ISBN 978-981-11-3671-9
Proceedings of 2017 the 7th International Workshop on Computer Science and Engineering (WCSE 2017)
Beijing, 25-27 June, 2017, pp.48-52
doi: 10.18178/wcse.2017.06.008

Accelerating The JPDA Filter on CPU-GPU Platform

Wen Sun [1+], Lin Gao [1], Xu Tang[1] and Ping Wei [1]

[1] Center for Cyber Security, School of Electronic Engineering University of Electronic Science and Technology of China, Chengdu, 611731, China

Abstract. This paper addresses the computational problem of Joint Probability Data Association (JPDA) filter in multitarget tracking with dense clutter. A parallelization implementation for JPDA on graphics processing unit (GPU) platform under the Compute Unified Device Architecture (CUDA) framework is presented. Where the JPDA filter is resolved into two parts in the view of computation. Specific parallelization scheme is designed for each part. Simulation results show that with the assistance of GPU, JPDA achieves up to a 55x speedup than its CPU implementation under dense clutter, while maintains the same computational accuracy to CPU method.

Keywords: multitarget tracking, dense clutter, compute unified device architecture (CUDA), graphic processing unit (GPU), real-time implementation

1. Introduction

The JPDA filter [1] is a widely used algorithm in the multitarget tracking along with miss detections and clutters. Whereas JPDA has the defect that the exact calculation of association probabilities is not attainable in real-time along with the growth of number of targets and measurements. To alleviate its computational complexity, some suboptimal association algorithms were attempted, e.g. NN-JPDA [3], and suboptimal-JPDA [4, 5]. The hallmarks of suboptimal algorithms are that they sacrificed the tracking accuracy to reach the real-time goal. To reconcile the contradiction between the accuracy and the computation complexity, [6, 7] has addressed the data association problems of multitarget tracking on the multi-core processor, but just considering the sparse clutter circumstance.

This paper is to propose a parallelization scheme for JPDA in multitarget tracking on GPU based on CUDA framework [8,9,10,11]. In the state estimation, we propose to renew all the targets states in parallel; In the data association, we propose to save all the possible cases of joint association events in global memory, and evaluate all matches in parallel. The results in the simulated scenario with dense clutter demonstrate that the proposed scheme has achieved a great speedup on GPU compared with the CPU implementation, but without any accuracy loss.

The reminder of this paper is structured as follows: A roughly introduction to JPDA is illustrated in Section 2. The details of our proposal are illustrated in Section 3. Simulation results and some analysis are exhibited in Section 4. At last, conclusions and expectation of conceivable work in future are in Section 5.

2. Introduction to JPDA

Assuming that T targets are moving in the surveillance area, by given kinematic model F^t, state vectors x_{k-1}^t, covariance matrix P_{k-1}^t and observations $Z_k = \left(z_k^i\right), i = 1,...,m_k$, four steps are carried out to estimate the target states [1]:

- Step 1: State Prediction:

+ Corresponding author. Tel.: + 8613679041690.
E-mail address: sw84114505@126.com.

For $t = 1,...,T$, compute

$$x_{k|k-1}^t = F^t \times x_{k-1}^t \tag{1}$$

$$P_{k|k-1}^t = F^t \times P_{k-1}^t \times F^{t'} + G^t \times Q^t \times G^{t'} \tag{2}$$

$$S_k^t = H \times P_{k-1}^t \times H' + R \tag{3}$$

$$K_k^t = P_{k|k-1}^t \times H' \times \left(S_k^t\right)^{-1} \tag{4}$$

where F^t, G^t, Q^t are dynamic function parameters and H, R are observation function parameters.

- Step 2: Measurement Validation

The validation matrix is denoted as $\Omega = \left[\omega_{jt}\right]$, $j = 1,...,m_k$, $t = 0,1,...,T$, where ω_{jt} is declared 1 if and only if

$$\left[v_k^t(j)\right]' \times \left(S_k^t\right)^{-1} \times v_k^t(j) < \delta \tag{5}$$

where δ is a properly chosen threshold.

- Step 3: Association Probabilities.

The conditional probability of $\theta(k) = \left\{\theta_i(k)\right\}_{i=1}^{\theta_k}$ is given in (6), where c is the normalizing factor, ϕ is the number of clutter along a target, V is the volume of the validation gate. τ_j equals to 1 if j th measurement is assigned to a track and 0 otherwise. δ_t is a binary variable to indicate whether target t is associated with a measurement in $\theta_i(k)$. P_D^t is the detection probability of target t. Finally, $\beta_{jt}(k)$ is calculated by (7), where $\omega_{jt}\left[\theta_i(k)\right]$ is a binary variable indicating whether target t is associated with measurement j in $\theta_i(k)$.

$$p\left[\theta_i(k)\middle|Z_k\right] = \frac{1}{c}\frac{\phi}{V^\phi}\prod_{j=1}^{m_k}\left\{N\left[v_k^t(j)\right]\right\}^{\tau_j} \times \prod_{t=1}^{T}\left(P_D^t\right)^{\delta_t}\left(1-P_D^t\right)^{1-\delta_t} \tag{6}$$

$$\beta_{jt}(k) = \sum_{i=1}^{\theta_k} p\left[\theta_i(k)\middle|Z_k\right] \times \omega_{jt}\left[\theta_i(k)\right] \tag{7}$$

- Step 4: State Updating

For $t = 1,...,T$ and $j = 1,...,m_k$, compute:

$$v_k^t(j) = z_j^k - H \times x_{k|k-1}^t \tag{8}$$

$$v_k^t = \sum_{j=1}^{m_k}\beta_{jt}(k) \times v_k^t(j) \tag{9}$$

$$x_k^t = x_{k|k-1}^t + K_k^t \times v_k^t \tag{10}$$

$$\begin{aligned} P_k^t = {} & P_{k|k-1}^t - \left(\sum_{j=1}^{m_k}\beta_{jt}(k)\right)K_k^t \times S_k^t \times \left(K_k^t\right)' \\ & + K_k^t \times \left(\sum_{j=1}^{m_k}\beta_{jt}(k) \times v_k^t(j) \times \left[v_k^t(j)\right]'\right. \\ & \left. - v_k^t \times \left(v_k^t\right)'\right) \times \left(K_k^t\right)' \end{aligned} \tag{11}$$

In above routine, the execution of (6), (7) takes the majority workload of JPDA, which grows exponentially in multitarget tracking scenario with dense clutters. More details discussion about the computational complexity of JPDA can be found in [12].

3. Parallelization of JPDA

JPDA can be resolved into two steps in the term of parallelization: state estimation and data association. The former is com-posed by step 1, 4, while the latter is composed by step 2, 3.

3.1. Parallelization of state estimation

Note that states of T targets are renewed with no correlation, as in Fig. 1. By considering the computational characteristic of state estimation, we propose a scheme to carry out each KF is executed

serially by a thread. The parallelization scheme can reduce the circulation to only one, which means that a T x speedup can be obtained in theory.

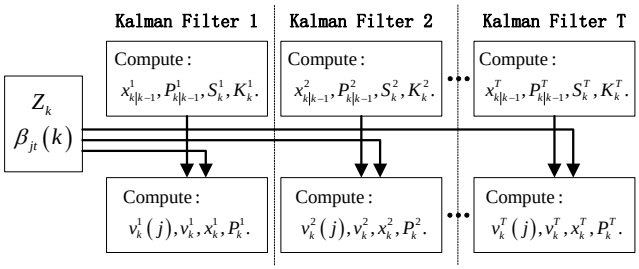

Fig. 1: Computation framework of state estimation.

3.2. Parallelization of data association

For building the validation matrix $\Omega=\left[\omega_{jt}\right]$, $m_k \times T$ threads can be employed directly to examine whether measurement j falls into validation gate of target t by (5), while listing all the joint association events can not be fully parallelized since it is the permutation and combination problem [2]. We propose to compute all possible cases offline and save them in the global memory. For the convenience of GPU to access the data, we propose to compress the joint association event matrix into the association vector:

$$\tilde{\theta}_i(k)=\left(\mathrm{I}_t^k\right), t=1,...,T, \mathrm{I}_t^k=0,\ldots,m_k, i=1,...,\theta_k \tag{12}$$

The dimension of each vector is $1\times T$, I_t^k indicates target t is associated with measurement I_t^k, $\mathrm{I}_t^k=0$ indicates target t is associated with no measurement, which means that target t is undetected in $\tilde{\theta}_i(k)$. Elements in $\tilde{\theta}_i(k)$ can be used as the index to compute $p\left[\theta_i(k)\middle|Z_k\right]$. Thus, (6) can be re-written as (13) and (14). Additionally, we employ the Parallel Prefix Sum (PPS) [13] to compute c in (6), which brings logarithmic decrease of execution times.

To compute $\beta_{jt}(k)$ in (7), we re-project $\tilde{\theta}_i(k)$ back to $\theta_i(k)$ in parallel firstly. If the execution time of a summation is T_p, by employing PPS $m_k\times(t+1)$ times, the execution time of (7) is $\log_2(\theta_k)\times m_k\times(t+1)\times T_p$ on GPU.

$$p\left[\theta_i(k)\middle|Z_k\right]=\frac{1}{c}\frac{\phi}{V^{\phi}}\prod_{t=1}^{T}L\left[\tilde{\theta}_i(t)\right] \tag{13}$$

$$L\left[\tilde{\theta}_i(t)\right]=\begin{cases}\exp\left\{-\frac{1}{2}\left[z\left(\tilde{\theta}_i(t)\right)-H\cdot x_{k|k-1}^t\right]^{\mathrm{T}}\right. \\ \quad\left.\times\left[S_k^t\right]^{-1}\left[z\left(\tilde{\theta}_i(t)\right)-H\cdot x_{k|k-1}^t\right]\right\} \\ \quad\times\left(P_D^t\right)^{\delta_t}\times\left(1-P_D^t\right)^{1-\delta_t}, z\left(\tilde{\theta}_i(t)\right)\neq 0 \\ \left(P_D^t\right)^{\delta_t}\times\left(1-P_D^t\right)^{1-\delta_t} \quad \text{, otherwise}\end{cases} \tag{14}$$

4. Simulations

4.1. Simulation environment

In this simulations, JPDA filter are carried out on GPU and CPU respectively. The hardware and software resources are illustrated in Table 1.

TABLE 1: SIMULATION ENVIRONMENT

	CPU	GPU
Hardware	Intel(R) Core(TM) i5-4430 @ 3.00GHz, 16.0GB RAM	NVIDIA GeForce GTX 760 @ 1.10GHz, 2.0GB RAM
Software	Matlab R2013a 64-bit version	Visual Studio 2010 CUDA Toolkit v6.0

4.2. Simulation scenario

Assume 3 targets move in monitor are and they follow a constant velocity motion model for 50 sampling times. The kinematic model F^t, initial state covariance matrix P^t, noise gain matrix G^t and noise

covariance matrix Q^t are the same and given in (15)-(16). Measurements are sampled with the interval $T_s = 1$, based on a single sensor. The observation matrix H and the covariance matrix of which is given in (17). Assuming that there is no missing detection. The threshold of the confirm gate is $\delta = 9.21$.

$$F^t = \begin{bmatrix} 1 & T_s & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T_s \\ 0 & 0 & 0 & 1 \end{bmatrix}, P^t = \begin{bmatrix} 100^2 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 100^2 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix} \tag{15}$$

$$G^t = \begin{bmatrix} T_s^2/2 & 0 \\ T_s & 0 \\ 0 & T_s^2/2 \\ 0 & T_s \end{bmatrix}, \quad Q^t = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \tag{16}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad R = \begin{bmatrix} 100^2 & 0 \\ 0 & 100^2 \end{bmatrix} \tag{17}$$

4.3. Accuracy comparison

The performances of JPDA are examined on GPU and CPU with the expected clutter number along a target $\phi = 5$. We employ the mean RMMSE of the target positions as an indicator. The results under 200 independently simulations per platform are illustrated in Fig. 2. As the result, the implementation of JPDA on the GPU has the same level of accuracy with that on the CPU.

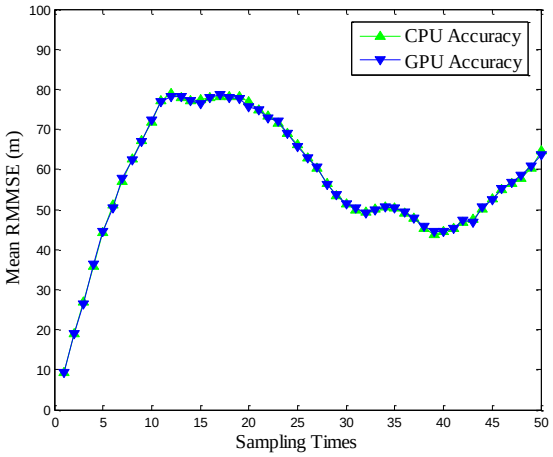

Fig. 2: Mean RMMSE of 3 targets on GPU and CPU.

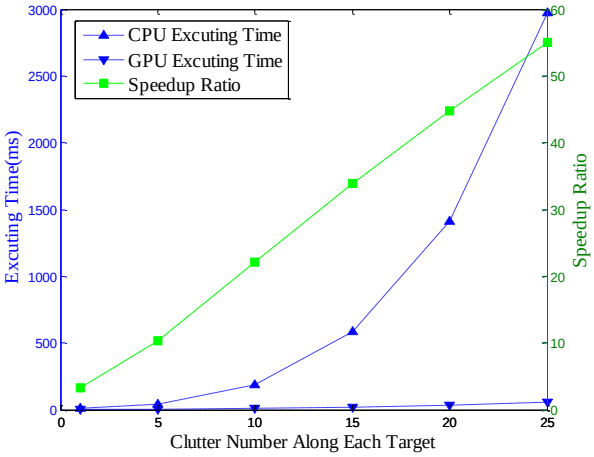

Fig. 3: Execution time of JPDA and the speedup on GPU versus CPU.

4.4. The speedup of GPU to CPU

To examine the speedup ratio between two platforms in situations, we artificially manipulate the clutter number along each target to $\phi = 1, 5, 10, 15, 20, 25$. The speedup of GPU versus CPU is illustrated in Fig. 3. As the result, the execution time on GPU increases slightly when clutter density rises, since the state estimation and data association are implemented in parallel. The extra time is spent on data exchanges between host and device. While the implementation on CPU has to face the NP-Hard problem and results an unacceptable time cost.

5. Conclusions

In this paper, we solved the computational problem of JPDA by implementing it with the assistance of GPU. The proposed scheme took full advantage of the parallel feature of GPU to compute multiple parallelized operations in the state estimation and joint association events probability in the data association

of JPDA. The proposed scheme is examined through a simulated scenario with dense clutter on a specified GPU. A 55x speedup is obtained with 25 clutters along each target. It exhibits that the JPDA filter can be real-time implemented in the dense clutter scenario by employing GPU. The future work will focus on extending the parallelization scheme to other association based multitarget tracking algorithms, and their maneuvering tracking versions.

6. References

[1] J. Li, Y. Jiang, R. Fan. Recognition of Biological Signal Mixed Based on Wavelet Analysis. In: Y. Jiang, et al (eds.). *Proc. of UK-China Sports Engineering Workshop*. Liverpool: World Academic Union. 2007, pp. 1-8. (Use "References" Style)

[2] R. Dewri, and N. Chakraborti. Simulating recrystallization through cellular automata and genetic algorithms. *Modelling Simul. Mater. Sci. Eng*. 2005, **13** (3): 173-183.

[3] A. Gray. *Modern Differential Geometry*. CRE Press, 1998.

[4] R. Dewri, and N. Chakraborti. Simulating recrystallization through cellular automata and genetic algorithms. *Modelling Simul. Mater. Sci. Eng*. 2005, **13** (3): 173-183.

[5] J. Li, Y. Jiang, R. Fan. Recognition of Biological Signal Mixed Based on Wavelet Analysis. In: Y. Jiang, et al (eds.). *Proc. of UK-China Sports Engineering Workshop*. Liverpool: World Academic Union. 2007, pp. 1-8. (Use "References" Style)

[6] T.E. Fortmann, Y. Bar-Shalom, and M. Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE J. Oceanic. Eng.*1983, **8** (3):173-184.

[7] J.B. Collins and J.K. Uhlamnn. Roberts. Efficient gating in data association with multivariate gaussian distributed states. *IEEE Trans. Aerosp. Electron. Syst.*1992, **28** (3): 909-916.

[8] J.A. Roecker, G.L. Phillis. Suboptimal joint probabilistic data association. *IEEE Trans. Aerosp. Electron. Syst.* 1993, **29**, (2): 510-517.

[9] Y. Bar-Shalom. *Multitarget-Multisensor Tracking: Applications and Advances*. Norwood, MA: Artech House, 1990.

[1] R.L. Popp, K. Pattipati, Y. Bar-Shalom and R. Ammar. Shared-memory parallelization of the data association problem in multitarget tracking. IEEE *Trans. Parall. Distr. Syst*. 1997, **8**, (10), 993-1005.

[10] R.L. Popp, K. Pattipati, Y. Bar-Shalom and M. Yeddanapudi. Parallelization of a multiple model multitarget tracking algorithm with superlinear speedups. *IEEE Trans. Aerosp. Electron. Syst.* 1997, **33**, (1), 281-290.

[11] J. Sanders and E. Kandort. *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley Professional. 2010.

[12] M.A. Chao, C.Y. Chu, C.H. Chao and A.Y. Wu. Efficient parallelized particle filter design on CUDA. In: M.A. Chao, et al (eds.). *Proc. IEEE Workshop Signal Process*. 2010, pp. 299-304.