants and computing relative errors between tracked positions and ground truth.

Table 3: Tracking Error Comparison of Two Algorithms in X Direction

Algorithm	Tracking Error Rate (%)
PF + K-means	[values]
Proposed Algorithm	[values]

The error rates in Table 3 show that the proposed algorithm achieves lower error rates than the traditional particle filter and K-means tracking algorithm, demonstrating higher tracking accuracy and better performance.

In summary, the parallel particle filter and GPU-based K-means clustering algorithm for multi-speaker tracking outperforms the particle filter and K-means clustering approach in terms of effective source number performance. The proposed algorithm can not only determine the effective number of sound sources in

real-time but also accurately track multiple speaker positions in high-intensity noise environments.

4 Conclusion

By analyzing the problems existing in particle filter-based multi-speaker tracking, this paper introduces parallel strategies to implement the particle filter algorithm in parallel, reducing algorithmic complexity. To address particle divergence issues when multiple targets appear in the tracking area, a GPU-based K-means clustering method is employed. Experimental results demonstrate that the proposed algorithm effectively improves hardware resource utilization in GPUs, achieves stronger particle convergence, and delivers higher tracking accuracy in multi-target tracking compared to the original particle filter tracking algorithm.

References

[1] Cao Jie, Yu Lizhen. Multi-speaker recognition based on MFCC and motion intensity clustering initialization [J]. Application Research of Computers, 2012, 29(9): 3295-3298.

[2] Qu Dan, Zhang Wenlin. Speaker adaptation algorithm based on eigenphone speaker subspace [J]. Journal of Electronics & Information Technology, 2015, 37(6): 1350-1356.

[3] Zhang Wei, Kang Baosheng. Survey on correlation filter target tracking [J]. Journal of Image and Graphics, 2017, 22(8): 1017-1033.

[4] Ren Hang. Improved particle filter target tracking algorithm based on quasi-Monte Carlo filter [J]. Journal of Electronic Measurement and Instrumentation, 2015(2): 289-295.

[5] Qin Yongyuan, Zhang Hongyue, Wang Shuhua. Kalman Filtering and Principle of Integrated Navigation [M]. 3rd ed. Xi' an: Northwestern Polytechnical University Press, 2015.

[6] Zhai Weixin, Cheng Chengqi. Camshift motion tracking algorithm based on Kalman filter [J]. Acta Scientiarum Naturalium Universitatis Pekinensis, 2015, 51(5): 799-804.

[7] Cheng Lan, Wang Zhiyuan, Chen Jie, et al. Multipath estimation algorithm based on particle filter and moving average extended Kalman filter [J]. Journal of Electronics & Information Technology, 2017, 39(3): 709-716.

[8] Lei Ming, Han Chongzhao, Xiao Mei. A correction method for extended Kalman particle filter algorithm [J]. Journal of Xi' an Jiaotong University, 2005, 39(8): 824-827.

[9] Zhang Yingbo. Jacobian matrix estimation based on unscented Kalman filter algorithm [J]. Journal of Computer Applications, 2011, 31(6): 1699-1702.

[10] Li Tiancheng, Fan Hongqi, Sun Shudong. Particle filter theory, method and its application in multi-target tracking [J]. Acta Automatica Sinica, 2015, 41(12): 1981-2002.

[11] Lin Jing, Yang Jichen, Zhang Xueyuan, et al. Audio feature extraction algorithm based on sparse representation weight tensor [J]. Journal of Computer Applications, 2016, 36(5): 1426-1429.

[12] Sun Haiyang, Zhang Li. Improvement of particle filter algorithm for UAV tracking scenarios [J]. Computer Simulation, 2017, 34(2): 84-87.

[13] Goyal B, Budhraja T, Bhatnagar R, et al. Implementation of Particle Filters for Single Target Tracking Using CUDA [C]// Proc of the 5th International Conference on Advances in Computing and Communications. 2016: 28-32.

[14] Baydoun M, Dawi M, Ghaziri H. Enhanced parallel implementation of the K-Means clustering algorithm [C]// Proc of International Conference on Advances in Computational TOOLS for Engineering Applications. 2016: 7-12.

[15] Li Xiaoyu, Yu Liying, Lei Hang, et al. Parallel implementation and application of an improved K-means algorithm [J]. Journal of University of Electronic Science and Technology of China, 2017, 46(1): 61-68.

[16] Chen Pinghua, Chen Chuanyu. Parallel recommendation algorithm based on full binary tree bisecting K-means clustering [J]. Computer Engineering & Science, 2015, 37(8): 1450-1457.

Note: Figure translations are in progress. See original paper for figures.