A Parallel Implementation of Hypothesis-Oriented Multiple Hypothesis Tracking

[1]Lin Wu, [1]Fei Wang, [1]Yongjun Xu, [1,2]Yu Jiang, [1,2]Jiakai Wang
[1] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2] University of Chinese Academy of Science, Beijing, China
{wulin, wangfei, xyj, jiangyu, wangjiakai19g}@ict.ac.cn

***Abstract*—Hypothesis-oriented Multiple Hypothesis Tracking (HOMHT) recursively generates hypotheses on the origins of measurements and manages them, therefore it is computationally intensive. To speed up HOMHT for tracking hundreds of targets in real time, we propose a parallel implementation of this algorithm which distributes hypotheses into independent worker threads residing in multiple CPU cores. The implementation in this paper is based on object-oriented programming: each hypothesis object manages its target data all by itself and the generation and pruning of a hypothesis is achieved by its copy constructor and destructor functions. We evaluate this method by tracking 150 targets through 3 heterogeneous sensors in real-time with 32-best hypotheses running in 1, 2, 4, 8, 16 and 32 worker threads respectively. The results validate the method's scalability in which measurement fusion latency is approximately inversely proportional to worker thread count. We also make a pressure test by tracking 500 targets through 3 sensors, and HOMHT is able to run concurrently in real-time with 32 worker threads.**

Keywords—multiple hypothesis tracking, MHT, HOMHT, parallel computing, hypothesis-oriented MHT

I. INTRODUCTION

Data association is a major challenge when tracking multiple targets especially if the ratio of average distance among targets to positioning error (i.e. Gap Error Ratio, GER for short) is low [1], because there exists high uncertainty about the origin of each measurement. Methods like GNN (Global Nearest Neighbor) [2], JPDA (Joint Probabilistic Data Association) [3] and multidimensional assignment [4] are prone to errors when ambiguities exist as they maintain only one hypothesis by making a "hard" decision on the current scan and cannot correct previous mistakes. MHT (Multiple Hypothesis Tracking) [5] improves association accuracy by keeping a hypothesis tree (hypothesis-oriented MHT, or HOMHT for short [5]) or a track forest (track-oriented MHT, or TOMHT for short [6]) over time and postponing decisions until ambiguity is resolved or computing resource is exhausted [7][8].

When tracking targets by HOMHT, one or more hypotheses may be spawned from each existing one for every new measurement acquired. Therefore nodes on the hypothesis tree grow exponentially with respect to the count of measurements received. As marine [9] and air traffic is quite busy nowadays, modern fusion systems need to associate large number of target measurement data from many heterogeneous and asynchronous sensors in real time, which is computationally intensive. On the other hand, the development of computing technologies has brought the ability of running an algorithm concurrently on multiple CPU (Central Processing Unit) cores and computers, making it possible to speed up tracking algorithms like MHT. However, there are few works considering the heterogeneity of sensors or utilizing the power of parallel computing to track numerous targets.

Previous works usually focused on tracking accuracy and evaluated their methods by offline simulations, assuming there are synchronous measurements from less than 10 targets [4][10][13][14][15] detected by sensors in batches scan by scan (or frame by frame), which is usually not satisfied when tracking targets in a large area. Michael and Joydeep [16] proposed several enhancements to improve computational efficiency of HOMHT, but they assumed synchronization and modeled tracking as a multi-frame association problem. Their method was tested with 100 stationary targets detected by 3 sensors. Angelos etc. [17] applied a clustering algorithm in offline HOMHT to save execution time and tested it in highway scenarios with targets less than 80. In addition, methods like multi-stage or distributed MHT have been proposed [18][19][20][21][22], which perform parallel single-sensor tracking followed by downstream multi-sensor fusion. These methods run multiple tracking algorithms in hierarchal fusion nodes (i.e., distribute the tracking process over multiple nodes) rather than distributing a single algorithm in multiple threads, and are mainly designed for fusion in sensor networks.

Although TOMHT has been proposed as a more efficient alternative to HOMHT, it is hard to run concurrently in multiple CPU cores or computers because every global hypothesis is composed of a subset of nodes in a track forest and the multi-dimensional assignment problem needs to be solved for forest pruning. On the other hand, each global hypothesis in HOMHT can be represented by a leaf node alone. Therefore, HOMHT is ideal for parallel implementation because a hypothesis can manage tracks in it and spawn new hypothesis independently in its own thread and different threads do not need to contend for the access to a memory block.

During the implementation of MHT, it is important to keep in mind that the number of tracks in each hypothesis may be different, and the state estimation of a specific track is not the same in different hypotheses. Thus each hypothesis needs to manage its own track set rather than sharing a global track set with other hypotheses, and the calculating speed of this process can also benefit from parallel computing. Meanwhile a software to stitch tracks between two hypotheses is needed for consistent output, because the best hypothesis picked out by a master thread will be changing with measurements accumulating. This software, which is called tracklet stitcher in this paper, communicates with HOMHT tracker and other applications like human computer software through a message broker in publish-subscribe pattern.

This work is partially supported by NSFC No. 61902376 and NSFC No. 61702487.

The structure of this paper is as follows. Section II introduces multiple target tracking problem and HOMHT algorithm. Section III describes our parallel implementation of HOMHT. Section IV evaluates the method with 150 and 500 targets measured by 3 heterogeneous sensors simulated by the open source simulator ICTTargets [1]. Section V concludes the paper.

II. Hypothesis-Oriented Multiple Hypothesis Tracking

A. Multiple Target Tracking Problem

Previous works usually assumed that multiple sensors are synchronized and measurements are being received frame by frame [11][12]. In this paper, we consider a more general situation in which sensors are heterogeneous (e.g., Synthetic Aperture Radar, optical remote sensing, Electronic Support Measures, Automatic Dependent Surveillance-Broadcast) and asynchronous. Different sensors are observing a portion of targets with diverse revisit intervals and sending measurements to a fusion node one by one through networks in stream rather than in batches.

Suppose there are S sensors monitoring a region with N targets in total and sensor s is able to capture N_s targets. For sensor s, each of the N_s targets is being detected by it with probability $p_D(s)$ at a mean interval of T_s, and the measurement $z_s^{t_s^j}$ generated at time t_s^j will be sent to fusion node through networks as stream. Due to the asynchronization of detection, time t_s^j is a continuous variable rather than a discrete one and $z_s^{t_s^j}$ is the *j-th* measurement generated by sensor s since start monitoring.

The track τ_i of target *i* in association hypothesis λ_k is a set of measurements $z_s^{t_s^j}$ from *S* sensors which are hypothesized to originate from the same target, and the hypothesis λ_k with N_k targets can be viewed as a partition of measurements (neglect the subscript *k* for simplicity):

$$\tau_i = \left\{ z_s^{t_s^j} | s \in \{1,2,\dots,S\}, z_s^{t_s^j} \; is \; from \; target \; i \right\} \quad (1)$$

$$\lambda_k = \left\{ \tau_1, \dots, \tau_{N_k} | \tau_m \cap \tau_n = \emptyset, 1 \le m, n \le N_k, m \ne n \right\} \quad (2)$$

A major challenge of multiple target tracking is finding the optimal partition of measurements, i.e. data association problem. Methods like GNN and JPDA only keep one best hypothesis (i.e., the value of subscript *k* is always 1), and new measurements are used to update this hypothesis. In spite of their computational efficiency, false association decisions made in the past will corrupt states of tracks and cannot be corrected afterwards. On the other hand, MHT keeps multiple candidate hypotheses and the maximum value of *k* depends on computational resources available.

B. HOMHT

When GER [1] is low (i.e., high target density and large measurement noise), there will be ambiguities about the partition of measurements. HOMHT recursively generates, evaluates and prunes hypotheses to make use of history information. Upon receiving a new measurement $z_s^{t_s^j}$ in fusion node, it may be associated with an existing target, with a new target, or with a false alarm. Thus each hypothesis λ_k may generate one or more new hypothesis. These hypotheses are kept until low probability ones are pruned when ambiguity is resolved or computational resource is going to be exhausted.

III. Parallel Computing of Multiple Target Tracking

To speed up HOMHT for tracking large number of targets in real time, we make use of the parallel computing capability in modern computing systems.

Real-time programs must guarantee response within specified time constraints. In the context of target tracking, a real-time tracking program must be able to fuse measurements at a rate faster than that they are being received. Otherwise, measurements waiting to be processed will accumulate over time and fusion latency will increase continuously. In this case, we should improve computation speed or lower the rate of measurement stream to recover the program's real-time processing. The term "real-time" is also used in simulation to mean that the simulation's clock runs at the same speed as a real clock.

A. System Design

The parallel computing system includes a HOMHT tracker and a tracklet stitcher, communicating with other components like multiple sensors (data sources) and Human Computer Interaction (HCI) software through an open source message broker RabbitMQ [24], see Fig. 1. Messages transmitted in the system are serialized by Google's protocol buffers, which is a language-neutral, platform-neutral and extensible mechanism for serializing structured data [25].

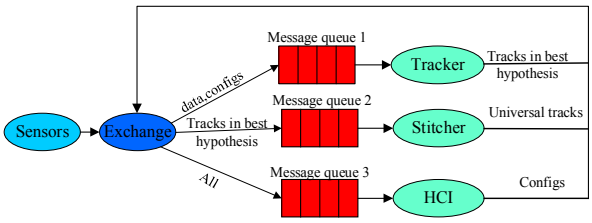

Fig. 1. The message stream inside the real-time tracking system converges in the topic exchange of RabbitMQ and are published to subscribers.

In this real-time multi-sensor multi-target tracking system, sensors (physical devices or target simulators) publish measurement data to HOMHT tracker as well as HCI software in real time. Once a new message has been fetched from its message queue, HOMHT tracker copies it to every existing and newly generated hypothesis running concurrently in different worker threads. At the same time, the master thread will select the best hypothesis λ_k and notify it periodically. Inside the best hypothesis, it will send fused data to tracklet stitcher in stream through RabbitMQ. When the value of k changes, tracklet stitcher will associate tracks between them using methods like GNN. In this way, universal tracks output from stitcher are consistent regardless of k. During tracklet stitching, new targets will be created and false targets will be destroyed when there are unmatched tracks in previous or current best hypothesis. Measurement data and fused universal tracks are subscribed by HCI for visualization. Besides, users can configure parameters of HOMHT on line through HCI.

In the RabbitMQ messaging model, a queue is a buffer that stores messages and it is bound with several routing keys. Applications send messages to an exchange, which is responsible for pushing messages to certain queues subscribing routing keys associated with these messages. In

this way, applications can be distributed inside a network and synchronization is not needed. From the Web UI of RabbitMQ management, we can see states of message queues that applications are consuming: consumer utilization (defined as proportion of time that a queue's consumers could take new messages), message count waiting to be consumed, count of consumers, etc. If a queue has a consumer utilization of 100%, then it never needs to wait for its consumers and it is always able to push messages out to them as fast as it can. In this case, the system can be regarded as tracking targets in real-time.

B. Parallel computing of HOMHT Tracker

In the process of multi-sensor multi-target tracking, child hypotheses are generated by their parents and moved to worker threads when ambiguities exist. The tree representation of hypotheses running in independent and asynchronous threads is depicted in Fig. 2. In our implementation, a leaf node alone contains all the information of its ancestors and represents a global hypothesis. Therefore branch nodes in this tree do not need to be stored in memory. As multiple independent hypotheses are processing measurements asynchronously, each one has processed different count of measurements at any time. The master thread will compare their probabilities only in the same level of tree (e.g., the third level on the origin of measurement 2 in Fig. 2) to pick out the best hypothesis or prune low probability ones.

When there are M execution threads available in CPUs, we start no larger than M worker threads as containers for K-best hypotheses [7], otherwise thread switching will hinder computing performance. For example, an Intel Xeon Processor E5-2680 v4 launched in 2016 has 14 cores and 28 threads of execution. On a computer installed with two CPUs of this model, we can run 56 hypotheses distributed in 56 threads and the computational complexity in each thread is approximately the same as a nearest neighbor method.

The flow chart and UML (Unified Modeling Language) description of parallel HOMHT are depicted in Fig. 3 and Fig. 4 respectively. Upon receiving a new measurement, the master thread will copy and pass it to every hypothesis distributed in multiple worker threads. Inside any hypothesis λ_k, the new measurement may be associated with an existing target, with a new target, or with a false alarm. When there are ambiguities, λ_k will spawn a new hypothesis λ_j which copies all the information in λ_k and diverges since this new measurement. This can be achieved by a copy constructor of class Hypothesis in OOP (Object-Oriented Programming, we use C++ in this paper), which is a member function initializing an object using another object of the same class:

Hypothesis::Hypothesis (const Hypothesis &parentHypo)

Inside this function, it will copy members of parentHypo. With this copy constructor function, a child hypothesis can be generated by the following code:

*Hypothesis *childHypo = new Hypothesis(parentHypo);*

After a new hypothesis has been constructed, it will be moved to a worker thread randomly or by turns and start running:

childHypo->moveToThread(workerThread);

Hypothesis pruning is performed whenever the count of hypotheses is above a predefined threshold K. In the process

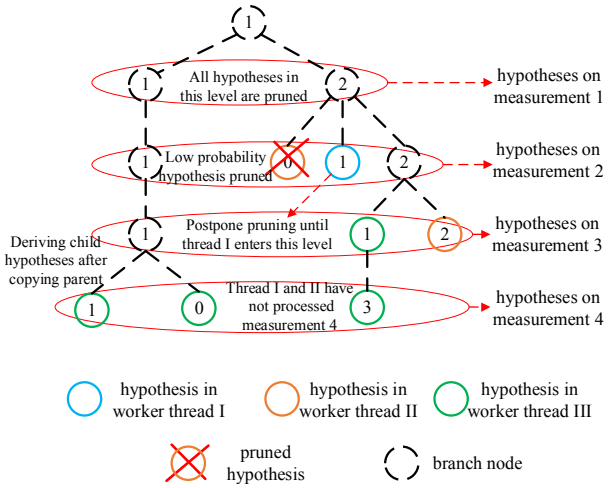

Fig. 2. Tree representation of hypotheses running in different threads. The numbers in the nodes indicate to which track the measurement is associated.

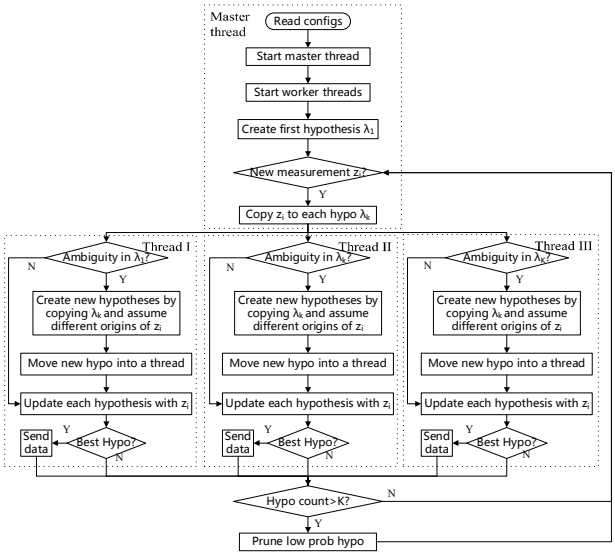

Fig. 3. Flow chart of parallel HOMHT. For simplicity, an example of 3 worker threads is shown.

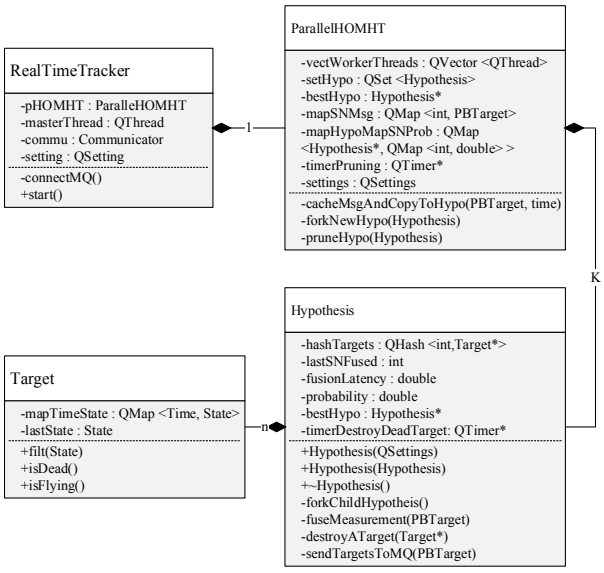

Fig. 4 UML description of main classes in parallel HOMHT. In

of pruning, the hypothesis with the lowest probability will be destroyed through a destructor of class Hypothesis in OOP:

Hypothesis::~ Hypothesis ()

With this destructor function, pruning a hypothesis can be achieved by the following code:

delete hypo;

The master thread will pick out the best hypothesis periodically or when the current best hypothesis is pruned. When the id of best hypothesis is changed, the master thread will notify all the hypotheses, because only the best hypothesis need to send fused tracks to the exchange of RabbitMQ in real time.

C. Tracklet Stitching

In the implementation of parallel HOMHT, each hypothesis runs independently in its own thread. Thus it is highly possible that there are different count of targets in these hypotheses. Besides, track ID and states of the same target will be different in these hypotheses. So we keep a map recording the relationship between track IDs in the best hypothesis and those global track IDs output by tracklet stitcher.

When a new best hypothesis is selected out by master thread, we need to stitch its tracks with those in previous best hypothesis to output consistent results. To improve stitching accuracy, the new best hypothesis will send recent tracklet rather than the last state of each target as its first message. Then the mapping of track IDs in this new best hypothesis and global track IDs will be established by performing track association algorithms such as GNN. When there is no corresponding target in the new hypothesis, the stitching program will send a command of destroying this global target to RabbitMQ. This process is depicted in Fig. 5.

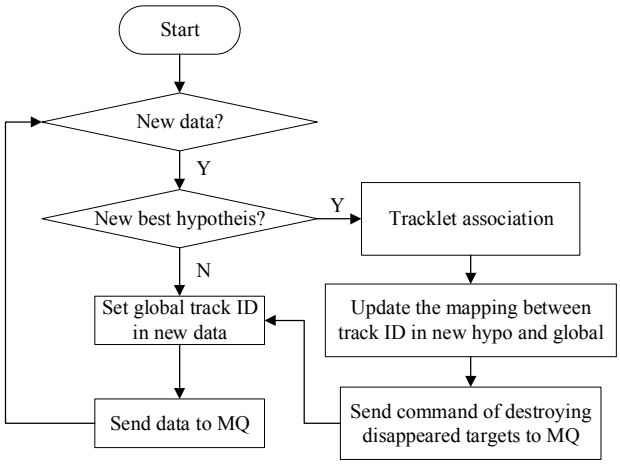

Fig. 5. The process in tracklet stitcher.

IV. Evaluation

Most previous works [13][14][15][26] evaluated tracking methods in an offline manner and processed data in batches. In this paper, we takes advantage of the customizable tracking benchmark proposed in 22nd International Conference on Information Fusion (FUSION2019) [1][23], which consists of an open source target simulator and a free visualization software. The simulator in this benchmark suit reads parameters from files and then simulates detections of targets from multiple sources in real-time. For the evaluation of this paper, a measurement is defined as:

$$z_s^{t_s^j} = < t_s^j, longitude_s^j, latitude_s^j > \quad (3)$$

To keep average target density constant throughout simulation, targets are bounded by a customized polygon. Each target moves with constant velocity when it is far from boundary. When a target is approaching or departing boarder, it slows down or speeds up by 3 m/s^2 until its speed is 0 or reaches this target's maximum velocity. A target will make a random degree turn on the boarder. The motion model is unknown to the tracking program.

In this paper, the parallel computing of HOMTH is implemented by C++ and it runs concurrently in a computer with 2 Xeon E5-2680 v4 CPUs installed. Inside each CPU, there are 14 cores and 28 threads of execution. This type of CPU was launched in first quarter of 2016 by Intel.

A. Scalability of Parallel HOMHT

To evaluate the scalability of parallel computing, we simulate a scenario with 150 targets in an area about 400 kilometers * 700 kilometers being observed by 3 sensors, see TABLE I. Each sensor can only capture part of these targets with a fixed detection probability due to factors such as occlusions, device characteristics, etc. As a result, a target *i* is being detected by a fixed number N_i of sensors throughout simulation. But values of N_i varies from 0 to 3, presenting a more challenging problem compared with scenarios in which every sensor can detect almost all targets. All the above information is unknown to the tracking system.

Their revisit time ranges from 4 to 6 seconds and measurements are generated one by one as streams rather than in batches or scans. In other words, at any moment there will be at most one measurement generated by a specific sensor. These sensors has a detection probability of 90% in each cycle and the standard deviation of positions measured ranges from 300 to 500 meters.

TABLE I. Scenerio configuration with 150 targets

sensor	targets detected	revisit /seconds	detection probability	noise /meters
1	60%	4	90%	500
2	30%	6	90%	300
3	40%	5	90%	400

To reduce message rate, we configured each sensor to store measurements in a cache and serialize them into a single message per second. Therefore, the message rate is 1/second and there are about 38 measurements with disparate timestamps in each message. The target ID of the same target may be different in different hypothesis, and tracklet stitcher is responsible for replacing it in the current best hypothesis with a global value and providing consistent results.

The maximum speed of targets simulated is between 300 and 937 knots, see Fig. 6. The ground truth and measurements of this scenario is depicted in Fig. 7. In the southwest of this region, target density is so high that positioning errors are larger than gap among targets, see Fig. 8 and Fig. 9.

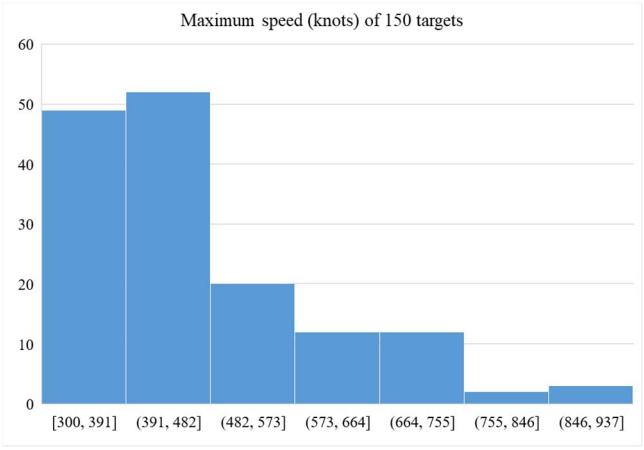

Fig. 6. The distribution of 150 targets' maximum speed. Horizontal axis: speed (knots) ranges. Vertical axis: target count.

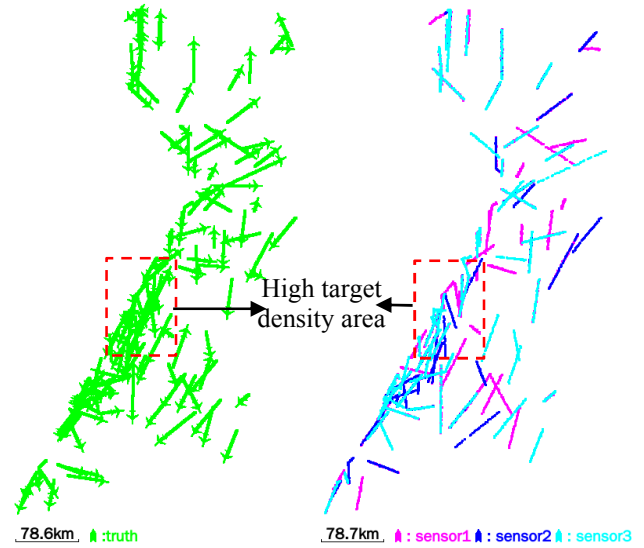

Fig. 7. The scenario simulated with 150 targets. Left: ground truth tracks for several minutes. Right: measurements received from 3 sensors for several minutes marked by different colors.

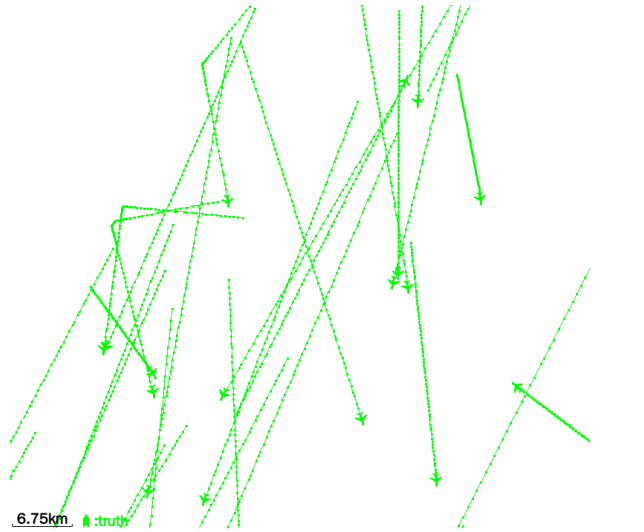

Fig. 8. Ground truth tracks in a high target density area of this region for several minutes. Targets will slow down before they make turns on the boundary.

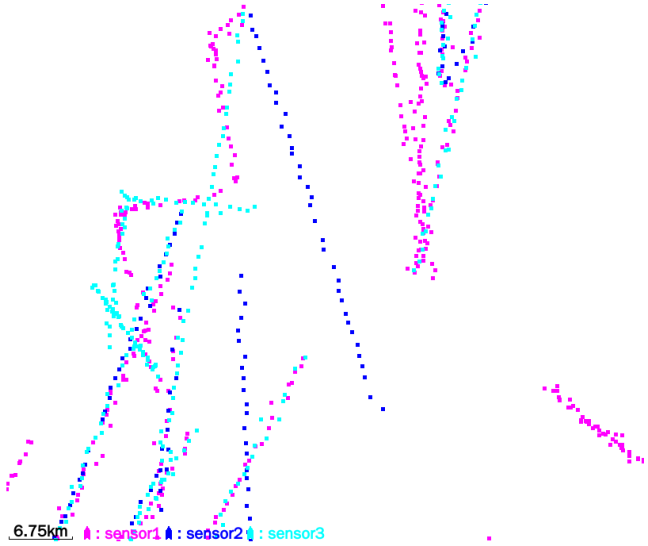

Fig. 9. Measurements received from 3 sensors in a high target density area of this region for several minutes. Each target is being detected by 0 to 3 sensors.

Measurements in a message are fused one by one independently. We define fusion latency as the time between a measurement is received and fusion is completed. The latency will keep growing if parallel HOMHT cannot process measurements faster than the rate they are being received. Otherwise, HOMHT can continuously run in real time because all the measurements are being fused almost immediately. Denote the time it takes to fuse the j-th measurement as t_j, then the fusion latency l_j of this measurement is:

$$l_j = \sum_{i=1}^{j} t_i \tag{4}$$

Assume that all t_j has the same value T_m, then we can simplify (uation (4) as:

$$l_j = jT_m \tag{5}$$

Suppose the mean count of measurements in a message is M. To run HOMHT in real-time when message rate is 1/second, it should be satisfied that:

$$MT_m \leq 1 \tag{6}$$

The mean value of fusion latency is:

$$\mu(l_j) \approx 0.5MT_m \leq 0.5 \tag{7}$$

Considering there are extra latency due to switching threads, data copying and message parsing, etc., the mean value of fusion latency of a real-time tracker under this scenario can be slightly larger than 0.5 seconds.

We compare the fusion latency with 32 hypotheses run in 1, 2, 4, 8, 16 and 32 threads respectively to inspect the scalability with respect to thread count, see Fig. 10 and Fig. 11. We can observe that the latency is approximately inversely proportional to thread count and its scalability is validated. It is also noted that the latency fluctuates widely through time, which may result from the intermittent burst of ambiguities when targets are moving, accelerating and crossing.

As this paper focuses on speeding up HOMHT by parallel computing, we use target count error ratio alone to estimate its accuracy, which is defined as target count output by tracker divided by true count of targets visible to at least one sensor. Among these 150 targets, 115 are visible to at least one sensor. The target count error ratio is depicted in Fig. 12. Since the 70-th second, target count error ratio drops to 0 and rises to 0.87% (115 ± 1 targets) occasionally. Considering GER [1] is quite low in some areas (see Fig. 9), this error ratio is satisfactory. With more hypotheses, the accuracy may be further improved.

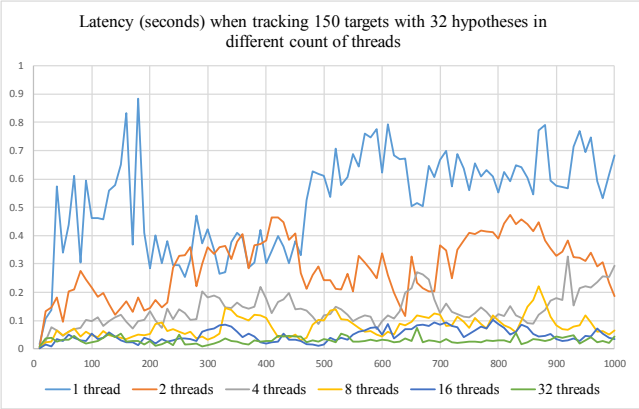

Fig. 10. The latency when 32 hypotheses of HOMHT run concurrently in different count of threads. Horizontal axis: time (seconds) HOMHT has run. Vertical axis: average latency (seconds) before a measurement has been fused into a target since it was received.

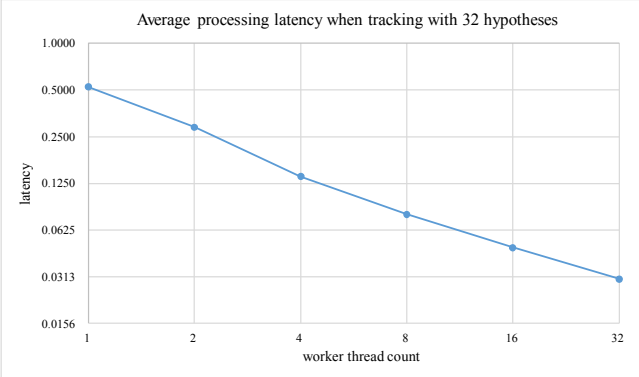

Fig. 11. Average processing latency when 32 hypotheses of HOMHT runs in different count of worker threads.

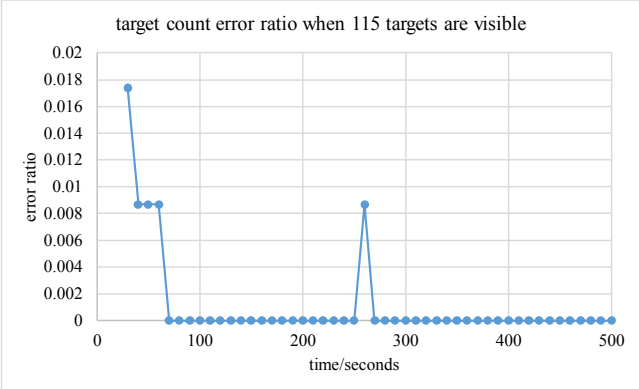

Fig. 12. Target count error ratio of HOMHT since the 30th second. During the first 30 seconds, plenty of targets have not been measured yet. In this scenario, 150 targets are being observed by 3 sensors and 115 targets can be detected by at least one of the sensors.

B. *Pressure Test*

To evaluate the tracking capacity of parallel HOMHT in our computer, we simulate a scenario with 500 targets and keep all other parameters unchanged, see Fig. 13, Fig. 14 and Fig. 15. The distribution of maximum speed is from 300 to 996 knots, see Fig. 16. The average latency of measurement fusion is 0.29 seconds, see Fig. 17. Since the 130-th second, target count error ratio drops below 2% and the mean value is about 0.9%, see Fig. 18. We have also tried to track 1000 targets with 3 sensors by HOMHT, but the latency keeps growing, indicating that the target capacity of this real-time system running on our computer is between 500 and 1000.

As a conclusion, the proposed parallel implementation of 32-best HOMHT is capable of performing real-time 3-sensor 500-target tracking on our computer. Owing to its scalability, it is assumed to be capable of tracking 500 targets by N-best HOMHT in a computer or cluster with threads of execution no less than N. Combined with other techniques like clustering [27][28], thousands of targets can be tracked in real-time.

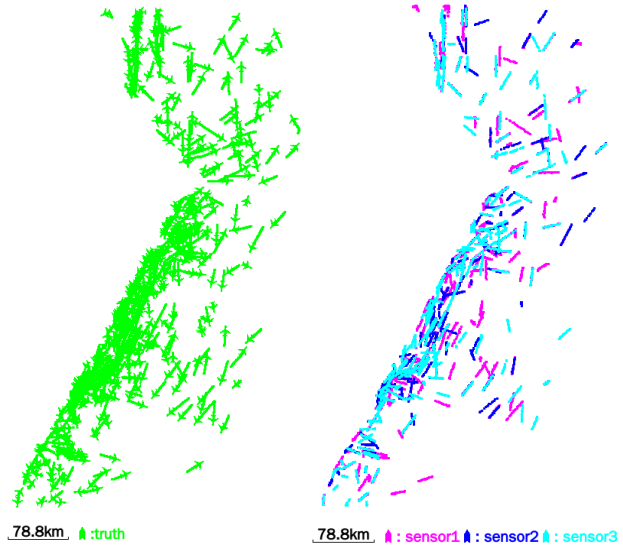

Fig. 13. The scenario simulated with 500 targets. Left: ground truth tracks for several minutes. Right: measurements received from 3 sensors for several minutes.

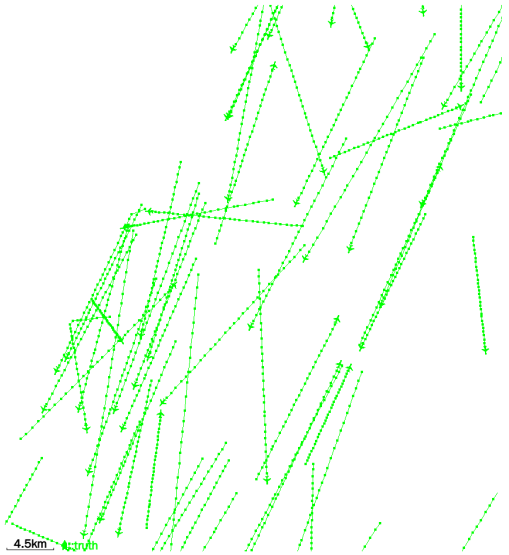

Fig. 14. Ground truth tracks in a high target density area of this region for several minutes.

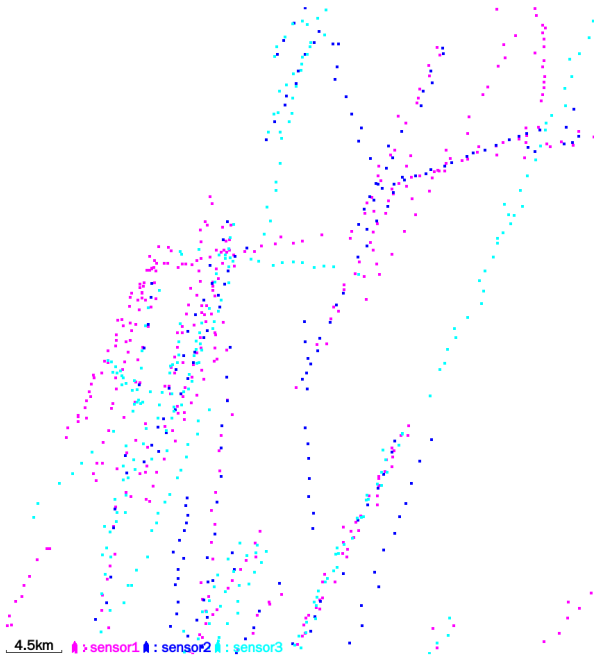

Fig. 15. Measurements received from 3 sensors in a high target density area of this region for several minutes. Each target is being detected by 0 to 3 sensors.

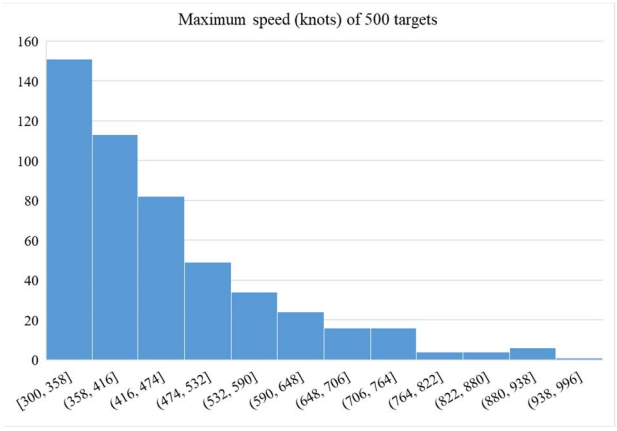

Fig. 16. The distribution of 500 targets' maximum speed. Horizontal axis: speed (knots) ranges. Vertical axis: target count.

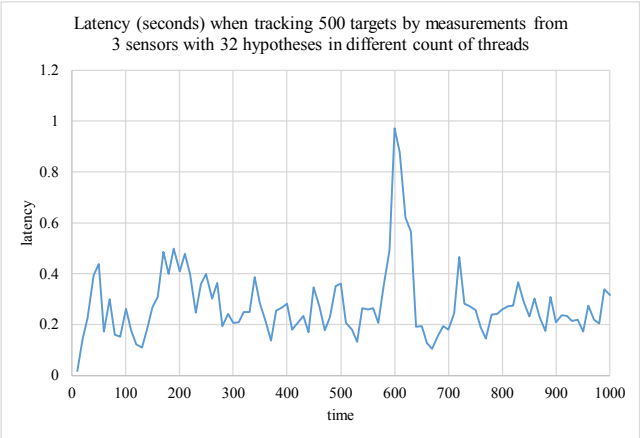

Fig. 17. Fusion latency when tracking 500 targets with 32 hypotheses of HOMHT running concurrently in 32 threads, average value is 0.29 seconds. Horizontal axis: time (seconds) HOMHT has run. Vertical axis: average latency (seconds) before a measurement has been fused into a target since it was received.

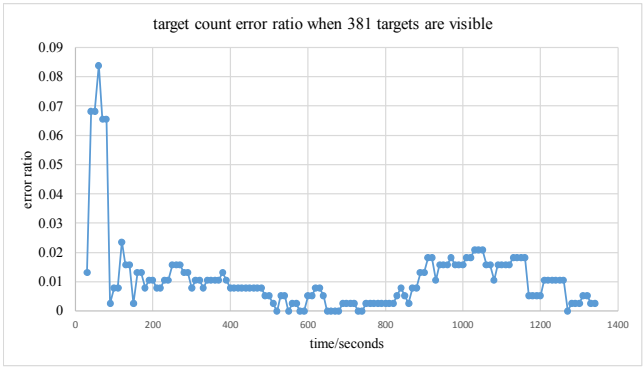

Fig. 18. Target count error ratio of HOMHT since the 30th second. During the first 30 seconds, plenty of targets have not been measured yet. In this scenario, 500 targets are being observed by 3 sensors and 381 targets can be detected by at least one of the sensors.

V. Conslusion

The adoption of hypothesis-oriented multiple hypothesis tracking is hindered by its computational complexity. In this paper, we proposed a parallel implementation of HOMHT, managing hypotheses by worker threads from multiple CPU cores or computers independently. The result of 3-sensor 150-target tracking with 1, 2, 4, 8, 16 and 32 threads validated its scalability: measurement fusion latency is approximately inversely proportional to thread count. The pressure test of 3-sensor 500-target showed that the proposed method is capable of running in real-time with an average fusion latency of 0.29 seconds. Owing to its scalability, it is capable of tracking 500 targets by N-best HOMHT in a computer or cluster with threads of execution no less than N.

Instead of assuming synchronization among multiple sensors and associating offline measurement data scan by scan or frame by frame in batches, we considered a more general and realistic scenario in which measurements are generated and communicated in streams and at any moment there will be at most one measurement generated by a specific sensor. The parallel HOMHT proposed in this paper proved to be able to fuse measurement data one by one in real time rather than scan by scan.

As described in Section III, hypothesis management is achieved by constructor and destructor functions of class *Hypothesis*. The continuously allocation and de-allocation of memory will be time consuming, considering each hypothesis contains its own target set. However, this is inevitable because association ambiguities keep emerging when tracking multiple targets and the same target has disparate states in different hypotheses. To accelerate memory allocation and de-allocation, we can distribute HOMHT among multiple computers. Besides, we can combine the parallel computing method in this paper with other techniques like clustering to further improve computational efficiency for tracking thousands of targets in real-time.

References

[1] L. Wu, Y. Xu, F. Wang, Q. Lu and M. Hu, "Real-time Ship Track Association: a Benchmark and a Network-Based Method," 2019 22th International Conference on Information Fusion (FUSION), Ottawa, ON, Canada, 2019.

[2] S. Blackman, R. Popoli, "Design and analysis of modern tracking systems," Arthech House, 1999.

[3] T. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Sonar tracking of multiple targets using joint probabilistic data association," IEEE Journal of Oceanic Engineering, 8(3), pp. 173-184, 1983.

[4] M. R. Chummun, T. Kirubarajan, K. R. Pattipati and Y. Bar-Shalom, "Fast data association using multidimensional assignment with clustering," in IEEE Transactions on Aerospace and Electronic Systems, 37(3), pp. 898-913, 2001.

[5] D. B. Reid. "An algorithm for tracking multiple targets." IEEE Transactions on Automatic Control, 24(6), pp. 843-854, 1979.

[6] T. Kurien, "Issues in the design of practical multitarget tracking algorithms," in Multitarget-Multisensor Tracking: Advanced Applications, Chapter 3, Y. Bar-Shalom, Ed. Artech House, 1989.

[7] R. Danchick and G. E. Newnam, "Reformulating Reid's MHT method with generalised Murty K-best ranked linear assignment algorithm," in IEE Proceedings - Radar, Sonar and Navigation, 153(1), pp. 13-22, 16 Feb. 2006.

[8] Y. Yao, J. Shang and Q. Wang, "Optimised multi-hypothesis tracking algorithm based on the two-dimensional constraints and manoeuvre detection," in The Journal of Engineering, 2019(21), pp. 7677-7682, 2019.

[9] L. Wu, Y. Xu, Q. Wang, F. Wang and Z. Xu, "Mapping Global Shipping Density from AIS Data," Journal of Navigation, 2017, 70(1), pp. 67-81

[10] D. Musicki and R. Evans, "Joint Integrated Probabilistic Data Association - JIPDA," Proceedings of the Fifth International Conference on Information Fusion, Annapolis, MD, USA, 2002, pp. 1120-1125 vol.2.

[11] Q. Li, J. Sun and W. Sun, "An efficient multiple hypothesis tracker using max product belief propagation," 2017 20th International Conference on Information Fusion (Fusion), Xi'an.

[12] C. Chong, S. Mori and D. B. Reid, "Forty Years of Multiple Hypothesis Tracking - A Review of Key Developments," 2018 21st International Conference on Information Fusion (FUSION), Cambridge, 2018, pp. 452-459.

[13] J. Sun, Q. Li, X. Zhang and W. Sun, "An Efficient Implementation of Track-Oriented Multiple Hypothesis Tracker Using Graphical Model Approaches," Mathematical Problems in Engineering, 2017.

[14] Y. Zhang, Y. Yang, S. Wei and J. Wang, "Fast data association approaches for multi-target tracking," CIE International Conference on Radar (RADAR), Guangzhou, 2016.

[15] F. Meyer, P. Braca, P. Willett and F. Hlawatsch, "A Scalable Algorithm for Tracking an Unknown Number of Targets Using Multiple Sensors," in IEEE Transactions on Signal Processing, 65(13), pp. 3478-3493, 2017.

[16] M. Motro and J. Ghosh, "Scaling Data Association for Hypothesis-Oriented MHT," 22th International Conference on Information Fusion (FUSION), Ottawa, ON, Canada, 2019.

[17] A. Amditis, G. Thomaidis, P. Maroudis, P. Lytrivis and G. Karaseitanidis, "Multiple Hypothesis Tracking Implementation," Laser Scaner Technology, 2012.

[18] M. E. Liggins, Chee-Yee Chong, I. Kadar, M. G. Alford, V. Vannicola and S. Thomopoulos, "Distributed fusion architectures and algorithms for target tracking," in Proceedings of the IEEE, 85(1), pp. 95-107, 1997.

[19] S. Coraluppi, C. Carthel, B. Zimmerman, T. Allen, J. Douglas and J. Muka, "Multi-stage MHT with airborne and ground sensors," IEEE Aerospace Conference, Big Sky, MT, 2018.

[20] M. E. Liggins, Chee-Yee Chong, I. Kadar, M. G. Alford, V. Vannicola and S. Thomopoulos, "Distributed fusion architectures and algorithms for target tracking," in Proceedings of the IEEE, vol. 85, no. 1, pp. 95-107, Jan. 1997.

[21] S. Coraluppi et al., "Distributed MHT with active and passive sensors," 2015 18th International Conference on Information Fusion (Fusion), Washington, DC, 2015, pp. 2065-2072.

[22] K. Yoon, Y. Song, M. Jeon. "Multiple hypothesis tracking algorithm for multi-target multi-camera tracking with disjoint views," IET Image Processing, 12(7), pp. 1175-1184, 2018.

[23] L. Wu, ICTTargets: Multisource Target Data Simulator. https://gitee.com/iOceanPlus/open_multisource_targetdata_simulator, 2020 (Accessed 24 March 2020).

[24] Pivotal Software Inc., Home page of RabbitMQ. http://www.rabbitmq.com/, 2020 (Accessed 24 March 2020).

[25] Google, Protocol Buffers, https://developers.google.cn/protocol-buffers/, 2020 (Accessed 24 March 2020).

[26] R. Danchick and G. E. Newnam, "Reformulating Reid's MHT method with generalised Murty K-best ranked linear assignment algorithm," in IEE Proceedings - Radar, Sonar and Navigation, 153(1), pp. 13-22, 2006.

[27] J. Olofsson, E. Brekke, T. I. Fossen and T. A. Johansen, "Spatially indexed clustering for scalable tracking of remotely sensed drift ice," IEEE Aerospace Conference, Big Sky, MT, 2017.

[28] D. M. Antunes, D. M. Matos and J. Gaspar, "A Library for Implementing the Multiple Hypothesis Tracking Algorithm", Computer Science, 2011.