Progress In Electromagnetics Research, Vol. 122, 137–154, 2012

MAPPING THE SBR AND TW-ILDCs TO HETEROGENEOUS CPU-GPU ARCHITECTURE FOR FAST COMPUTATION OF ELECTROMAGNETIC SCATTERING

P. C. Gao, Y. B. Tao, Z. H. Bai, and H. Lin*

State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou 310058, P. R. China

Abstract—In this paper, the shooting and bouncing ray (SBR) method in combination with the truncated wedge incremental length diffraction coefficients (TW-ILDCs) is implemented on the heterogeneous CPU-GPU architecture to effectively solve the electromagnetic scattering problems. The SBR is mapped to the GPU because numerous independent ray tubes can make full use of the massively parallel resources on the GPU, while the TW-ILDCs are implemented on the CPU since they require complex and high-precision numerical calculation to get the accurate result. As the computation times of neighboring aspect angles are similar, a dynamic load adjustment method is presented to achieve reasonable load balancing between the CPU and GPU. Applications, including the radar cross section (RCS) prediction and inverse synthetic aperture radar (ISAR) imaging, demonstrate that the proposed method can greatly improve the computational efficiency by fully utilizing all available resources of the heterogeneous system.

1. INTRODUCTION

The development of accurate and efficient high-frequency asymptotic algorithms for electromagnetic scattering by large complex objects has been of interest for many years. The shooting and bouncing ray (SBR) method [1] is one of the most popular and novel high-frequency methods. However, the SBR is very time-consuming for extremely electrically large targets because the density of ray tubes should be greater than ten rays per wavelength to ensure the convergence of results. In order to reduce the computation time, many acceleration

Received 23 September 2011, Accepted 2 November 2011, Scheduled 16 November 2011
* Corresponding author: Hai Lin (lin@cad.zju.edu.cn).

techniques have been developed. The octree and kdtree [2, 3] were applied to decrease the number of intersection tests each ray, and the latter has been proved as the best acceleration structure for ray tracing of static scenes in computer graphics [4]. On the other hand, the multiresolution grid algorithm was proposed to reduce the initial number of ray tubes in the SBR [5–7]. The angular division algorithm was employed to reduce both the number of initial rays and operations for intersection tests [8, 9]. Recently, graphics processing unit (GPU) has become the general computational resources due to the programmability and the great success of GPGPU (general-purpose processing on the GPU) [10]. The tremendous computational resources on the GPU have already been utilized to accelerate the SBR [7, 11]. Although the efficiency of the SBR has been significantly increased, its accuracy is restricted because it only takes into account multiple bounces. Thus, many efforts have concentrated upon the high order diffraction phenomena, especially, the edge-diffraction effect. In contrast to other common diffraction methods, the truncated wedge incremental length diffraction coefficients (TW-ILDCs) [12] are well behaved for all directions of incidence and observation and take a finite value for zero strip length.

Over the past few years, with the rapid development of the hardware, parallel computing has become an evolving research field. Several approaches (e.g., [13–17]) were implemented on the CPU clusters, while others (e.g., [18–20]) leveraged the parallel resources on the GPU. Nowadays commodity computers are usually equipped with high-performance graphics hardware. The CPU-GPU cooperative computing, instead of the CPU or GPU only, makes them as a heterogeneous system in modern computing [21]. The applications that use CPU or GPU only waste substantial available resources. Some works have been reported to success in performing heterogeneous computing on the hybrid CPU-GPU architecture, such as high-quality rendering [22] and fast multipole method (FMM) [23]. The key challenge is how to achieve good load balance between the CPU and GPU, i.e., dispatching the most suitable work to the CPU and GPU to maximize the use of all computational resources.

In this paper, we utilize the GPU to accelerate the SBR, and use the CPU to execute the TW-ILDCs concurrently. The load is reasonably distributed to the CPU and GPU according to the execution time of the previous angle to achieve load balancing on the heterogeneous CPU-GPU architecture. The scattered fields are post-processed for the radar cross section (RCS) prediction and inverse synthetic aperture radar (ISAR) imaging. In this way, the proposed method ensures the accuracy and makes the best use of all potential

computational power simultaneously.

2. MAPPING THE SBR AND TW-ILDCs ON HETEROGENEOUS CPU-GPU ARCHITECTURE

2.1. Shooting and Bouncing Ray Method

The procedure of the SBR involves two steps: ray tube tracing and electromagnetic computing. Firstly, a set of parallel ray tubes modeling the incident plane wave are shot to the target along the incident direction. Each corner ray is recursively traced to obtain the path according to Snell's law. Then the intersections with the target of the central rays of ray tubes are also evaluated via ray tracing, and the fields of them are determined by the theory of geometrical optics (GO). Finally, the physical optics (PO) integral is carried out on the four-sided polygon modeled by the exit positions of the ray tube to calculate the scattered field contribution of this ray tube, and the formula at an observation point (r, θ, ϕ) is given:

$$\mathbf{E}(r,\theta,\phi) \approx \frac{e^{-jkr}}{r}(\hat{\theta}E_\theta + \hat{\phi}E_\phi). \tag{1}$$

The (E_θ, E_ϕ) can be represented as the exit field (E, H) of the four-sided polygon S:

$$\begin{bmatrix} E_\theta \\ E_\phi \end{bmatrix} = \left(\frac{jk}{2\pi}\right)\iint_S e^{j\mathbf{k}\cdot\mathbf{r}'}\left\{\begin{bmatrix} -\hat{\phi} \\ \hat{\theta} \end{bmatrix} \times \mathbf{E}(r')f_e + Z_0\begin{bmatrix} \hat{\theta} \\ \hat{\phi} \end{bmatrix} \times \mathbf{H}(r')f_h\right\} \cdot \hat{\mathbf{n}}dx'dy'. \tag{2}$$

Equation (2) contains three forms with different values of the coefficients f_e and f_h. As indicated in [24], the EH formula provides the best approximation for the PO induced currents. The GPU-based SBR [7, 11] is used in our implementation.

2.2. Truncated Wedge Incremental Length Diffraction Coefficients

Truncated wedge incremental length diffraction coefficients (TW-ILDCs) calculate the fringe wave field from a line integral along the illuminated part of the strips located on a canonical wedge. The truncated equivalent edge currents (EECs) are expressed by the magnetic current M_T and the electric current I_T, and the formula of the electric fringe wave field is given:

$$\mathbf{E} = jk\int_l \left(ZI_T\hat{s}\times\left(\hat{s}\times\hat{t}\right) + M_T\hat{s}\times\hat{t}\right)\frac{\exp\left(-jkr\right)}{4\pi r}dl, \tag{3}$$

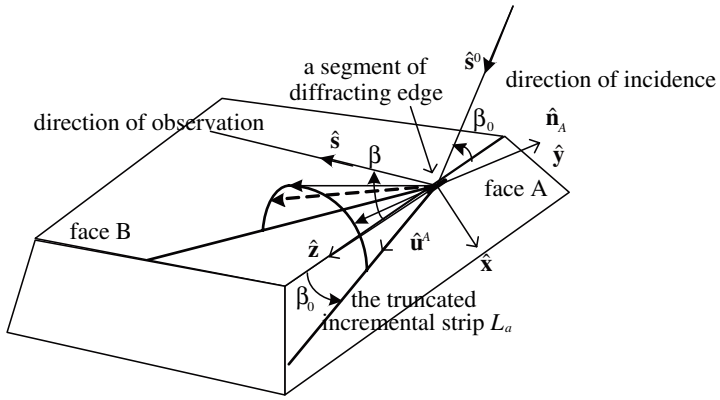

Figure 1. Three-dimensional view of the wedge calculated with the TW-ILDCs.

where $\hat{s}$ is the unit vector to the observation point, and $\hat{t}$ is the edge unit tangent vector.

As illustrated in Figure 1, the previous infinite incremental strip is truncated and it extends from the edge to the boundary of the wedge face A. The truncated equivalent edge currents are represented as the differences between the untruncated EECs and the correction EECs, and their detailed expressions can be found in [12].

In our implementation, all of the triangles are processed to construct a data structure that associates each edge with its two adjoining faces (except boundary edges which have only one adjoining face). Then, the edges are chosen as the diffracting edges according to the criteria that whether the interior wedge angle of the edge is less than the user-defined maximum angle. As the edge may be visible partially, the edge is split into small segments to ensure the accuracy of the result. From the midpoint of each segment, the truncated incremental strip is traced along the intersection of the Keller cone and the wedge face. The visibility and the truncated segments of the edges described above are recalculated for each incident angle because they vary with the angle of incidence.

2.3. Mapping Strategy

As illustrated in Figure 2, the GPU is connected to the CPU via the PCI express bus in the mainstream configuration of a computing node. It is desirable to maximize parallel execution at a higher level by explicitly exposing concurrency between the CPU and GPU for the applications on the heterogeneous CPU-GPU architecture. Thus, in order to achieve optimal performance, we map the SBR and the TW-

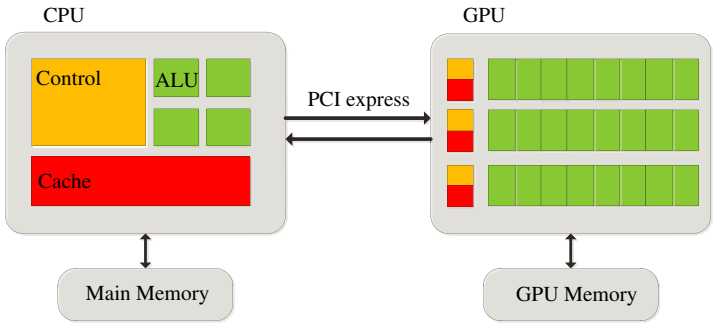

Figure 2. The CPU in combination with the GPU composes the heterogeneous system in this paper. The GPU is designed for compute-intensive, highly parallel computation, therefore more resources are devoted to data processing. On the other hand, the CPU is well-suited for data caching and flow control.

ILDCs to the CPU and GPU, respectively. The mapping strategy is designed according to the characteristics of both the algorithms and hardware.

The modern GPU is massively parallel and follows a single program multiple data (SPMD) programming model. All threads on the GPU run the same program, referred to as a kernel. It requires a large number of threads at one time to maximize opportunities to hide the memory latency for optimal performance. As ray tubes are independent of each other, the SBR can be easily changed into the multi-threaded fashion. Moreover, the density of ray tubes should be greater than ten rays per wavelength to ensure the accuracy. For electrically large targets, the number of ray tubes can reach a million or even more. Thus, the SBR can maintain sufficient numbers of threads to hide the memory latency.

The edges are split into small segments according to the frequency in our implementation. The edges of two aircrafts shown in Section 3.2 are divided into about 14000 and 40000 segments at 10 GHz, respectively. However, for the number of cores of the GPU (e.g., 448 CUDA cores on GeForce GTX 470), the number of segments is still not large enough to fully hide the memory latency as compared to the number of ray tubes, which are about 30 and 17 millions for the two aircrafts at 10 GHz, respectively. In contrast to the SBR, the TW-ILDCs can not make full use of the massively parallel resources on the GPU.

To improve the overall performance of the GPU, the data transfers between the CPU and GPU should be minimized because those

transfers have much lower bandwidth than the internal GPU data transfer. As compared to the TW-ILDCs, the SBR has less data to be transferred from the CPU to the GPU because it does not need the data structure recording the edge-to-face associations.

Although double precision is supported on the current GPU, its speed is limited to 1/8 of single precision speed for most applications. It is suggested to use single precision for better performance when not affecting the accuracy. The edge-diffraction effect is more sensitive to round off than multiple bounces. In addition, complex Fresnel integrals are required for the TW-ILDCs, but the relatively simple Gordon's contour integration [25] can be used to calculate the PO integral in the SBR. Thus, the TW-ILDCs are suitable to be implemented on the CPU with double precision, while single precision should be used to achieve optimal performance without much loss of accuracy in the GPU-based SBR. Because all of the reasons discussed above, we map the SBR to the GPU and the TW-ILDCs to the CPU. The CPU and

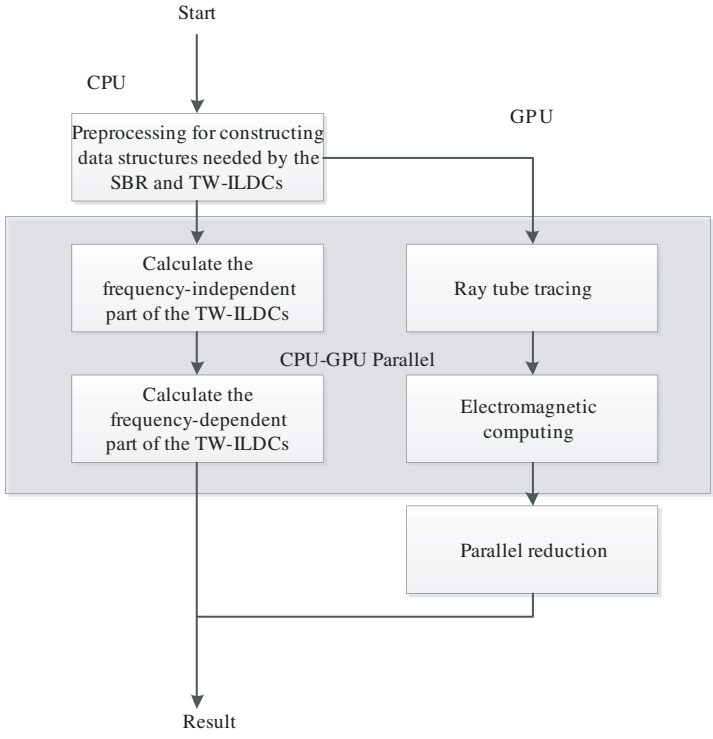

Figure 3. Flow chart of the SBR and TW-ILDCs on the heterogeneous architecture. Steps in light gray are concurrently executed on the CPU and GPU.

GPU are assigned with the most appropriate works they are good at by using our mapping strategy.

The procedure of our heterogeneous version of the SBR + TW-ILDCs is shown in Figure 3. The GPU-based SBR implemented with CUDA is divided into three stages, and each stage corresponds to one kernel on CUDA. Firstly, the corner rays of ray tubes are launched in parallel and recursively traced to obtain the intersections with the target. The second stage is firstly to check the validity of the ray tubes, then trace the central rays of the valid ray tubes and calculate the reflected fields with GO, finally carry out the PO integrals to determine the scattered fields of the ray tubes. In the last stage, the parallel reduction is employed to sum the scattered fields of ray tubes. As long as the virtual aperture is divided according to the highest frequency of a certain frequency bandwidth, the first step of the SBR can be thought to be independent of frequency. Similarly, the calculation of the TW-ILDCs also has the frequency-independent part, which is only related to the geometry of the wedge and the directions of incidence and observation. By using asynchronous function calls on CUDA, the control is returned to the CPU immediately after the kernel launch, and

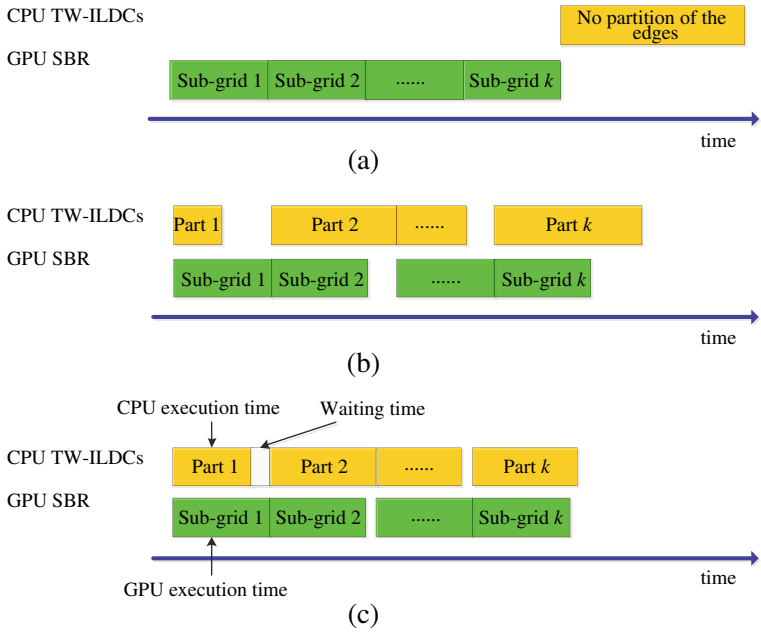

Figure 4. Three implementations of the SBR + TW-ILDCs. (a) Serial version. (b) Heterogeneous version with the static method. (c) Heterogeneous version with the dynamic method.

then the TW-ILDCs are executed on the CPU. Thus, the corresponding steps of the TW-ILDCs and the SBR are overlapped, for example, the frequency-independent TW-ILDCs calculation and ray tube tracing.

2.4. Load Balancing between the CPU and GPU

Due to the limited memory on the GPU, the grid on the virtual aperture is partitioned into sub-grids [7, 11], which are 2048×2048 in our implementation.

As illustrated in Figure 4(a), the traditionally serial implementation does not overlap the computation between the CPU and GPU. To improve the hardware utilization, the workload of the TW-ILDCs is also divided into a few parts to overlap the computation of sub-grids on the GPU. A straightforward method is to divide the edges into parts according to the size percentage of each sub-grid on the virtual aperture. As a result, each sub-grid and its corresponding part have the same percentage of the workload. We call the approach the static method.

However, as shown in Figure 4(b), the same proportion of workload does not mean the same computation time of the SBR and the TW-ILDCs, so the load imbalance still exists. Using the observation that the computation times of neighboring aspect angles are almost the same, we dynamically calculate the size of each part of the TW-ILDCs based on the execution time of each sub-grid at the previous angle. Initially, the workload is partitioned with the static method, and then the number of edges in each part is dynamically adjusted as follows:

$$n_k^i = \frac{t_k^{i-1}}{\sum_{m=1}^{M} \tau_m^{i-1}/N}, \tag{4}$$

where n_k^i is the number of edges of the part k of the TW-ILDCs at the aspect angle i, and N represents the total number of edges at the previous angle $i-1$. t_k^{i-1} is the computation time of the sub-grid k at the angle $i-1$, and $\sum_{m=1}^{M} \tau_m^{i-1}$ means the total execution time of the TW-ILDCs at the previous angle $i-1$. Thus, the denominator of Equation (4) represents the average computation time of one edge. The loads on the CPU and GPU can be reasonably balanced by using our dynamic load adjustment method, as shown in Figure 4(c).

The CPU or GPU may be unstable due to some reasons (e.g., excessive heat) at some aspect angles. In this situation, the performance may be affected at the following angle. However, it will only affect the computation at the next one angle and the bad influence

will not propagate to the other angles. As shown in Figure 4(c), the CPU and GPU times used in Equation (4) do not include the waiting time due to the CPU-GPU synchronization. Although the total computation time increases, as long as the CPU and GPU resume normal operation, the proposed method will achieve good load balance at the following angles again. As a result, the bad load balance may occur at only few angles, and the proposed dynamic load adjustment method can achieve reasonable load balance at most angles.

2.5. Implementation Details

The pseudocode of the proposed method is elaborated in Algorithm 1. There are three main differences between the traditionally serial implementation and the proposed method. First, in order to return the control to the CPU immediately after the kernel launch, the GPU kernel calls are changed to asynchronous function calls. Second, the GPU's progress is monitored to synchronize the CPU with the GPU. Third, the dynamic load adjustment method is designed to achieve good load balance between the CPU and GPU. Note that we divide the virtual aperture according to the highest frequency of a certain frequency bandwidth, i.e., the computation time of the sub-grid at any frequency is the same. Although several differences exist, the major parts (e.g., the implementation of the GPU-based SBR and TW-ILDCs) of concurrent version are almost the same as the serial version. Thus, the traditional method can be upgraded to the proposed method with relatively fewer modifications.

3. NUMERICAL RESULTS

In order to evaluate the accuracy, the efficiency and the capability of the proposed heterogeneous version of the SBR + TW-ILDCs, several experiments are performed on Intel Core i5 2.8 GHz CPU and a NVIDIA GeForce 470 GTX with CUDA Toolkit 3.2. The targets are perfect electric conductor (PEC) in this paper.

3.1. RCS Prediction

The monostatic RCS of a ship at 10 GHz is calculated with the GPU-based SBR, our heterogeneous algorithm, and the MLFMM, respectively. The result comparison is used to demonstrate the edge-diffraction effect is necessary for accurately predicting electromagnetic scattering. The size of the ship is $0.9\,\mathrm{m} \times 0.2\,\mathrm{m} \times 0.2\,\mathrm{m}$, and its geometrical model is shown in Figure 5(a). The incident direction is rotated around the Y axis from 0° to 360° with the interval of 1°.

Algorithm 1 The SBR and TW-ILDCs on heterogeneous CPU-GPU architecture

```
Initialization:
  edgeNum ⇐ 0
  gpuTime[gridNum] ⇐ new array
  cpuTime[freqNum] ⇐ new array
  modelPreprocessing()
Calculate scattering field:
  for each elevation angle i do
    for each azimuth angle j do
      getDiffractionEdge(edgeNum)
      for each sub-grid k do
        \\frequency-independent part of both the SBR and TW-ILDCs
        \\asynchronous GPU kernel call for GPU-based SBR ray tracing
        rayTracing()
        \\CPU-GPU synchronization
        while isGPUBusy() do
          \\calculate the frequency-independent part of the TW-ILDCs with CPU
          preCalculate()
        end while
        \\frequency-dependent part of both the SBR and TW-ILDCs
        for each frequency f in a bandwidth do
          \\asynchronous GPU kernel call for GPU-based SBR electromagnetic computing
          startTimer(time1)
          electromagneticComputing()
          stopTimer(time1)
          \\CPU-GPU synchronization
          while isGPUBusy() do
            dynamicLoadAdjustment(gpuTime[k]^previous, cpuTime[f]^previous, edgeNum^previous)
            \\calculate the frequency-dependent part of the TW-ILDCs with CPU
            startTimer(time2)
            calculate()
            stopTimer(time2)
          end while
          gpuTime[k] ⇐ time1
          cpuTime[f] ⇐ cpuTime[f] + time2
          \\sum the scattered fields of a sub-grid at frequency f
          parallelReduction()
        end for
      end for
    end for
  end for
```

As can be seen clearly from Figure 5, there is a good agreement between the GPU-based SBR result and the MLFMM result when the values of RCS are relatively large, as multiple bounces dominate at those angles. However, the edge-diffraction effect, as the secondary dominant scattering mechanism, can not be ignored in the case of weak scattering. Thus, the result of the SBR + TW-ILDCs is more accurate than the SBR result, especially in the incident angles after 180°.

In order to demonstrate the capability of the proposed method for scattering by extremely electrically large targets, the monostatic RCS of aircraft A illustrated in Figure 7(a) at 100 GHz is calculated. In our experiment, aircraft A is 3920λ, and it is located on the x-y plane with

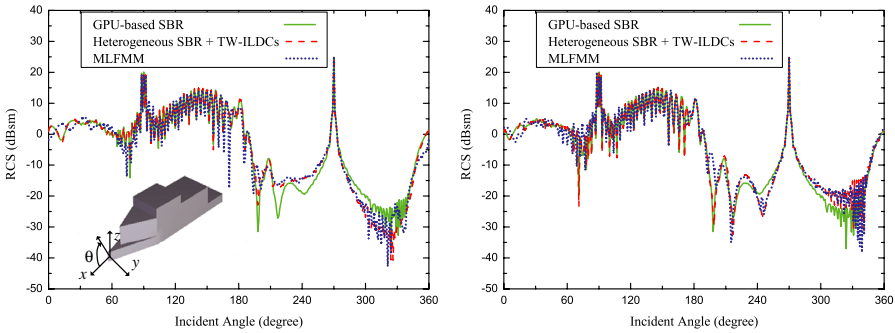

Figure 5. The comparison of our heterogeneous algorithm result, the GPU-based SBR result and the MLFMM result for the ship at 10 GHz. (a) VV-polarization result. (b) HH-polarization result.

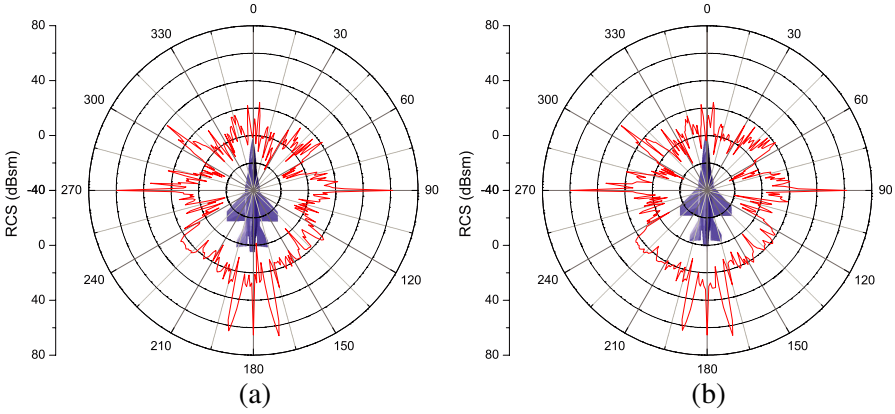

Figure 6. The monostatic RCS of airplane A at 100 GHz. (a) VV-polarization result. (b) HH-polarization result.

its nose directed towards the x-axis. Figure 6 presents the RCS result of aircraft A on the x-y plane, and the computation time is 1.83 hours. Most recently, the parallel implementation of the MLFMM can solve the problems including scatterers larger than 1000λ. In [26], the sizes of the NASA almond and the Flamme are 1177λ and 1240λ, respectively. And the scattering problem is solved on the Nehalem cluster, which consists of 64 computing nodes, and each node has 24 GB of memory and two Intel Xeon Nehalem quad-core processors with 2.67 GHz clock rates. The computation times are 11 hours and 17 hours for the NASA almond and the Flamme, respectively. It proves that the proposed method has the feasibility of calculating the scattering by the extremely electrically large and complex targets.

3.2. ISAR Imaging

ISAR imaging widely used in automatic target recognition (ATR) is also used to demonstrate the validity and efficiency of the proposed method. It is very efficient to generate the ISAR image with the direct image domain formation [27, 28]. However, we apply the traditional two-step approach, which computes scattered fields and applies the inverse Fourier transform. The reason is that we aim at developing a general simulation tool for electromagnetic scattering problems.

The ISAR images of two electrically large and complex aircrafts are investigated, and the incident directions are around the noses of them illustrated in Figure 7.

Figure 8 compares the GPU-based SBR result only considering the multiple-bounce with the result of the heterogeneous version of the SBR + TW-ILDCs of the aircraft A. The noises in both images are partially caused by the drawback of the SBR [29]. Apparent deviation can be observed at the locations marked with numbers due

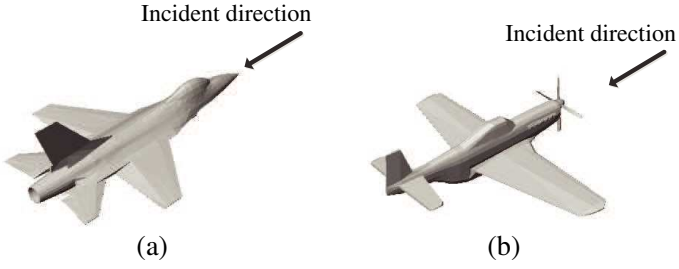

Figure 7. Two targets for ISAR imaging. (a) Aircraft A consists of 13050 triangular facets, and its size is 11.76 m × 7.4 m × 3.67 m. (b) Aircraft B has 20120 triangular facets and its geometry size is 14.12 m × 16.98 m × 4.5 m.

to the lack of edge diffraction. With the contribution of the trailing edge diffractions, ATR can be done more easily with Figure 8(b) than Figure 8(a). The same conclusion can be obtained from the Figure 9.

Compared with the GPU-based SBR, the proposed method

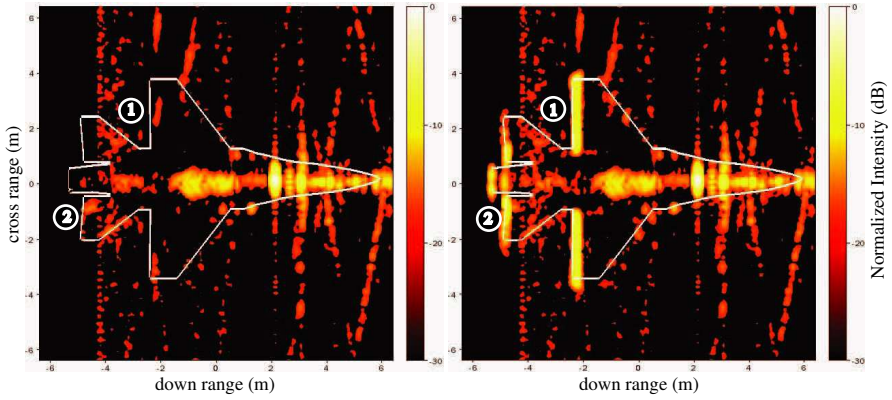

Figure 8. Comparison of ISAR images of the aircraft A at 10 GHz in vv-polarization, $\Delta f = 1.5$ GHz, Hamming window, $\theta = 80°$, $\delta_r = \delta_\varphi = 0.1$ m: (a) Image generated with the GPU-based SBR. (b) Image obtained by the heterogeneous SBR + TW-ILDCs.

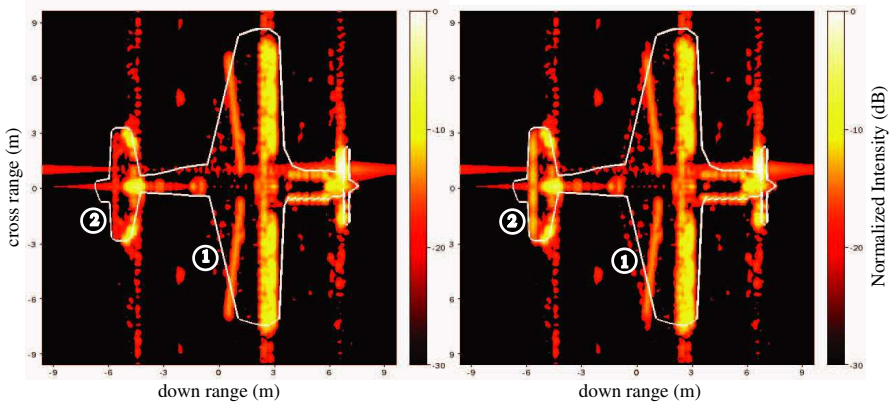

Figure 9. Comparison of ISAR images of the aircraft B at 10 GHz in vv-polarization, $\Delta f = 1.0$ GHz, Hamming window, $\theta = 70°$, $\delta_r = \delta_\varphi = 0.15$ m: (a) Image generated with the GPU-based SBR. (b) Image obtained by the heterogeneous SBR + TW-ILDCs.

significantly increases the accuracy at the cost of efficiency decreased slightly. The computation time of each step of the proposed method and the execution time of the GPU-based SBR are shown in Table 1. The ray tube tracing only need to be executed once at the highest frequency of a certain bandwidth, and the first part of the TW-ILDCs is similar. As can be seen from Table 1, the main parallel region between the CPU and GPU are the electromagnetic computing stage of the SBR and the frequency-dependent part of the TW-ILDCs. The efficiency decreases by only 4.6% for aircraft A as compared to the GPU-based SBR, however, the value is 17.2% for the aircraft B. The ratio is much higher because the CPU dominates the concurrent execution time, and therefore the computation time of the TW-ILDCs can not be overlapped very well with the GPU-based SBR.

The previous algorithms usually neglect the CPU when they utilize the parallel resources on the GPU for acceleration, and therefore the CPU is idle during the GPU-based computations. The serial version that the GPU-based SBR and the TW-ILDCs do not concurrently execute is compared with our heterogeneous version. The computation times of the serial version and the proposed approach are available in Table 2. The proposed algorithm has other steps besides the five steps, for example, the model preprocessing step. Thus, the maximum speedup ratio is infinitely close to 2, and it can be achieved by completely overlapping the execution time of the CPU and GPU. The performance improvement using both the CPU and GPU can reach up to about 40% as compared to the serial version for both the aircraft A and B. Taking aircraft A as an example, although the total execution time of the GPU is more than the CPU, the CPU takes more time at 24% of aspect angles. As a result, it is hard to optimally balance the load between the CPU and GPU. However, the performance is improved much more by using our dynamic method than the static method (e.g., the improvement is increased from 16.6% to 38.5% for

Table 1. The computation time (seconds) of each step of our heterogeneous version of the SBR + TW-ILDCs, and the execution time of the GPU-based SBR. Step 1, step 2 and step 3 represent the ray tube tracing, electromagnetic computing and parallel reduction of the GPU-based SBR. The frequency-independent part and the frequency-dependent part of the TW-ILDCs are step 4 and 5, respectively.

Target	Parallel		Parallel		Step 3	Total Time	SBR
	Step 1	Step 4	Step 2	Step 5			
Aircraft A	18.2	7.9	1046.8	906	219.3	1371.4	1311
Aircraft B	21.5	17.8	2153.5	2410.4	118.4	2699.9	2302.7

Table 2. The computation time (seconds) comparison of the serial version and the heterogeneous version with the static method and the dynamic method for aircraft A and B.

Method	Aircraft A	Aircraft B
Serial version	2230.8	4712.2
Heterogeneous version (the static method)	1859.6	3311.3
Heterogeneous version (the dynamic method)	1371.4	2699.9
Performance improvement (static\dynamic)	16.6%\38.5%	29.7%\43%

aircraft A). The dynamic method ensures excellent load balance. The ISAR imaging is very time-consuming for electrically large and complex targets, and thus about 40% or more computation time reduction saves a large amount of time, which are 859.4 and 2012.3 seconds for aircraft A and B, respectively.

As we all know, the MLFMM as a fast algorithm has been widely used to solve many large-scale problems. However, as described in Section 3.1, the parallelization of the MLFMM still needs a large amount of time to calculate the RCS of electrically large targets. It will be beyond the capability of the MLFMM implemented on relatively small and inexpensive clusters when intensive scattering calculations of electrically large targets are required (e.g., building the database for ATR with large amount of ISAR images). The proposed method makes full use of all available resources on a single computing node by utilizing the concurrency of the CPU and GPU. It can obtain a reasonably accurate ISAR image of the real-life aircraft within an hour. For a cluster currently composed of heterogeneous nodes, the proposed algorithm can be efficiently parallelized by distributing the computations of different angles to each node, and the execution time will decrease to several minutes. Thus, utilizing the clusters to further increase the capacity of the proposed algorithm will be our future work.

4. CONCLUSION

The SBR and the TW-ILDCs are combined and implemented on the heterogeneous CPU-GPU architecture. The main contribution of this paper is the use of the neglected resource (CPU) in heterogeneous CPU-GPU environments, maximizing hardware utilization to provide an efficient tool for electromagnetic scattering problems. The heterogeneous algorithm maximally utilizes the parallel resources of the GPU to accelerate the SBR. It also enables efficient double precision computations to ensure the accuracy of the TW-ILDCs

and dynamically adjusts the load of the CPU to achieve good load balancing. The proposed method not only provides higher accuracy with relatively little efficiency penalty as compared to the GPU-based SBR, but also improves the performance up to about 40% in comparison with the traditionally serial implementation. Compared with the parallel MLFMM algorithm, the heterogeneous method also proves its capability of solving large-scale scattering problems.

ACKNOWLEDGMENT

The authors would like to thank Prof. T.-J. Cui from South East University for providing the MLFMM method used in this paper. This work was partially supported by NFS of China (No. 61171035).

REFERENCES

1. Ling, H., R. C. Chou, and S. W. Lee, "Shooting and bouncing rays: Calculating the RCS of an arbitrarily shaped cavity," *IEEE Trans. Antennas Propag.*, Vol. 37, No. 2, 194–205, 1989.
2. Jin, K.-S., T.-I. Suh, S.-H. Suk, B.-C. Kim, and H.-T. Kim, "Fast ray tracing using a space-division algorithm for RCS prediction," *Journal of Electromagnetic Waves and Applications*, Vol. 20, No. 1, 119–126, 2006.
3. Tao, Y. B., H. Lin, and H. J. Bao, "Kd-tree based fast ray tracing for RCS prediction," *Progress In Electromagnetics Research*, Vol. 81, 329–341, 2008.
4. Havran, V., "Heuristic ray shooting algorithms," Ph.D. Dissertation, Univ. Czech Technical, Prague, 2000.
5. Suk, S. H., T. I. Seo, H. S. Park, and H. T. Kim, "Multiresolution grid algorithm in the SBR and its application to the RCS calculation," *Microw. Opt. Technol. Lett.*, Vol. 29, No. 6, 394–397, 2001.
6. Bang, J.-K., B.-C. Kim, S.-H. Suk, K.-S. Jin, and H.-T. Kim, "Time consumption reduction of ray tracing for RCS prediction using efficient grid division and space division algorithms," *Journal of Electromagnetic Waves and Applications*, Vol. 21, No. 6, 829–840, 2007.
7. Gao, P. C., Y. B. Tao, and H. Lin, "Fast RCS prediction using multiresolution shooting and bouncing ray method on the GPU," *Progress In Electromagnetics Research*, Vol. 107, 187–202, 2010.
8. Kim, B.-C., K.-K. Park, and H.-T. Kim, "Efficient RCS prediction method using angular division algorithm," *Journal of*

Electromagnetic Waves and Applications, Vol. 23, No. 1, 65–74, 2009.

9. Park, K.-K. and H.-T. Kim, "RCS prediction acceleration and reduction of table size for the angular division algorithm," *Journal of Electromagnetic Waves and Applications*, Vol. 23, Nos. 11–12, 1657–1664, 2009.
10. Owens, J. D., D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell, "A survey of general-purpose computation on graphics hardware," *Comput. Graphics Forum.*, Vol. 26, No. 1, 80–113, 2007.
11. Tao, Y. B., H. Lin, and H. J. Bao, "GPU-based shooting and bouncing ray method for fast RCS prediction," *IEEE Trans. Antennas Propag.*, Vol. 58, No. 2, 494–502, 2010.
12. Johansen, P. M., "Uniform physical theory of diffraction equivalent edge currents for truncated wedge strips," *IEEE Trans. Antennas Propag.*, Vol. 44, No. 7, 989–995, 1996.
13. Yang, M.-L. and X.-Q. Sheng, "Parallel high-order FE-BI-MLFMA for scattering by large and deep coated cavities loaded with obstacles," *Journal of Electromagnetic Waves and Applications*, Vol. 23, No. 13, 1813–1823, 2009.
14. Guo, L.-X., A.-Q. Wang, and J. Ma, "Study on EM scattering from 2-D target above 1-D large scale rough surface with low grazing incidence by parallel MoM based on PC clusters," *Progress In Electromagnetics Research*, Vol. 89, 149–166, 2009.
15. Taboada, J. M., M. G. Araujo, J. M. Bertolo, L. Landesa, F. Obelleiro, and J. L. Rodriguez, "MLFMA-FFT parallel algorithm for the solution of large-scale problems in electromagnetics," *Progress In Electromagnetics Research*, Vol. 105, 15–30, 2010.
16. Zhao, X.-W., Y. Zhang, H.-W. Zhang, D. Garcia-Donoro, S.-W. Ting, T. K. Sarkar, and C.-H. Liang, "Parallel MoM-PO method with out-of-core technique for analysis of complex arrays on electrically large platforms," *Progress In Electromagnetics Research*, Vol. 108, 1–21, 2010.
17. Garcia-Donoro, D., I. Martinez-Fernandez, L. E. Garcia-Castillo, Y. Zhang, and T. K. Sarkar, "RCS computation using a parallel in-core and out-of-core direct solver," *Progress In Electromagnetics Research*, Vol. 118, 505–525, 2011.
18. Zainud-Deen, S. H., E. Hassan, M. S. Ibrahim, K. H. Awadalla, and A. Z. Botros, "Electromagnetic scattering using GPU-based finite difference frequency domain method," *Progress In Electromagnetics Research B*, Vol. 16, 351–369, 2009.

19. Jiang, W.-Q., M. Zhang, and Y. Wang, "CUDA-based radiative transfer method with application to the EM scattering from a two-layer canopy model," *Journal of Electromagnetic Waves and Applications*, Vol. 24, Nos. 17–18, 2509–2521, 2010.
20. Xu, K., Z. Fan, D.-Z. Ding, and R.-S. Chen, "GPU accelerated unconditionally stable Crank-Nicolson FDTD method for the analysis of three-dimensional microwave circuits," *Progress In Electromagnetics Research*, Vol. 102, 381–395, 2010.
21. Brodtkorb, A. R., C. Dyken, T. R. Hagen, J. M. Hjelmervik, and O. O. Storaasli, "State-of-the-art in heterogeneous computing," *Journal of Scientific Programming*, Vol. 18, No. 1, 1–33, 2010.
22. Pajot, A., L. Barthe, M. Paulin, and P. Poulin, "Combinatorial bidirectional path-tracing for efficient hybrid CPU/GPU rendering," *Computer Graphics Forum.*, Vol. 30, No. 2, 315–324, 2011.
23. Hu, Q., N. A. Gumerov, and R. Duraswami, "Scalable fast multipole methods on distributed heterogeneous architectures," *SC'11 Proceedings of the 2011 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011.
24. Baldauf, J., S. W. Lee, L. Lin, S. K. Jeng, S. M. Scarborough, and C. L. Yu, "High frequency scattering from trihedral corner reflectors and other benchmark targets: SBR vs. experiments," *IEEE Trans. Antennas Propag.*, Vol. 39, No. 9, 1345–1351, 1991.
25. Gordon, W. B., "Far-field approximation to the Kirchhoff-Helmhotlz representation of scattered fields," *IEEE Trans. Antennas Propag.*, Vol. 23, 590–592, Jul. 1975.
26. Ergül, O. and L. Gürel, "Rigorous solutions of electromagnetic problems involving hundreds of millions of unknowns," *IEEE Antennas Propag. Mag.*, Vol. 53, No. 1, 18–26, 2011.
27. Buddendick, H. and T. F. Eibert, "Bistatic image formation from shooting and bouncing rays simulated current distributions," *Progress In Electromagnetics Research*, Vol. 119, 1–18, 2011.
28. He, X.-Y., X.-B. Wang, X. Zhou, B. Zhao, and T.-J. Cui, "Fast ISAR image simulation of targets at arbitrary aspect angles using a novel SBR method," *Progress In Electromagnetics Research B*, Vol. 28, 129–142, 2011.
29. Bhalla, R. and H. Ling, "Cross range streaks in ISAR images generated via the shooting and bouncing ray technique: Cause and solution," *IEEE Antennas Propag. Mag.*, Vol. 39, No. 2, 76–80, Apr. 1997.