IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 16, 2021, accepted April 1, 2021, date of publication April 5, 2021, date of current version April 13, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3070980

Optimized Real-Time MUSIC Algorithm With CPU-GPU Architecture

QINGHUA HUANG AND NAIDA LU
Key Laboratory of Specialty Fiber Optics and Optical Access Networks, Shanghai University, Shanghai 200444, China
Corresponding author: Qinghua Huang (qinghua@shu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61571279.

ABSTRACT Direction-of-arrival (DOA) estimation algorithm for uniform planar arrays has been applied in many fields. The multiple signal classification (MUSIC) algorithm has obvious advantage in high-resolution signal source estimation scenarios. However, the MUSIC algorithm has high computational costs, therefore it is hard to be used in real-time scenes. Many studies are dedicated to accelerating MUSIC algorithm by parallel hardware, especially by Graphics Processing Units (GPU). MUSIC algorithm based on Central Processing Unit (CPU) -GPU architecture acceleration is rarely investigated in previous literatures, and how well MUSIC Algorithm with CPU-GPU architecture could perform remains unknown. In this paper, we present and evaluate a model of search parallel MUSIC algorithm with CPU-GPU architecture. In the proposed model, the steering vector of each candidate incident signal and the corresponding value of 2D spatial pseudo-spectrum (SPS) function are sequentially calculated in a single core of the GPU, and the subsequent calculation of each elevation or azimuth is parallel in batches. Furthermore, in order to improve the peak search speed, we propose a new Coarse and Fine Traversal (CFT) peak search algorithm via CPU and a new parallel peak search algorithm based on GPU acceleration. Across strategy comparison, utilizing CPU-GPU architecture for processing, a 150-160x performance gain is achieved compared to using CPU only. Besides, the resolution of uniform planar arrays is also analyzed.

INDEX TERMS Direction-of-arrival (DOA) estimation, uniform planar arrays (UPA), high-resolution, real-time, CPU-GPU architecture.

I. INTRODUCTION

Direction-of-arrival (DOA) estimation has become a popular branch of array signal processing during the past four decades and has extensive applications in various fields, such as radar, sonar, mobile communications, and electrocardiograms [1]–[4]. The DOA estimation algorithms have in-depth study on accuracy and efficiency and have developed into many types of methods, such as beamforming methods [5], subspace decomposition methods, subspace fitting (SF) methods [6]–[8], and sparse representation (SR) methods [9], [10]. The beamforming algorithm reduces the effect of the co-channel on DOA estimation by using the phase-shifted reference signal that is obtained after interference suppression, accompanied by mediocre performance in high-resolution scenes. Maximum likelihood (ML) algorithm and SF algorithm are concise in theory and excellent in performance, yet they require complex and computationally intensive multi-dimensional nonlinear optimization [11]. The division of grid determines the estimation accuracy and computational complexity of SR algorithm. Subspace decomposition methods have been divided into two types in terms of processing methods. One is the noise subspace algorithm represented by multiple signal classification (MUSIC). And the other is the signal subspace algorithm represented by estimation of signal parameter via rotational invariance technique (ESPRIT). Algorithms represented by ESPRIT mainly include Toeplitz approximation approach (TAM), least-squares ESPRIT (LS-ESPRIT) and total least squares ESPRIT (TLS-ESPRIT) [12]–[15]. ESPRIT estimates the DOA by employing the rotation invariance among the signal subspace of each sub-array. The performance of these ESPRIT algorithms is substantially resembling, yet the performance of TLS-ESPRIT is somewhat better than other ESPRIT algorithms in the case of low signal-to-noise ratios (SNRs). The algorithms represented by MUSIC include MUSIC algorithm, Root-MUSIC algorithm, and minimum-norm methods (MNM) [16]–[18]. Root-MUSIC realizes

The associate editor coordinating the review of this manuscript and approving it for publication was Hasan S. Mir.

DOA estimation by solving the polynomial constructed using the orthogonality of steering vector and noise subspace. ESPRIT algorithm has a more diminutive computational complexity than the MUSIC algorithm and does not require SPS computation. However, the performance of ESPRIT is poor than MUSIC. The MUSIC algorithm has distinguished advantages in high-resolution applications. However, the 2D SPS computation of MUSIC algorithm makes it difficult to exploit in real-time scenes. Reduced-dimensional MUSIC algorithm was proposed for far-field signal source localization with sphere microphone arrays [19]. One-dimensional MUSIC-type algorithm (ODMUSIC) constructs a mapping matrix between the spherical harmonic function and Fourier series only depending on the array configuration. Two root polynomials in ODMUSIC are established to estimate the elevation and azimuth iteratively by exploiting the Vandermonde of Fourier series.

Moreover, GPU computation has developed extremely rapidly during the past two decades. GPU is a typical multi-core co-processor where 32 threads of a warp are controlled by a controller and process one instruction at the same time. In the single instruction multiple thread (SIMT) paradigm, threads are automatically grouped into 32-wide bundles called warps. Warps are the base unit used to schedule both computation on Arithmetic and Logic Units (ALUs) and memory accesses. Threads within the same warp follow the single instruction multiple data (SIMD) pattern, i.e., they are supposed to execute the same operation at a given clock cycle [20]. From the original image processing engine to the more powerful high-performance computing processor, it has been maturely applied to various fields [21], [22]. The wideband DOA estimation for uniform linear arrays (ULA) of 16 and 4 sensors has been parallelized, partitioned, mapped, and scheduled on Multi-Core, GPU, and IBM Cell BE processor [23]. Broadband underwater sound source estimation of MUSIC algorithm has been implemented based on GPU [24]. Four noise subspace DOA algorithms including Pisarenko Harmonic Decomposition, Eigen Vector, MUSIC, and MNM have been analyzed on CPU and GPU with uniform circle array (UCA) [25]. These aforementioned GPU-accelerated MUSIC algorithms have executed parallel of SPS computation.

However, existing GPU-based parallel MUSIC algorithms pay insufficient attention to the parallel calculation of candidate steering vector calculation, optimization of CPU-GPU memory transmission, and parallel optimization of peak search. The research on optimized computing, parallel computing and CPU-GPU acceleration has penetrated into many fields. In order to solve the problem of high computational cost and storage cost of large-scale 3D models, differential evolution and whale optimization algorithm are combined to calculate the simplified mesh model more effectively [26]. [27] proposes a GPU-based parallel tabu search algorithm that uses a single GPU kernel of compacting neighborhood and a kernel fusion strategy to reduce the amount of GPU global memory accesses. [28] proposes a model of vector parallel ant colony optimization for multi-core SIMD CPU architecture and accelerates the tour construction of each ant by vector instructions. GPU-based parallel ant colony optimization has been analyzed [29]. Blockwise Weighted Least Square Active Noise Control ANC algorithm for CPU-GPU Architecture has designed a real-time execution framework using GPU asynchronous parallel computing [30], [31]. [32] proposed a hybrid parallel algorithm for beamforming using the parallel mode of just-in-time (JIT) on multi-CPU and multi-GPU platform.

Inspired by optimized computing, parallel computing, CPU-GPU acceleration, real-time block diagram, and fine-grained parallelism [26]–[32], this paper presents an idea to accelerate a fully developed parallel MUSIC model with CPU-GPU architecture. In our algorithm, steering vector calculation, 2D SPS calculation and peak search are all parallelized and optimized. To the best of our knowledge, this is the first parallel MUSIC algorithm exploiting maximum parallelization with CPU-GPU architecture. In the 2D SPS calculation stage, we design an algorithm to calculate the steering vector and spatial pseudo-spectrum in parallel, and each GPU core is responsible for a task corresponding to the elevation and azimuth. Furthermore, in the peak search stage, we propose a new CFT peak search approach based on CPU and a new parallel peak search approach based on GPU acceleration. More importantly, we compare our algorithm with previous MUSIC algorithm based on CPU, The results indicate the strong potential of parallel MUSIC model with CPU-GPU architecture. Finally, it is theoretically analyzed why the resolution of 2D SPS is insufficient in specific angle ranges.

The remainder of this paper is organized as follows. In Section II, a data model of MUSIC algorithm for uniform planar arrays is given. A real-time CPU-GPU architecture for the MUSIC algorithm is described in Section III. Finally, simulation results are employed to illustrate real-time DOA estimation capability in Section IV. Conclusions are drawn in Section V.

II. DATA MODEL

As shown in Fig. 1, there is a uniform planar arrays with MN mutually independent and omnidirectional sensors, assuming that the sensors are not mutually coupled. We set the reference sensor is at the origin of coordinate system. Supposed that D signals impinge on the uniform planar arrays from the far-field. Let $\Phi_d = (\theta_d, \phi_d)$, where θ_d and ϕ_d denote the elevation and azimuth of the d th incident plane wave, respectively. The discrete signal output of the uniform planar arrays can be described in a matrix form as follows [33]

$$\mathbf{X}(t) = \mathbf{A}(\Phi)\mathbf{s}(t) + \mathbf{n}(t), \tag{1}$$

where $\mathbf{A}(\Phi) = [\mathbf{a}(\Phi_1), \mathbf{a}(\Phi_d), \ldots, \mathbf{a}(\Phi_D)] \in \mathbb{C}^{MN\times D}$, it means that $\mathbf{A}(\Phi)$ is a $MN\times D$ steering matrix related to the shape of the array and the direction of the signal source. $\mathbf{A}(\Phi)$ represents the transfer function from the source to the array and holds the DOA information. In general practical

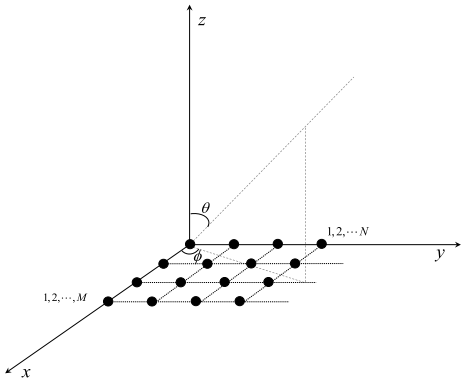

FIGURE 1. Uniform planar arrays coordinate system for localization.

applications, the shape of the sensor array will not change once it is fixed. $\mathbf{X}(t) = [x_{11}(t), x_{mn}(t), \ldots, x_{MN}(t)]^{\mathrm{T}}$ is the output vector of the uniform planar arrays, where $(\cdot)^{\mathrm{T}}$ represents the transpose operation, $t = 1, 2, \ldots, T$ is the snapshot index, $m = 1, 2, \ldots, M$, and $n = 1, 2, \ldots, N$. $\mathbf{s}(t) = [s_1(t), s_2(t), \ldots, s_D(t)]^{\mathrm{T}}$ is the source signal vector and $\mathbf{n}(t) = [n_{11}(t), n_{mn}(t), \ldots, n_{MN}(t)]^{\mathrm{T}}$ is the additive white Gaussian noise (AWGN) vector, whose elements are usually assumed to be Gaussian random variables with zero means and variance σ_n^2 as follows [33]

$$E\{\mathbf{n}(t)\mathbf{n}^{\mathrm{T}}(t)\} = \mathbf{0}, \quad E\{\mathbf{n}(t)\mathbf{n}^{\mathrm{H}}(t)\} = \sigma_n^2\mathbf{I}, \tag{2}$$

where $E\{\cdot\}$ represents mathematical expectation, $(\cdot)^{\mathrm{H}}$ represents Hermitian transpose operation, $\mathbf{0}$ is used throughout the paper to mean that all elements of matrices or vectors are zeros and $\mathbf{I}$ represents the identity matrix. The wave path difference between the mnth sensor and the reference sensor can be described as

$$\beta = \frac{2\pi}{\lambda}(x_{mn}\cos\phi_d \sin\theta_d + y_{mn}\sin\phi_d \sin\theta_d + z_{mn}\cos\theta_d), \tag{3}$$

where λ represents source signal wavelength and (x_{mn}, y_{mn}) represents the coordinate of the mnth sensor. The uniform planar arrays is generally in the x-y plane, so z_{mn} is generally 0. The steering vector of M sensors on the x-axis $\mathbf{a}_x(\Phi_d)$ can be defined as follows

$$\mathbf{a}_x(\Phi_d) = [1, e^{j2\pi\frac{\zeta}{\lambda}\cos\phi_d \sin\theta_d}, \ldots, e^{j2\pi\frac{\zeta}{\lambda}(M-1)\cos\phi_d \sin\theta_d}]^{\mathrm{T}}, \tag{4}$$

where ζ is the distance between adjacent sensors. Since the array is uniform, the distance between adjacent sensors of x-axis is the same as y-axis. Then the steering vector of N sensors on the y-axis $\mathbf{a}_y(\Phi_d)$can be defined as follows

$$\mathbf{a}_y(\Phi_d) = [1, e^{j2\pi\frac{\zeta}{\lambda}\sin\phi_d \sin\theta_d}, \ldots, e^{j2\pi\frac{\zeta}{\lambda}(N-1)\sin\phi_d \sin\theta_d}]^{\mathrm{T}}. \tag{5}$$

Assuming that the steering vector of sub-array 1 is $\mathbf{a}_x(\Phi_d)$, and the steering vector of sub-array 2 must consider the offset along the y-axis. The wave path difference of each sensor relative to the reference sensor must be added $2\pi\zeta\sin\phi\sin\theta/\lambda$ to the wave path difference of sub-array 1, so the steering vector $\mathbf{a}(\Phi_d)$ can be described as

$$\mathbf{a}(\Phi_d) = \mathbf{a}_y(\Phi_d) \otimes \mathbf{a}_x(\Phi_d), \tag{6}$$

where $\otimes$ is Kronecker product. The $W \times W$ uniform planar arrays covariance matrix can be written as

$$\mathbf{R} = E\{\mathbf{X}(t)\mathbf{X}^{\mathrm{H}}(t)\} = \mathbf{A}(\Phi)\mathbf{R}_s\mathbf{A}^{\mathrm{H}}(\Phi) + \mathbf{R}_n, \tag{7}$$

where $W = MN$, $\mathbf{R}_s = E\{\mathbf{s}(t)\mathbf{s}^{\mathrm{H}}(t)\}$ and $\mathbf{R}_n = E\{\mathbf{n}(t)\mathbf{n}^{\mathrm{H}}(t)\}$. $\mathbf{R}_s$ is the source signal covariance matrix, and $\mathbf{R}_n$ is the additive white Gaussian noise covariance matrix. According to (2), (7) can be reconstructed as

$$\mathbf{R} = \mathbf{A}(\Phi)\mathbf{R}_s\mathbf{A}^{\mathrm{H}}(\Phi) + \sigma_n^2\mathbf{I}, \tag{8}$$

where $\mathbf{R}$ is positive definite Hermitain matrix. If the unitary transformation is used to achieve diagonalization, the similar diagonal matrix is composed of W different positive real numbers, and the corresponding W feature vectors are linearly independent. The eigenvalue decomposition (EVD) of the estimation covariance matrix $\mathbf{R}$ can be described as

$$\mathbf{R} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^{\mathrm{H}} = \sum_{i=1}^{W} \eta_i \mathbf{U}_i\mathbf{U}_i^{\mathrm{H}}, \tag{9}$$

where $\mathbf{\Sigma} = diag(\eta_1, \eta_2, \ldots, \eta_W)$, and it can be proved that its eigenvalue obeys the order $\eta_1 \geq \ldots \geq \eta_D > \eta_{D+1} = \ldots = \eta_W = \sigma_n^2$. First D eigenvalues are related to the source signal, and their values are greater than σ_n^2. The eigenvectors corresponding to these D larger eigenvalues $\eta_1, \eta_2, \ldots, \eta_D$ are denoted as $\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_D$, and they form the signal subspace $\mathbf{U}_s$. Let $\mathbf{\Sigma}_s$ be diagonal matrix composed of D larger eigenvalues. The $W - D$ eigenvalues completely depend on the noise, their values are all equal to σ_n^2. The eigenvectors corresponding to $\eta_{D+1}, \eta_{D+2}, \ldots, \eta_W$ constitute the noise subspace $\mathbf{U}_n$, and $\mathbf{\Sigma}_n$ is a diagonal matrix composed of $W-D$ smaller eigenvalues. Therefore, the EVD of the exact covariance matrix $\mathbf{R}$ is given by

$$\mathbf{R} = \mathbf{U}_s\mathbf{\Sigma}_s\mathbf{U}_s^{\mathrm{H}} + \mathbf{U}_n\mathbf{\Sigma}_n\mathbf{U}_n^{\mathrm{H}}. \tag{10}$$

However, the covariance matrix $\mathbf{R}$ is not available in practice, so the theoretical uniform planar arrays covariance matrix $\mathbf{R}$ given in (7) is usually replaced by $\hat{\mathbf{R}}$

$$\hat{\mathbf{R}} = \frac{1}{T}\mathbf{X}\mathbf{X}^{\mathrm{H}}, \tag{11}$$

where $\mathbf{X} = [\mathbf{X}(1), \mathbf{X}(t), \ldots, \mathbf{X}(T)]$ is the uniform planar arrays output discrete signal matrix. The EVD of the exact covariance matrix $\hat{\mathbf{R}}$ in (10) can be described as

$$\hat{\mathbf{R}} = \hat{\mathbf{U}}_s\hat{\mathbf{\Sigma}}_s\hat{\mathbf{U}}_s^{\mathrm{H}} + \hat{\mathbf{U}}_n\hat{\mathbf{\Sigma}}_n\hat{\mathbf{U}}_n^{\mathrm{H}}. \tag{12}$$

Among the eigenvalue-decomposition-like DOA estimation algorithms for the sensor array covariance matrix, the MUSIC algorithm has universal applicability. As long as the layout of the sensor array is known, whether it is a linear array or a circular array, and whether the array senors are equally spaced, high-resolution estimation results can be

obtained. The uniform planar arrays covariance matrix $\hat{\mathbf{R}}$ can be divided into two parts, we can get

$$\hat{\mathbf{R}}\hat{\mathbf{U}}_n = \mathbf{A}(\Phi)\mathbf{R}_s\mathbf{A}^{\mathrm{H}}(\Phi)\hat{\mathbf{U}}_n + \sigma_n^2\hat{\mathbf{U}}_n = \sigma_n^2\hat{\mathbf{U}}_n, \tag{13}$$

according to (13), we get

$$\mathbf{A}(\Phi)\mathbf{R}_s\mathbf{A}^{\mathrm{H}}(\Phi)\hat{\mathbf{U}}_n = \mathbf{0}. \tag{14}$$

The matrix $\mathbf{R}_s$ is a full-rank and not singular matrix, so there is an inverse. Therefore the above (14) can be changed to $\mathbf{A}^{\mathrm{H}}(\Phi)\hat{\mathbf{U}}_n = \mathbf{0}$, and it shows that each column vector in matrix A is orthogonal to the noise subspace. We can get

$$\hat{\mathbf{U}}_n^{\mathrm{H}}\mathbf{a}(\Phi_d) = \mathbf{0}, \quad d = 1, 2, \ldots, D. \tag{15}$$

According to the orthogonal relationship between the noise eigenvector and the source signal steering vector, the following 2D SPS function is obtained

$$\mathrm{SPS}_{\mathrm{MUSIC}}(\theta, \phi) = \frac{1}{\mathbf{a}(\theta, \phi)^{\mathrm{H}}\hat{\mathbf{U}}_n\hat{\mathbf{U}}_n^{\mathrm{H}}\mathbf{a}(\theta, \phi)}. \tag{16}$$

It is easy to find that (16) can be regarded as an optimization problem, so (16) can be reconstructed as

$$\min_{(\theta,\phi)} \left\| \mathbf{a}(\theta, \phi)^{\mathrm{H}}\hat{\mathbf{U}}_n\hat{\mathbf{U}}_n^{\mathrm{H}}\mathbf{a}(\theta, \phi) \right\|_{\mathrm{F}}, \tag{17}$$

where $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm. We can obtain elevation and azimuth of $1, 2, \ldots, D$ source signal as follows

$$\{\hat{\theta}_d, \hat{\phi}_d\}_{d=1}^{D} = \arg\min_{(\theta,\phi)} \left\| \mathbf{a}(\theta, \phi)^{\mathrm{H}}\hat{\mathbf{U}}_n\hat{\mathbf{U}}_n^{\mathrm{H}}\mathbf{a}(\theta, \phi) \right\|_{\mathrm{F}}. \tag{18}$$

The estimation of the elevation and azimuth can be accomplished by exercising the 2D peak search.

III. REAL-TIME CPU-GPU ARCHITECTURE FOR MUSIC ALGORITHM

In this section, we will introduce a CPU-GPU architecture that can reduce the computation time cost of MUSIC algorithm to a very low level to fit real-time requirements. This architecture can be used for real-time sound source location, real-time signal source search, and medical imaging systems. The working flow of the proposed architecture is described.

As mentioned above, the MUSIC algorithm is a powerful and high-performance algorithm for determining the DOA of common wideband or narrowband signal sources, especially when the number of arrays is appropriate and the scan resolution is moderate. However, in the current high-precision positioning applications, more attention is paid to whether it can satisfy real-time requirements while ensuring accuracy. In the case of only one-dimensional scan, the MUSIC algorithm can almost be an excellent one of the same type of DOA algorithms, once the elevation and azimuth are searched simultaneously, the entire computation time will become unbearable. We proposed a CPU-GPU architecture for the MUSIC algorithm. This architecture fully utilizes the high computational power coming from the parallel processing of GPU and overcomes the mentioned limited computational efficiency.

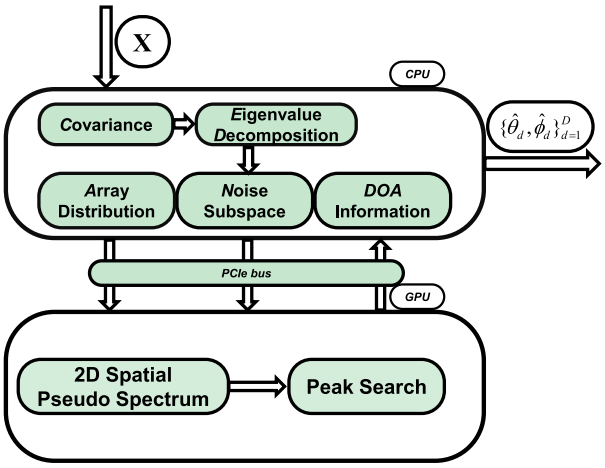

FIGURE 2. Model of CPU-GPU architecture for MUSIC algorithm.

A. WORKING FLOW OF THE PROPOSED CPU-GPU ARCHITECTURE

Fig.2 shows a simplified block diagram within interval of the computation based on CPU-GPU architecture where $\mathbf{X}$ and $\{\hat{\theta}_d, \hat{\phi}_d\}_{d=1}^{D}$ represent the input and output of this CPU-GPU framework, respectively. The CPU executes some modules with complex functions and complex logic operations, which mainly include the EVD and separation of noise subspace, while the GPU mainly executes the remaining 2D spectrum generation and peak search with high repeatability and low interdependence. First, the proposed architecture takes the sampled signal $\mathbf{X}$ as the input. Secondly, CPU is the calculation platform of the previous stage, including covariance calculation (11), eigenvalue decomposition (9), and noise subspace extraction (12). Then the inner product of the noise subspace (16) is transmitted to the GPU, and the candidate steering vector (4), (5), (6) and the 2D SPS (16) are calculated in parallel. Finally, the 2D spectral peak search (18) is optimized in parallel and the DOA information $\{\hat{\theta}_d, \hat{\phi}_d\}_{d=1}^{D}$ is transmitted back to the CPU. More details of the calculation scheme and memory transfer between CPU and GPU are as follows.

1) CPU COMPUTATION PART

Generally, the complex calculations in the solving process are executed on the CPU. The multi-core CPU is mainly responsible for complex logic operations on the basis of providing vector processing. To minimize the memory transmission between the CPU and GPU, steering vectors of candidate angles are calculated via the GPU. Since covariance calculation, eigenvalue decomposition, and noise subspace separation are complicated and not repetitive, they are calculated via the CPU.

2) GPU COMPUTATION PART

The design goal of GPU is its ultra-high floating-point computing power. The GPU avoids or weakens complex functions that are not related to floating-point calculations, such as

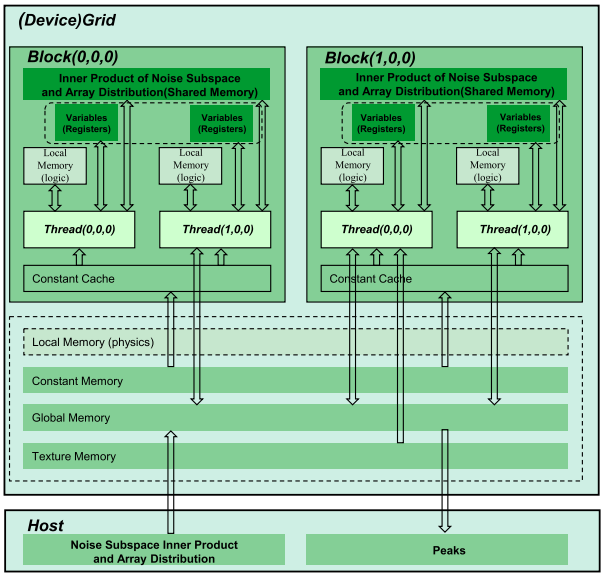

FIGURE 3. Each memory transfer inside GPU and CPU-GPU memory transfer model.

branch processing and logic control, and focuses on floating-point calculations [34], [35]. For the estimation of source signal elevation and azimuth based on the MUSIC algorithm in the same time block, a vast quantity of parallel computing branches are worth using GPU to optimize the computation task of the 2D spectrum. Since the noise subspace required for the elevation and azimuth search of the MUSIC algorithm in the same time block is constant, the noise subspace is placed in the on-chip shared memory of the GPU to facilitate accessing the memory unit. Moreover, the peak search task is handed over to the GPU for execution.

As shown in Fig. 3, the inner product of noise subspace and array distribution are transferred to the GPU global memory through the CPU memory and then to the shared memory. Some of the memory types are on-chip memory whose access speed is faster than off-chip memory. Therefore, utilization of shared memory to store the inner product of noise subspace and array distribution enhances data access rate. It is worth noting that although local memory is unique to each thread, this part of memory still belongs to off-chip memory, which belongs to the same physical memory as global memory.

3) MEMORY TRANSMISSION BETWEEN CPU AND GPU

Memory transmission in the CPU-GPU architecture is always an unavoidable issue that demands to be considered. Due to the distinct physical memory addresses of CPU and GPU, as well as the bandwidth and transmission speed limitations of PCIe bus, it is necessary to transfer less data between CPU and GPU. If the calculation of the candidate steering vector is established into the CPU and a unique steering vector is required for each DOA information update, then the one-by-one transmission of this portion of data will consume a lot of time. Therefore, the proposed CPU-GPU architecture for the MUSIC algorithm arranges the calculation of the steering vector on the GPU and only requires to transmit the array distribution vector once during the DOA estimation of the first time block. Similarly, if the GPU transmits the calculated 2D spectrum back to the CPU each time, the time of each transmission delays the entire calculation, so the task of peak search is likewise scheduled for the GPU calculation. More optimization details will be discussed next.

TABLE 1. Computation complexity of MUSIC algorithm.

Computation Items	Equation Number	Computation Complexity
Covariance	(11)	$O(W^2T)$
EVD	(9)	$O(W^3)$
Candidate Steering Vector	(4), (5), (6)	$O(L_\theta L_\phi W)$
Inner Product of Noise Subspace	(16)	$O(W^2(W-D))$
2D Spectrum	(16)	$O(L_\theta L_\phi W^2)$
Peak Search	(18)	$O(L_\theta L_\phi)$

4) CONDITION FOR REAL-TIME EXECUTION

Since the DOA information of a time block must be updated before the next time block. The condition of the proposed CPU-GPU architecture for MUSIC algorithm to be executed in real-time is as follows

$$T_{\text{CPU}} + T_{\text{GPU}} + T_{\text{Transfer}} \leq T_{\text{Block}}, \tag{19}$$

where T_{CPU} is the preprocessing time of CPU calculation includes covariance calculation, eigendecomposition, and inner product of noise subspace. T_{GPU} is the computation time of all kernels on the GPU that includes 2D spectrum computation and peak search. T_{Transfer} is the time of data transmitted between CPU and GPU. T_{Transfer} does not include the transmission time of array distribution and T_{Block} represents the sampling period of the signal block.

B. PARALLEL OPTIMIZATION STRATEGIES

The computation complexity of MUSIC algorithm is listed in Table 1. Inner Product of Noise Subspace represents $\hat{\mathbf{U}}_n\hat{\mathbf{U}}_n^{\text{H}}$, 2D Spectrum is the remainder of the (16). L_θ, L_ϕ represents scan range of elevation and azimuth, respectively. The number of snapshots T only affects the computation complexity of the covariance matrix linearly. The 2D spectrum is related to the number of sensors W and the scan range of the elevation and azimuth. Through comparative analysis, we can perceive that candidate steering vector, 2D spectrum, and peak search have fine parallelism.

1) PARALLEL OPTIMIZATION OF CANDIDATE STEERING VECTOR AND 2D SPS COMPUTATION

Due to the computation of the spectrum of each candidate elevation and azimuth are not related to each other, the high-level parallel structure is determined to estimate the 2D spectrum. The DOA estimation of each sampling block does not depend on the previous sampling block. Such excellent independent computing features are very suitable for parallelism.

In Algorithm 1, after the shared memory allocation is completed, only one thread synchronization is performed within

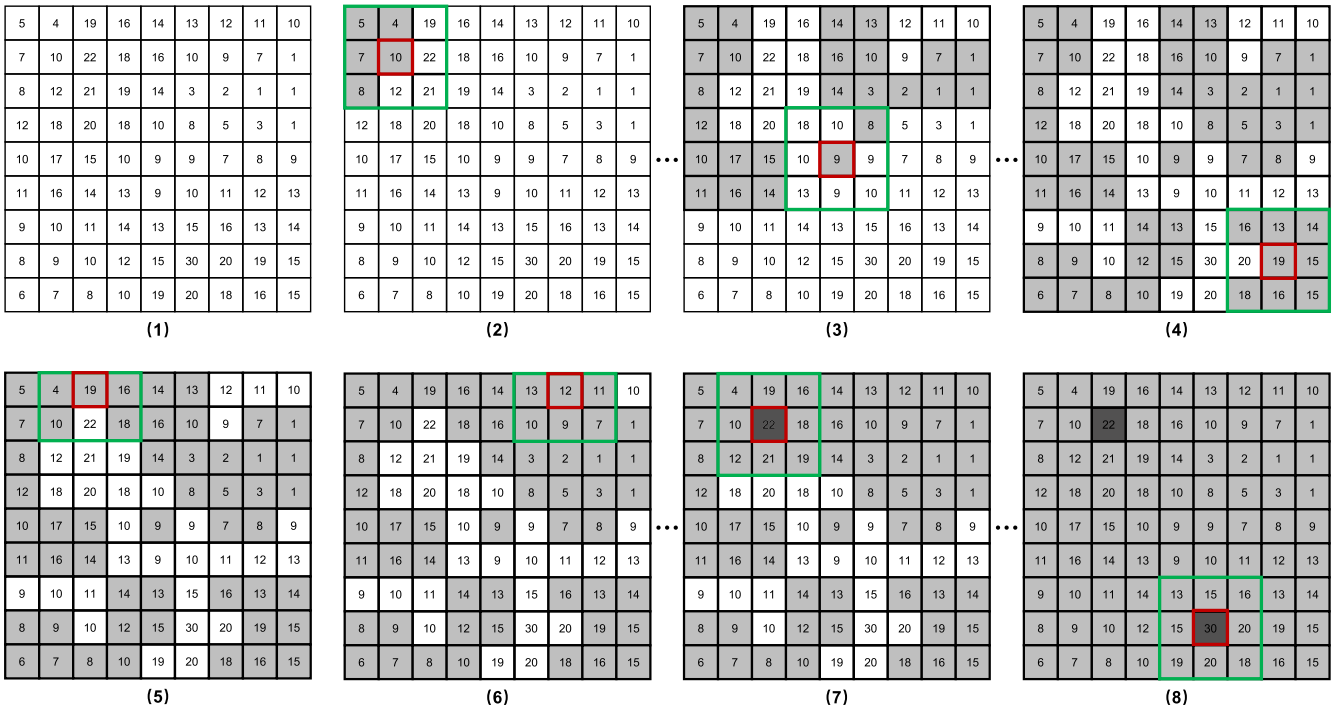

FIGURE 4. The upper four figures represent the first coarse traversal process, the lower four figures represent the second fine traversal process.

the block, and the rest proceed downward smoothly. The outermost loop is a manifestation of SIMT, where multiple threads are bound into a warp to execute the same instruction stream. Only one main branch is set up in the kernel, which is not disturbed by other branches. While a certain instruction is scheduled, the cores in SIMT is executed according to SIMD model, that is, each core in the warp or arithmetic and logic unit (ALU) executes the same operation on multiple data operands concurrently. It is worth mentioning that all intermediate variables in Algorithm 1 are stored in registers to ensure the access speed. Therefore, it is critical to ensure data size of variables in the block does not exceed the total capacity of registers available in each block. Furthermore, it is necessary to clear the status of accumulators promptly.

2) PARALLEL OPTIMIZATION OF PEAK SEARCH

In Algorithm 2, we set the number of surrounding points as C_n, and the entire comparison times are approximately $L_\theta \times L_\phi \times C_n$. Each point must be compared with all surrounding points. The *Priority_queue*(D) is used to maintain the D max peaks. The number of surrounding points of four vertices, four edges, and the rest of the spectrum are 3, 5, and 8 respectively. About half of the comparisons are repeated. Therefore, Algorithm 2 is modified to use a peak label spectrum to record the result of the current comparison. As shown in Algorithm 3, if the adjacent point is judged not to be the peak point, then this point is ignored. In this way, the redundant comparison is well removed, and comparison times is lower than $1/2$. Moreover, comparison times can be reduced to a lower level by changing the step size.

As shown in Fig. 4, the first four subgraphs show that the first rough traversal, which can filter out more than half of the non-peak points. The last four subgraphs show that the second fine traversal can skip non-peak points and filter some unpassed points. Ultimately, two peak points are obtained. Nevertheless, the disadvantage of this algorithm is that it needs to transfer the 2D spectrum from the GPU back to the CPU, so the calculation efficiency is only increased by about twice. If the DOA information can be calculated in the GPU, the transfer time from GPU to CPU can be saved. Firstly, Algorithm 4 utilizes the high-efficiency access characteristic of shared memory in GPU to search for peaks within blocks and assigns them to global memory. Secondly, after all peaks within blocks are searched, the peaks of each block are scanned one by one and placed in *Priority_queue*(D). Finally, DOA information is transmitted back to the CPU to indicate that the DOA estimation within a time block is completed.

IV. SIMULATIONS AND DISCUSSION

In this section, we carried out comparative experiments in time domain to verify the real-time feasibility of the proposed calculation models and strategies.

A. SIMULATIONS SETTINGS

The MUSIC algorithm simulations based on CPU are implemented in MATLAB 2018a. The code implementation of real-time MUSIC algorithm with CPU-GPU architecture uses a combination of VS2019, C++, and CUDA. The computational performance of strategies is analyzed on the Intel i5-8400 and NVIDIA GeForce GTX 1080 Ti. Specifications

Algorithm 1 Parallel Optimization of Candidate Steering Vector and 2D Spectrum

Input: Inner Product of Noise Subspace $\hat{\mathbf{U}}_n\hat{\mathbf{U}}_n^{\mathrm{H}}$
Sensor array distribution dis_x and dis_y
Output: 2D $sps(\theta,\phi)$, *where* $\theta\ \forall\ L_\theta$ *and* $\phi\ \forall\ L_\phi$

Cuda Thread $\leftarrow$ **Copy2GPU**($\hat{\mathbf{U}}_n\hat{\mathbf{U}}_n^{\mathrm{H}}, dis_x, dis_y$)
Define Shared Memory : ad_x, ad_y **and** u
$tid \leftarrow threadIdx$
$bid \leftarrow blockIdx$
$t \leftarrow bid \times blockDim + tid$
where $blockDim \geq W+M+N$
$t_n \leftarrow blockDim \times gridDim$
if $tid < W$ **then**
  $u(tid) = \hat{\mathbf{U}}_n\hat{\mathbf{U}}_n^{\mathrm{H}}(tid)$
else if $tid < W+M$ **then**
  $ad_x(tid-W) = dis_x(tid-W)$
else if $tid < W+M+N$ **then**
  $ad_y(tid-W-M) = dis_y(tid-W-M)$
end if
Synchronize Threads
for t **to** $L_\theta \times L_\phi$ **do**
  for $i := 0$ **to** $M-1$ **do**
    $a_x(i) = e^{j2\pi ad_x(i)\sin(\frac{t}{L_\phi})\cos(t \bmod L_\phi)}$
  end for
  for $i := 0$ **to** $N-1$ **do**
    $a_y(i) = e^{j2\pi ad_y(i)\sin(\frac{t}{L_\phi})\sin(t \bmod L_\phi)}$
  end for
  for $i := 0$ **to** $M-1$ **do**
    for $j := 0$ **to** $N-1$ **do**
      $a(i\times N+j) = a_x(j) \times a_y(i)$
    end for
  end for
  for $i := 0$ **to** $W-1$ **do**
    for $j := 0$ **to** $W-1$ **do**
      $temp1(i) += a(j) \times u(i\times W+j)$
    end for
  end for
  for $j := 0$ **to** $W-1$ **do**
    $temp2 += temp1(i) \times$ **Conj**$(a(i))$
  end for
  $sps(\theta,\phi) = \|\frac{1}{temp2}\|_{\mathrm{F}}^2$
  Clear the accumulators $temp1[W]$ and $temp2$
  $t += t_n$
end for

Algorithm 2 Original Implementation of Peak Search

Input: 2D $sps(\theta,\phi)$
Output: DOA information $\{\hat{\theta}_d, \hat{\phi}_d\}_{d=1}^{D}$

for $i := 0$ **to** L_θ **do**
  for $j := 0$ **to** L_ϕ **do**
    if $sps(i,j) >$ all surrounding points **then**
      $Priority\_queue(D) \leftarrow sps(i,j)\ \&\ (i,j)$
    end if
  end for
end for
$\{\hat{\theta}_d, \hat{\phi}_d\}_{d=1}^{D} \leftarrow (i,j) \in Priority\_queue(D)$

Algorithm 3 Optimization of Peak Search via CPU Through CTF

Input: 2D $sps(\theta,\phi)$
Output: DOA information $\{\hat{\theta}_d, \hat{\phi}_d\}_{d=1}^{D}$

for $i := 0$ **to** L_θ **do**
  for $j := 0$ **to** L_ϕ **do**
    if $p(i,j) ==$ *non-peak label* **then**
      Continue
    end if
    if $sps(i,j) > sps(i\pm1, j\pm1)$ **then**
      $p(i\pm1, j\pm1) \leftarrow$ *non-peak label*
      if $sps(i,j) >$ all surrounding points **then**
        $Priority\_queue(D) \leftarrow sps(i,j)\ \&\ (i,j)$
      else
        $p(i,j) \leftarrow$ *non-peak label*
      end if
    end if
  end for
end for
$\{\hat{\theta}_d, \hat{\phi}_d\}_{d=1}^{D} \leftarrow (i,j) \in Priority\_queue(D)$

TABLE 2. Specifications of CPU and GPU used for simulation.

Specifications	CPU	GPU
Manufacture	Intel	NVIDIA
Product	Intel i5-8400	GeForce GTX 1080 Ti
CUDA runtime version	-	11.1
CUDA capability	-	6.1
Clock rate	2.80 GHz	1.63 GHz
RAM/Global memory	16.0 GB	11.0 GB
Number of cores	6	3584
Shared memory per block	-	48 KB
Registers per block	-	65536
CPU-GPU bandwidth	9.8 GB/s (H2D)	8.2 GB/s (D2H)
Memory bus width	64 bit	352 bit

of CPU, GPU, and system parameters used in the algorithm are listed in Table 2. The parameters of uniform planar arrays are $M = 4$ and $N = 4$.

B. INFLUENCE OF SNAPSHOTS AND MULTIPLE SOURCES ON COMPUTATION TIME AND ACCURACY

We simulate in the narrowband far-field environment. The sensor interval is set to 0.01 m to satisfy $\zeta \leq \frac{\lambda}{2}$. The center frequency of the narrowband signal is 2 kHz. As shown in Table 3, We use different number of snapshots and signal sources to compare the impact on CPU calculation time and accuracy. 100, 200, 500, and 1000 snapshots correspond to the sampling time of 0.05 s, 0.1 s, 0.25 s, and 0.5 s respectively. The number of signal sources has a slight influence on accuracy of DOA estimation. Since multiple sources in random experiments may be more concentrated, the RMSE is slightly higher than that of a single source. The simulation results are all averaged on 500 independent experiments.

Algorithm 4 Parallel Optimization of Peak Search via GPU

Input: 2D $sps(\theta, \phi)$
Output: DOA information $\{\hat{\theta}_d, \hat{\phi}_d\}_{d=1}^{D}$

Define Shared Memory : $shared\_sps$
$t \leftarrow blockIdx \times blockDim + threadIdx$
$t_n \leftarrow blockDim \times gridDim$
Global2Shared :$shared\_sps \leftarrow sps \in blockDim$
Synchronize Threads
for t **to** $L_\theta \times L_\phi$ **do**
  if $shared\_sps(t)$ all surrounding points **then**
    $peak(\frac{t}{t_n}, blockIdx) \leftarrow shared\_sps(t), t$
  end if
  $t+ = t_n$
end for
Synchronize All Threads
$gridSize = \frac{L_\theta \times L_\phi}{t_n}$
for $i := 0$ **to** $gridSize \times gridDim$ **do**
  if $peak(i) \neq NULL$ **then**
    $Priority\_queue(D) \leftarrow peak(i)$
  end if
end for
$\{\hat{\theta}_d, \hat{\phi}_d\}_{d=1}^{D} \leftarrow (i, j) \in Priority\_queue(D)$

TABLE 3. Comparison of the influence of snapshots and multiple sources on estimation accuracy and computation time where SNR = 10 dB.

Snapshots	Sources	Computation time(ms)		RMSE(°)	
		Covariance	$\hat{\mathbf{U}}_n\hat{\mathbf{U}}_n^{\mathrm{H}}$	θ	ϕ
100	1	0.487	2.74	0.078	0.077
	2	0.486	3.13	0.080	0.079
	3	0.479	3.14	0.095	0.093
200	1	0.565	3.21	0.076	0.078
	2	0.555	3.20	0.078	0.077
	3	0.574	3.19	0.081	0.082
500	1	0.609	3.41	0.050	0.051
	2	0.600	3.44	0.054	0.054
	3	0.610	3.42	0.064	0.062
1000	1	0.764	3.50	0.050	0.050
	2	0.754	3.51	0.053	0.051
	3	0.763	3.46	0.059	0.058

The CPU overhead is about a quarter for the proposed algorithm to execute in the worst environmental conditions. Besides, the RMSE gradually decreases with the increase of the number of snapshots, yet the increase of accuracy is not linear but logarithmic. We cannot rely on increasing the number of snapshots to improve the estimation accuracy, after all the calculation time also increases linearly. Across the comprehensive comparison, 500 snapshots are used as the calibration parameter in subsequent simulations.

C. MAXIMUM NUMBER OF SIGNALS

The estimated range of UPA elevation is 0°-90°, and azimuth is 0°-360°. The maximum number of signal sources can be estimated in this simulation test. Since we use the $W \times W$ Noise Subspace $\mathbf{U}_n$, theoretically W-1 signal source estimation can be achieved. We set SNR = 15 dB and test 15 uncorrelated random signals. As shown in Fig. 5, to facilitate the observation of the 2D SPS, 8 signals

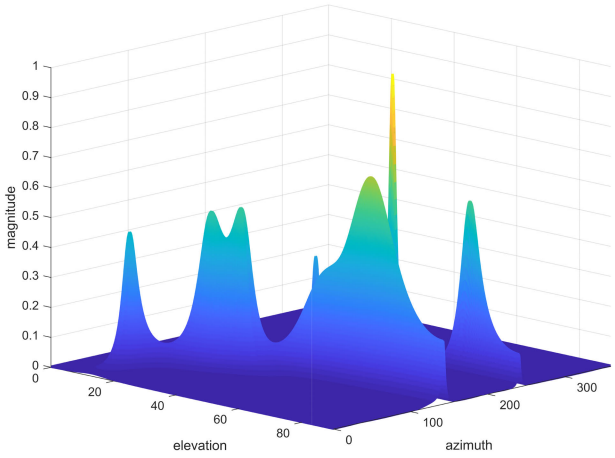

FIGURE 5. Simulation 2D SPS of the maximum estimated number of signal sources where SNR = 15 dB.

of them are selected to analyze accuracy. The elevation and azimuth of the 8 uncorrelated signals are (15°, 40°), (30°, 80°), (43°, 101°), (56°, 132°), (71°, 142°), (45°, 160°), (63°, 204°) and (76°, 239°), respectively. We observe that 7 prominent crests are corresponding to 7 signal sources, and a small bump is on the waist of the widest wave, which corresponds to the signal source of (56°, 132°).

D. RESOLUTION ANALYSIS

As shown in Fig. 5, as the incident elevation of the signal exceeds 60°, the waveform of the 2D SPS becomes less sharp. The resolution of signal source in a certain direction is directly related to the rate of change of steering vector near that direction. Near the direction where the steering vector changes rapidly, the difference of array wave path changes greatly with the change of the source angle, and the corresponding resolution is also high [36]. Define the resolution $\Gamma(\theta)$ as follows

$$\Gamma(\theta) = \left\| \frac{d\mathbf{a}(\theta)}{d\theta} \right\| \propto \left\| \frac{d\tau}{d\theta} \right\|, \tag{20}$$

where τ represents delay among sensors in the array. The spectrum in the θ direction becomes sharper as $\Gamma(\theta)$ increases. For the uniform linear array, τ is as follows

$$\tau_l = \frac{1}{c}(x_l \sin\theta) = \frac{\zeta}{c}(l-1)\sin\theta, \tag{21}$$

where τ_l is the delay of the lth sensor relative to the reference sensor, c is signal propagation speed, and x_l is the x-axis coordinate of the lth sensor. Then

$$\Gamma(\theta) \propto \cos(\theta). \tag{22}$$

It shows that the resolution of signal in the location of 0° is the highest, and the resolution in the position of 60° direction is reduced by half.

TABLE 4. Comparison of the run-time and speed-up ratio of different strategies in different scan ranges. And the sampling period of the signal block is 0.25s, snapshots is 500, bold of each column is related to maximum speed-up ratio, and the run-time is in ms.

Strategies	Scan Range	Preprocessing via CPU			H2D		2D	D2H	Peak	D2H	Total	Speed-up
		Cov	EVD	$\hat{\mathbf{U}}_n\hat{\mathbf{U}}_n^{\mathrm{H}}$	$\hat{\mathbf{U}}_n\hat{\mathbf{U}}_n^{\mathrm{H}}$	$dis_{x,y}$	Spectrum		Search		Runtime	
CPU-Config1	90° × 90°	0.601	3.45	0.257	-	-	27817	-	55.4	-	27877	-
	90° × 180°	0.600	3.45	0.257	-	-	55932	-	92.8	-	56029	-
	90° × 360°	0.603	3.45	0.256	-	-	112270	-	167.2	-	**112442**	-
CPU-Config2	90° × 90°	0.601	3.44	0.255	-	-	27825	-	27.4	-	27857	-
	90° × 180°	0.602	3.45	0.253	-	-	55944	-	46.0	-	55994	-
	90° × 360°	0.602	3.45	0.258	-	-	**112274**	-	83.3	-	112362	-
GPU-Config3	90° × 90°	0.603	3.46	0.257	0.065	0.031	180	1.162	547.5	-	733	38.03x
	90° × 180°	0.601	3.46	0.256	0.064	0.030	354	2.304	1091.1	-	1452	38.59x
	90° × 360°	0.599	3.45	0.255	0.065	0.031	703	4.633	**2192.0**	-	2904	38.72x
GPU-Config4	90° × 90°	0.599	3.44	0.255	0.066	0.032	179	1.159	272.2	-	457	61.00x
	90° × 180°	0.600	3.44	0.255	0.060	0.031	356	2.301	543.0	-	907	61.77x
	90° × 360°	0.601	3.46	0.255	0.063	0.030	**692**	4.605	1084.3	-	1785	62.99x
GPU-Config5	90° × 90°	0.601	3.45	0.254	0.064	0.030	180	-	1.12	0.039	186	149.88x
	90° × 180°	0.599	3.45	0.253	0.065	0.031	354	-	2.03	0.038	362	154.78x
	90° × 360°	0.603	3.45	0.257	0.066	0.032	694	-	**2.99**	0.039	**702**	160.17x
Max Speed-up	-	-	-	-	-	-	**162.25x**	-	**733.11x**	-	**160.17x**	-

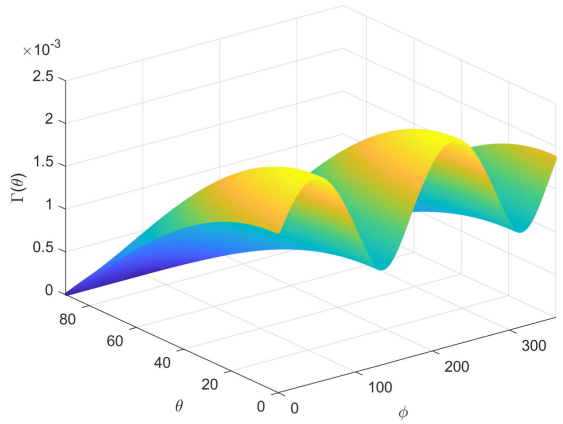

FIGURE 6. Resolution $\Gamma(\theta)$, where ϕ is a parameter.

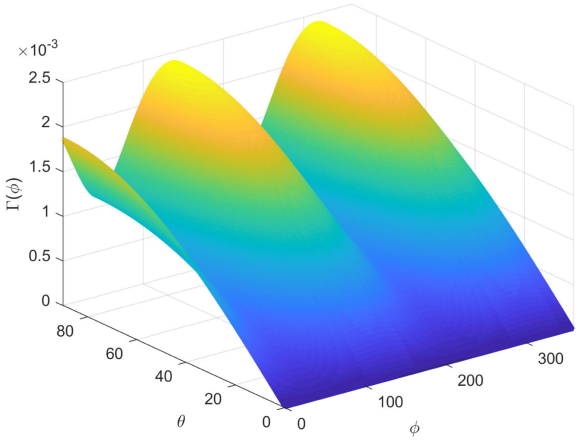

FIGURE 7. Resolution $\Gamma(\phi)$, where θ is a parameter.

For the uniform planar arrays placed horizontally, τ_{mn} can be written as follows

$$\begin{aligned}\tau_{mn} &= \frac{1}{c}(x_m \cos\phi \sin\theta + y_n \sin\phi \sin\theta) \\ &= \frac{\zeta}{c}((m-1)\cos\phi + (n-1)\sin\phi)\sin\theta. \end{aligned} \quad (23)$$

According to (20), we can get $\Gamma(\theta)$ of the uniform planar arrays as follows

$$\begin{aligned}\Gamma(\theta) &= \left\| \frac{d\mathbf{a}(\theta,\phi)}{d\theta} \right\| \propto \left\| \frac{d\tau_{mn}}{d\theta} \right\| \\ &= \frac{\zeta\cos\theta}{c} \sqrt{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}(m\cos\phi + n\sin\phi)^2}. \end{aligned} \quad (24)$$

Similarly, we can get $\Gamma(\phi)$ as follows

$$\begin{aligned}\Gamma(\phi) &= \left\| \frac{d\mathbf{a}(\theta,\phi)}{d\phi} \right\| \propto \left\| \frac{d\tau_{mn}}{d\phi} \right\| \\ &= \frac{\zeta\sin\theta}{c} \sqrt{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}(n\cos\phi - m\sin\phi)^2}. \end{aligned} \quad (25)$$

The above analysis verifies that the resolution is insufficient if elevation exceeds 60° in Fig. 5. In the light of (24), $\Gamma(\theta)$ keeps getting smaller as θ increases where ϕ is a parameter. As shown in Fig. 6, the spectrum has the highest resolution where the elevation is 0°, and the resolution is reduced to half at 60°, until reduced to 0. As shown in Fig. 7, $\Gamma(\phi)$ fluctuates sinusoidally in an interval as ϕ increases where θ is parameter. However, as θ gets closer to 0, the fluctuation range of $\Gamma(\phi)$ gets smaller until it finally reaches 0.

E. STRATEGY COMPARISON

To verify the rationality in real-time application, the proposed algorithm is compared on the CPU and GPU. For notational convenience, we abbreviate five calculation strategies Algorithm2, Algorithm3, Algorithm1-Algorithm2, Algorithm1-Algorithm3 and Algorithm1-Algorithm4 to CPU-Config1, CPU-Config2, GPU-Config3, GPU-Config4, and GPU-Config5, respectively. The specifications of the hardware required for the experiment are shown in Table 2. The sampling block period of the signal is 0.25 s, and the scan step is 0.1°. The angle scan range are 90° × 90°, 90° × 180°,

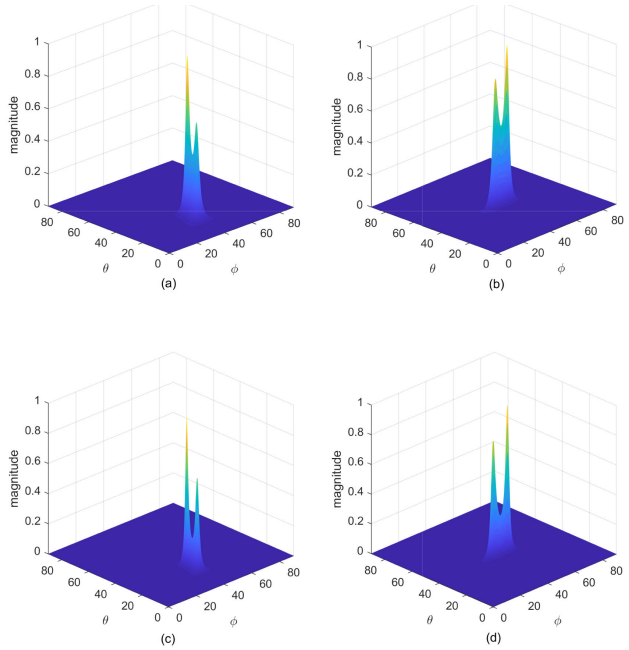

FIGURE 8. 2D SPS of two closely spaced simultaneous signals. (a) SNR = 0 dB; (20°,40°), (28°,40°). (b) SNR = 0 dB; (40°,40°), (40°,48°). (c) SNR = 2 dB; (20°,40°), (28°,40°). (d) SNR = 2 dB; (40°,40°), (40°,48°).

and 90° × 360°, respectively. All the simulation results are obtained by averaging 500 independent experiments.

The simulation results are shown in Table 4, the preprocessing via CPU calculation includes covariance calculation, eigendecomposition, and inner product of noise subspace. H2D is memory transfer from CPU to GPU, and D2H is memory transfer from GPU to CPU. From GPU back to CPU, the transmission time of 2D SPS is much longer than that of DOA information. The highest speed-up ratio for computing the 2D spectrum is 162.25x, the highest speed-up ratio for peak search is 733.11x, and the speed-up ratio for the whole calculation is 160.17x. By using 90° × 90° scan range, the GPU outputs DOA information every 0.25 s. The scan range of 90° × 180° and 90° × 360° also be simulated. According to Table 3, increasing the sampling block of the signal hardly affect the T_{CPU}. Therefore, the real-time condition is still contented and the output frequency of DOA information is less than 4 Hz.

In practical conditions, the signal sources may propagate from adjacent directions. Here, we simulate two sets of adjacent signal sources with SNR = 0 dB and 2 dB whose elevation and azimuth are (20°,40°), (28°,40°) and (40°,40°), (40°,48°), respectively, i.e., the elevation and azimuth only differ 8° between the adjacent source. Fig. 8 plots 2D SPS of the proposed method in minimum SNR scenario. Under this challenging scenario, two peaks can still be detected. In the same SNR scenario, if the angle difference between adjacent signals continues to decrease, the two peaks cannot be clearly distinguished. In this set of simulations, we use GPU-Config5 as the calculation strategy. Through statistics, the time spent in each part is $T_{\text{CPU}} = 4.3$ ms, $T_{\text{GPU}} = 180.1$ ms, $T_{\text{Transfer}} = 0.137$ ms, and $T_{\text{Block}} = 250$ ms. According to the above statistics and (19), we observe that this set of simulations still meet the real-time execution condition.

V. CONCLUSION

This paper presents an optimized real-time MUSIC algorithm using CPU-GPU architecture for uniform planar arrays. Joint parallel of candidate steering vector and 2D spectrum make full use of shared memory, SIMT, and SIMD, reducing transmission time and delay waiting for calculation. Besides, the parallel optimization of peak search via the GPU saves the transmission time of SPS and accelerates the entire process. Importantly, the GPU-Config5 strategy was shown to have performance benefits of 150x to 160x in comparison to the CPU-Config1 strategy. Moreover, theoretical analysis shows why the resolution of the 2D SPS continues to decrease with the increasing elevation. In the future, we plan to further improve the estimation effectiveness on the basis of ODMUSIC by remodeling the CPU-GPU architecture.

REFERENCES

[1] H. Krim and M. Viberg, "Two decades of array signal processing research: The parametric approach," *IEEE Signal Process. Mag.*, vol. 13, no. 4, pp. 67–94, Jul. 1996.
[2] A. Hero, H. Messer, J. Goldberg, and D. Thomson, "Highlights of statistical signal and array processing," *IEEE Signal Process. Mag.*, vol. 15, no. 5, pp. 21–64, Sep. 1998.
[3] W. C. Knight, R. G. Pridham, and S. M. Kay, "Digital signal processing for sonar," *Proc. IEEE*, vol. 69, no. 11, pp. 1451–1506, Nov. 1981.
[4] J. C. Mosher, R. M. Leahy, and P. S. Lewis, "Biomagnetic localization from transient quasi-static events," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, vol. 1, Apr. 1993, pp. 91–94.
[5] N. Wang, P. Agathoklis, and A. Antoniou, "A new DOA estimation technique based on subarray beamforming," *IEEE Trans. Signal Process.*, vol. 54, no. 9, pp. 3279–3290, Sep. 2006.
[6] P. Stoica and A. B. Gershman, "Maximum-likelihood DOA estimation by data-supported grid search," *IEEE Signal Process. Lett.*, vol. 6, no. 10, pp. 273–275, Oct. 1999.
[7] R. P. Lemos, H. V. L. E. Silva, E. L. Flores, J. A. Kunzler, and D. F. Burgos, "Spatial filtering based on differential spectrum for improving ML DOA estimation performance," *IEEE Signal Process. Lett.*, vol. 23, no. 12, pp. 1811–1815, Dec. 2016.
[8] P. Wang, Y. Kong, X. He, M. Zhang, and X. Tan, "An improved squirrel search algorithm for maximum likelihood DOA estimation and application for MEMS vector hydrophone array," *IEEE Access*, vol. 7, pp. 118343–118358, 2019.
[9] J. Dai, X. Bao, W. Xu, and C. Chang, "Root sparse Bayesian learning for off-grid DOA estimation," *IEEE Signal Process. Lett.*, vol. 24, no. 1, pp. 46–50, Jan. 2017.
[10] E. C. Marques, N. Maciel, L. Naviner, H. Cai, and J. Yang, "A review of sparse recovery algorithms," *IEEE Access*, vol. 7, pp. 1300–1322, 2019.
[11] M. Viberg and B. Ottersten, "Sensor array processing based on subspace fitting," *IEEE Trans. Signal Process.*, vol. 39, no. 5, pp. 1110–1121, May 1991.
[12] R. Roy, A. Paulraj, and T. Kailath, "Estimation of signal parameters via rotational invariance techniques-ESPRIT," in *Proc. IEEE Mil. Commun. Conf.*, vol. 3, Oct. 1986, pp. 6–41.
[13] S. Kung, C. Lo, and R. Foka, "A Toeplitz approximation approach to coherent source direction finding," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, vol. 11, Apr. 1986, pp. 193–196.
[14] R. Roy, A. Paulraj, and T. Kailath, "ESPRIT—A subspace rotation approach to estimation of parameters of cisoids in noise," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 34, no. 5, pp. 1340–1342, Oct. 1986.
[15] B. Ottersten, M. Viberg, and T. Kailath, "Performance analysis of the total least squares ESPRIT algorithm," *IEEE Trans. Signal Process.*, vol. 39, no. 5, pp. 1122–1135, May 1991.

[16] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Trans. Antennas Propag.*, vol. AP-34, no. 3, pp. 276–280, Mar. 1986.
[17] D. Zhang, Y. Zhang, G. Zheng, C. Feng, and J. Tang, "Improved DOA estimation algorithm for co-prime linear arrays using root-MUSIC algorithm," *Electron. Lett.*, vol. 53, no. 18, pp. 1277–1279, Aug. 2017.
[18] C. K. E. Lau, R. S. Adve, and T. K. Sarkar, "Minimum norm mutual coupling compensation with applications in direction of arrival estimation," *IEEE Trans. Antennas Propag.*, vol. 52, no. 8, pp. 2034–2041, Aug. 2004.
[19] Q. Huang and T. Chen, "One-dimensional MUSIC-type algorithm for spherical microphone arrays," *IEEE Access*, vol. 8, pp. 28178–28187, 2020.
[20] H. Wong, M.-M. Papadopoulou, M. Sadooghi-Alvandi, and A. Moshovos, "Demystifying GPU microarchitecture through microbenchmarking," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Mar. 2010, pp. 235–246.
[21] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing," *Proc. IEEE*, vol. 96, no. 5, pp. 879–899, Apr. 2008.
[22] J. Nickolls and W. J. Dally, "The GPU computing era," *IEEE Micro*, vol. 30, no. 2, pp. 56–69, Mar. 2010.
[23] M. W. Majid, T. E. Schmuland, and M. M. Jamali, "Parallel implementation of the wideband DOA algorithm on single core, multicore, GPU and IBM cell BE processor," *Sci. J. Circuits, Syst. Signal Process.*, vol. 2, no. 2, p. 29, 2013.
[24] Z. Lu, L. Zhang, J. Zhang, and J. Zhang, "Parallel optimization of broadband underwater acoustic signal MUSIC algorithm on GPU platform," in *Proc. 4th Int. Conf. Syst. Informat. (ICSAI)*, Nov. 2017, pp. 704–708.
[25] H. Eray and A. Temizel, "Performance analysis of noise subspace-based narrowband direction-of-arrival (DOA) estimation algorithms on CPU and GPU," 2020, *arXiv:2007.14135*. [Online]. Available: http://arxiv.org/abs/2007.14135
[26] Y. Liang, F. He, and X. Zeng, "3D mesh simplification with feature preservation based on whale optimization algorithm and differential evolution," *Integr. Comput.-Aided Eng.*, pp. 1–19, Jan. 2020.
[27] N. Hou, F. He, Y. Zhou, and Y. Chen, "An efficient GPU-based parallel tabu search algorithm for hardware/software co-design," *Frontiers Comput. Sci.*, vol. 14, no. 5, pp. 1–18, Oct. 2020.
[28] Y. Zhou, F. He, N. Hou, and Y. Qiu, "Parallel ant colony optimization on multi-core SIMD CPUs," *Future Gener. Comput. Syst.*, vol. 79, pp. 473–487, Feb. 2018.
[29] Y. Zhou, F. He, and Y. Qiu, "Dynamic strategy based parallel ant colony optimization on GPUs for TSPs," *Sci. China Inf. Sci.*, vol. 60, no. 6, Jun. 2017, Art. no. 068102.
[30] Y. Kim and Y. Park, "CPU-GPU architecture for active noise control," *Appl. Acoust.*, vol. 153, pp. 1–13, Oct. 2019.
[31] Y. Kim and Y. Park, "Blockwise weighted least square active noise control for CPU-GPU architecture," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 28, pp. 951–963, 2020.
[32] C. Sarofeen and P. Gillett, "A high performance parallel and heterogeneous approach to narrowband beamforming," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 8, pp. 2196–2207, Aug. 2016.
[33] F. Yan, M. Jin, and X. Qiao, "Low-complexity DOA estimation based on compressed MUSIC and its performance analysis," *IEEE Trans. Signal Process.*, vol. 61, no. 8, pp. 1915–1930, Apr. 2013.
[34] *Cuda C Programming Guide*, NVIDIA, Santa Clara, CA, USA, Jul. 2020.
[35] J. Cheng, M. Grossman, and T. McKercher, *Professional CUDA C Programming*. 2014.
[36] Y. Wang *et al.*, *Theory and Algorithm of Spatial Spectrum Estimation*. Beijing, China: House Tsinghua Univ., 2004, pp. 38–39.

QINGHUA HUANG received the B.S. degree from the School of Control Science and Control Engineering, Shangdong University, in 2001, the M.S. degree from the Institute of Pattern Recognition, Shangdong University, in 2004, and the Ph.D. degree from the Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University. In 2014, she was a Visiting Scholar with the University of Maryland, College Park, MD, USA, for one year. She is currently an Associate Professor with the School of Communication and Information Engineering, Shanghai University. Her research interests include array signal processing, statistical signal processing, and Bayesian statistical learning.

NAIDA LU was born in Lanzhou, China, in 1996. He received the B.S. degree in communication engineering from the School of Communication and Information Engineering, Shanghai University, China, in 2018, where he is currently pursuing the M.S. degree in signal and information processing. His research interests include array signal processing and GPU parallel optimization.