جهت‌یابی زمان‌حقیقی منابع صوت زیر آب با استفاده از واحد پردازنده گرافیکی

احسان ایمانی‌فر[۱]، امیر اخوان*[۲] و علی‌اصغر آبنیکی[۳]

[۱]دانشکده مهندسی پزشکی-بیوالکتریک، دانشگاه صنعتی امیرکبیر، تهران، ایران

[۲]دانشکده مهندسی برق و کامپیوتر، دانشگاه صنعتی اصفهان، اصفهان، ایران

[۳]مرکز پژوهشی مهندسی دریا، دانشکده مهندسی مکانیک، دانشگاه صنعتی شریف، تهران، ایران

چکیده

جهت‌یابی منابع صوت به‌کمک روش‌های مبتنی بر آرایه فازی، اهمیت فراوانی در حوزه‌های مختلف از جمله سونار، بینایی ربات و تشخیص عیوب مکانیکی دارد. روش‌های شکل‌دهی‌پرتو وفقی، از جمله الگوریتم کمینه‌واریانس بدون‌اعوجاج از قدرت تفکیک بالایی نسبت به روش‌های غیروفقی برخوردار هستند؛ اما این برتری در ازای پیچیدگی محاسباتی این الگوریتم‌ها به‌دست آمده است. این مسأله باعث می‌شود در کاربردهایی که نیاز به جهت‌یابی زمان‌حقیقی منبع صوت دارند، به‌ندرت از این الگوریتم‌ها استفاده شود. از سوی دیگر، یک ویژگی مهم روش‌های شکل‌دهی‌پرتو وفقی از جمله کمینه‌واریانس، پتانسیل بالای این الگوریتم‌ها برای موازی‌سازی است. هدف این مقاله، پیاده‌سازی موازی الگوریتم کمینه‌واریانس با به کارگیری واحد پردازنده گرافیکی (GPU) به جای واحد پردازنده مرکزی (CPU)، به‌منظور افزایش سرعت اجرا و رسیدن به حالت زمان‌حقیقی است. برای دست‌یابی به این هدف از مدل برنامه‌نویسی کودا برای پیاده‌سازی الگوریتم بر روی پردازنده گرافیکی استفاده شده است. به‌منظور بررسی عملکرد پیاده‌سازی موازی الگوریتم کمینه‌واریانس، دو مدل GPU متفاوت و همچنین CPU به‌کاربرده شده است. صحت عملکرد پیاده‌سازی‌های مختلف در این مقاله به‌وسیلهٔ داده‌های واقعی سونار و همچنین داده‌های شبیه‌سازی تأیید شد. نتایج نشان می‌دهد که می‌توان با استفاده از یک آرایه ۶۴ حس‌گره، جهت منابع صوت زیر آب را با استفاده از الگوریتم کمینه‌واریانس به‌صورت زمان‌حقیقی و با قدرت تفکیک بالا تخمین زد.

واژگان کلیدی: جهت‌یابی منابع صوت، الگوریتم کمینه‌واریانس، پردازش موازی، واحد پردازنده گرافیکی، مدل برنامه‌نویسی کودا.

Real-Time DOA Estimation of Underwater Sound Sources Using GPU

Ehsan Imanifar[1], Amir Akhavan[2*] & Ali Asghar Abniki[3]

[1]Department of Biomedical Engineering-Bioelectric, Amirkabir University of Technology, Tehran, Iran

[2]Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran

[3] Center of Excellence in Hydrodynamics and Dynamic of Marine Vehicles, Department of Mechanical Engineering, Sharif University of Technology, Tehran, Iran

Abstract

Direction of Arrival (DOA) estimation of sound sources using phased array-based methods has a lot of importance in various fields, including sonar, robot vision, and mechanical defect detection. Adaptive beamforming methods, such as the MVDR (Minimum Variance Distortionless Response) algorithm, have high resolution compared to non-adaptive methods (Delay and Sum algorithm); but this advantage is achieved in return for the computational complexity of these algorithms. This makes it hard to use these algorithms in applications that require real-time sound source DOA estimation. On the other hand, an important feature of the adaptive beamforming methods including MVDR is the high potential of these algorithms for parallelization. The purpose of this paper is the parallel implementation of the MVDR algorithm by employing Graphical Processor Unit (GPU) instead of Central Processor Unit (CPU) to increase the execution speed and achieve the real-time mode. For this purpose, the CUDA

* Corresponding author

*نویسندهٔ عهده‌دار مکاتبات

●تاریخ ارسال مقاله: ۱۳۹۸/۱/۱۱ ●تاریخ پذیرش: ۱۳۹۹/۲/۲ ●تاریخ انتشار: ۱۴۰۰/۰۷/۱۷ ●نوع مطالعه: پژوهشی

(Compute Unified Device Architecture) programming model has been used to implement the algorithm on the GPU. CUDA is a parallel computing platform and application programming interface (API) model created by Nvidia. It allows software developers to use a CUDA-enabled GPU for parallel processing. In order to investigate the performance of parallel implementation of the MVDR algorithm, two different GPUs, as well as CPUs, have been used. The performance validity of various implementations in this paper was confirmed by real sonar data as well as simulation data. The results show that using an array of 64 sensors, it is possible to estimate the DOA of underwater sound sources in real-time mode and with high resolution using the MVDR algorithm.

Keywords: DOA estimation of sound sources, MVDR algorithm, Parallel processing, GPU, CUDA.

۱- مقدمه

یکی از موضوعاتی که امروزه ذهن بسیاری از پژوهش‌گران را به خود مشغول کرده است، مسأله موقعیت‌یابی و جهت‌یابی یک شیء و یا یک هدف است. یکی از اقسام جهت‌یابی، جهت‌یابی صوتی است که هدف آن، ره‌گیری جسمی است که منبع سیگنال صوت بوده و یا صوت ارسالی را منعکس می‌کند.

روش‌های جهت‌یابی صوت اغلب روش‌های مبتنی بر آرایه فازی و شکل‌دهی‌پرتو [1–3] هستند. در این نوع جهت‌یابی، از سیگنال صوت دریافتی، به‌منظور تعیین جهت دقیق منبع استفاده می‌شود. به این شکل که در آن، جهت‌یابی بر اساس تغییرات سیگنال منتشرشده از هدف و دریافت آن در آرایه‌ای از حس‌گرها و بررسی تفاوت‌های سیگنال‌های دریافت‌شده انجام می‌گیرد. پایه اصلی روش‌های شکل‌دهی‌پرتو، جبران مناسب تأخیرهای زمانی ای است که امواج هنگام رسیدن به چند گیرنده جدا ازهم دچار آن می‌شوند.

از مهم‌ترین روش‌های جهت‌یابی می‌توان به تأخیر و جمع (DAS[1]) [4]، کمینه‌واریانس بدون‌اعوجاج (MVDR[2])[4]، طبقه‌بندی سیگنال چندگانه (MUSIC[3])[5] و روش‌های مبتنی بر تنک‌بودن [4][6,7] اشاره کرد. در این میان کمینه‌واریانس از معروف‌ترین روش‌های جهت‌یابی است. الگوریتم کمینه‌واریانس در سال ۱۹۶۷ توسط Capon معرفی شد [8]. جهت‌یابی با استفاده از این روش، کاربرد زیادی در تصویربرداری با شیوه‌های مختلف از جمله سونار [9]، فراصوت [10–13] و رادار [14] دارد. این الگـوریـتم جزو دستهٔ شکل‌دهنده‌های پرتو وفقی است؛ به این معنی که جهت‌یابی در آن متناسب با ویژگی‌های سیگنال رسیده به حس‌گرها انجام می‌شود. شکل‌دهنده‌های پرتو وفقی نسبت به الگوریتم‌های غیروفقی، قدرت تفکیک بالاتری دارند؛ اما به‌دلیل حجم و پیچیدگی زیاد محاسـبات در این شکل‌دهنده‌ها، زمان اجرای آن‌ها نیز بیشتر شده و مانع تحقق ویژگی زمان‌حقیقی‌بودن این الگوریتم‌ها می‌شود [13]. کاربرد سونار یکی از حوزه‌هایی است که جهت‌یابی زمان‌حقیقی منبع صوت در آن ضروری است؛ بنابراین در این مقاله تلاش شده است که داده‌های واقعی ثبت‌شدهٔ سونار را پردازش کرده و راستای منابع صوت زیر آب را به‌صورت زمان‌حقیقی تعیین کنم. پردازش زمان‌حقیقی الگوریتم‌های جهت‌یابی منجر به حذف تأخیر در دریافت نتیجه جهت‌یابی و به‌روزبودن همیشگی اطلاعات خروجی آن می‌شود. پیچیدگی محاسباتی الگوریتم کمینه‌واریانس ناشی از محاسبه بردار وزن موردنیاز به‌منظور شکل‌دهی پرتو دریافت شده است. چنانچه تعداد حس‌گرهای آرایه با M نمایش داده شود، پیچیدگی الگوریتم DAS و کمینه‌واریانس به‌ترتیب از مرتبه M و M^3 است. علت این پیچیدگی محاسباتی در الگوریتم کمینه‌واریانس، نیاز به تخمین ماتریس کواریانس[5] به ابعاد M×M و محاسبه معکوس آن به‌ازای هر جهت خاص است. با توجه به عملکرد مناسب الگوریتم کمینه‌واریانس، در مقاله پیش‌رو از این روش به‌عنوان روش مبنا استفاده می‌شود.

تاکنون روش‌های زیادی برای کاهش پیچیدگی الگوریتم کمینه‌واریانس ارائه شده‌اند. در روش‌های مبتنی بر فضای پرتو[6] از یک ماتریس تبدیل متعامد برای تصویرکردن داده‌ها به یک فضای با بُعد پایین و در نتیـجه کاهش حجم محاسبات استفاده می‌شود [15,16]. در [17] تعدادی پنجره وزن‌دار از پیش تعیین‌شده در نظر گرفته و از بین آنها، پنجره‌ای که کمترین توان خروجی را ایجاد می‌کند انتخاب می‌شود. به این ترتیب پیچیدگی الگوریتم کمینه‌واریانس پیشنهادشده در این مقاله، تا P برابرِ پیچیدگی الگوریتم DAS کاهش یافته و P تعداد پنجره‌های وزن‌دار از پیش تعیین‌شده است. در [18] از تحلیل مؤلفه‌های اصلی[7] برای کاهش پیچیدگی محاسبه بردار وزن بهینه در روش کمینه‌واریانس استفاده شده است. رویکردهای دیگری نیز در [19–21] مطرح شده‌اند که هر یک از آنها فرض‌ها و یا تقریب‌هایی را برای کاهش پیچیدگی محاسباتی الگوریتم کمینه‌واریانس استفاده

[1] Delay and Sum
[2] Minimum Variance Distortionless Response
[3] Multiple Signal Classification
[4] Sparsity based methods
[5] Covariance matrix
[6] Beam Space
[7] Principal Component Analysis (PCA)

کرده‌اند؛ در نتیجه هیچکدام دقت و وضوح الگوریتم اصلی معرفی‌شده توسط کَپُن[1] را ندارند؛ لذا هدف اصلی از این مـقاله افزایش سرعت پیاده‌سازی الگوریتم کمینه‌واریانس به‌منظور اجرای زمان‌حقیقی آن است. در این راستا از رویکرد برنامه‌نویسی موازی به‌منظور رسیدن به سرعت اجرای قابل‌قبول استفاده شده است [22].

یکی از حوزه‌های پردازشی که قابلیت موازی‌سازی زیادی دارد، شکل‌دهی‌پرتو است؛ به همین دلیل برای رسیدن به هدف اصلی این مقاله، از قدرت پردازش موازی در واحدهای پردازنده گرافیکی (GPUها) استفاده می‌کنیم. تعداد هسته‌های پردازشی واحد پردازنده گرافیکی، چندین برابر بیشتر از تعداد این هسته‌ها در واحد پردازنده مرکزی (CPU) است؛ اما قدرت پردازشی هر هسته GPU بسیار کمتر از یک هسته CPU است [23]. در این مقاله به‌منظور پردازش موازی الگوریتم کمینه‌واریانس روی GPU از مدل برنامه‌نویسی کودا (CUDA[2]) استفاده می‌کنیم. کودا در ماه ژوئن سال ۲۰۰۷ به‌وسیله شرکت انویدیا معرفی شد [23] و تا به امروز توابع و جعبه‌ابزار[3]های کاربردی بسیاری به آن اضافه شده‌اند.

ادامه مقاله به این شکل است. در بخش ۲ الگوریتم کمینه‌واریانس و روابط حاکم بر آن بیان می‌شود. بخش ۳ به معرفی واحد پردازنده گرافیکی و مدل برنامه‌نویسی کودا می‌پردازد؛ در ادامه در بخش ۴ نحوه پیاده‌سازی الگوریتم کمینه‌واریانس بر روی GPU بیان و سپس نتایج به‌دست‌آمده از اجرای این الگوریتم روی داده‌های واقعی و شبیه‌سازی‌شده در بخش ۵ نمایش داده و در انتها نتیجه‌گیری ارائه می‌شود.

۲- شکل‌دهی‌پرتو به روش کمینه واریانس

با توجه به این‌که روش DAS مبنای روش کمینه‌واریانس است در ادامه به‌اختصار روش DAS در حوزه فرکانس بیان می‌شود و سپس به ارائه روش شکل‌دهی‌پرتو کمینه واریانس می‌پردازیم. فرض کنید یک آرایه متشکل از M حس‌گر داریم که در موقعیت‌های $\{r_1, r_2, \dots, r_M\}$ قرار گرفته‌اند. در الگوریتم‌های شکل‌دهی‌پرتو، خروجی حس‌گر mاُم را به‌صورت رابطه (۱) در نظر می‌گیریم:

$$\hat{y}_m(t) = s_m(t-\tau_m) + n_m(t-\tau_m) \qquad (۱)$$

که در آن s سیگنال دلخواه، n نویز موجود در محیط، τ_m و t به‌ترتیب تأخیر زمانی سیگنال رسیده به حس‌گر mاُم و شاخص زمان هستند. خروجی شکل‌دهنده پرتو (B) در حوزه فرکانس، طبق رابطه (۲) به‌دست می‌آید (شکل-۱):

$$B(\omega,\theta) = \sum_{m=1}^{M} w_m^*(\omega,\theta)\cdot y_m(\omega) = w^H(\omega,\theta)\times y(\omega) = w^H y \qquad \dots. \quad (۲)$$

که در آن w_m وزن مربوط به حس‌گر mاُم، y_m خروجی حس‌گر mاُم در حوزه فرکانس، $w=[w_1,w_2,\dots,w_M]^T$ بردار وزن‌ها، $y=[y_1,y_2,\dots,y_M]^T$ بردار خروجی حس‌گرها، $(.)^*$ و $(.)^H$ به‌ترتیب نشان‌دهندهٔ مزدوج و هِرمیتیَن[4] هستند و ω و θ به‌ترتیب شاخص فرکانس زاویه‌ای و شاخص زاویه هستند.

الگوریتم DAS از بیشینه توان خروجی شکل‌دهنده پرتو به‌ازای زوایای مختلف، به‌منظور تخمین راستای منبع سیگنال دریافتی استفاده می‌کند [3]. توان خروجی شکل دهنده‌پرتو در حالت کلی، طبق رابطه (۳) به‌دست می‌آید:

$$P_{out}(\omega,\theta) = E(|B(\omega,\theta)|^2) = w^H R w \qquad (۳)$$

که در آن E نماد امید ریاضی و R ماتریس کواریانس مکانی آرایه است و به‌صورت رابطه (۴) تعریف می‌شود.

$$R(\omega) = E(y\times y^H) \qquad (۴)$$

الگوریتم DAS یک روش شکل‌دهی‌پرتو غیروفقی است، در این الگوریتم بردار وزن‌ها (w) برابر بردار جهت‌دهی (a) است که به‌صورت رابطه زیر است:

$$w_{DAS}(\omega,\theta) = [e^{-j\omega\tau_1}, e^{-j\omega\tau_2}, \dots, e^{-j\omega\tau_m}]^T \qquad (۵)$$

در شکل‌دهنده‌پرتو کمینه‌واریانس، بردار وزن‌ها متناسب با سیگنال‌های دریافتی به‌دست می‌آیند. در این الگوریتم بردار وزن‌ها به گونه‌ای انتخاب می‌شود که توان خروجی شکل‌دهنده کمینه شده، به شرطی که توان سیگنال اصلی در خروجی تضعیف نشود این مسأله به زبان ریاضی به صورت رابطه (۶) بیان می‌شود [24].

$$min\ (w^H R w)\ \ s.t.\ w^H a = 1 \qquad (۶)$$

با حل این مسأله کمینه‌سازی مقید توسط روش ضرایب لاگرانژ، بردار وزن بهینه الگوریتم کمینه‌واریانس به‌صورت رابطه (۷) تعیین می‌شود [25]:

$$w_{MVDR}(\omega,\theta) = \frac{R^{-1}a}{a^H R^{-1} a} \qquad (۷)$$

برای رفع مشکل تکینگی[5] ماتریس کواریانس و همچنین کاهش حساسیت الگوریتم کمینه‌واریانس به خطای بردار

[1] Capon
[2] Compute Unified Device Architecture
[3] Toolbox

[4] Hermitian
[5] Singularity

جهت‌دهی [26]، از بارگذاری روی درایه‌های قطری به‌صورت زیر استفاده می‌شود:

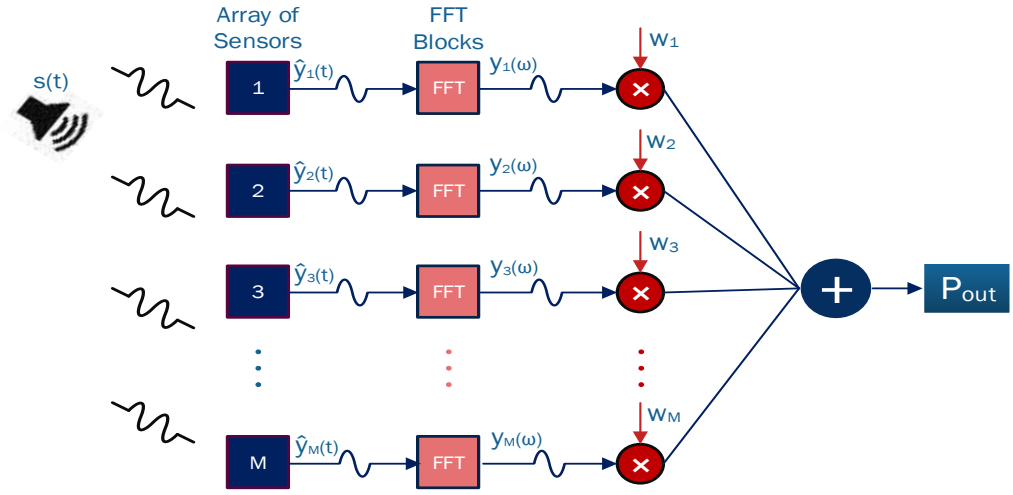
$$\hat{R}(\omega) = R(\omega) + I \cdot \varepsilon \cdot trace\{R(\omega)\} \qquad (۸)$$

که در آن I ماتریس همانی، ε یک ضریب از پیش تعیین‌شده و $trace\{.\}$ عملگر محاسبه‌گر مجموع مقادیر قطر اصلی است.

توان خروجی شکل‌دهنده پرتو در الگوریتم کمینه‌واریانس طبق رابطه‌های (۳) و (۷) به‌صورت رابطه (۹) ساده می‌شود:

$$P_{out\_MVDR}(\omega,\theta) = \frac{1}{a^H \hat{R}^{-1} a} \qquad (۹)$$

رابطه (۹) طیف توان فرکانسی-زاویه‌ای الگوریتم کمینه‌واریانس را نشان می‌دهد.



(شکل-۱): نمایش ساختار شکل‌دهنده پرتو در حوزه فرکانس

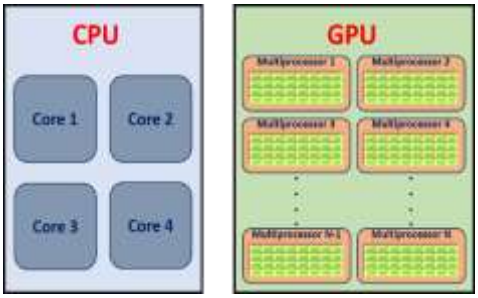
(Figure-1): Exhibition of the beamformer structure in frequency domain

با استفاده از بردار وزن به‌دست‌آمده در روش کمینه‌واریانس (رابطه ۷) توان نوفه در خروجی شکل‌دهنده‌پرتو کمینه می‌شود. گفتنی است که به‌طور کلی نوفه موجود در سامانه‌های جهت‌یابی صوت را می‌توان به دو نوعِ نوفه‌های جهت‌دار (ناشی از منابع صوت غیر هدف) و نوفه‌های تصادفی (نوفه‌های بدون جهت) تقسیم بندی کرد. در الگوریتم پیاده‌سازی‌شده در این مقاله دو نوع فیلتر برای کاهش اثر نوفه‌های یادشده وجود دارد. نخستین فیلتر در حوزه فرکانس اعمال شده است؛ بدین صورت که در پردازش داده ثبت‌شده در هر زیرباند، سهم نوفه‌های خارج از آن محدوده فرکانسی به‌طور کامل حذف شده است. به عبارت دیگر پیاده‌سازی الگوریتم کمینه‌واریانس در حوزه فرکانس به‌طور ذاتی باعث اعمال فیلترهای فرکانسی می‌شود. چنین فیلترهایی اثر هر دو نوع نوفه جهت‌دار و بدون جهت را در خروجی الگوریتم کاهش می‌دهد. دومین سازوکار برای حذف نوفه‌های موجود در داده‌های ثبت‌شده، فیلترهای مکانی هستند. با توجه به این‌که روش کمینه‌واریانس یک نوع فیلتر مکانی است، به هنگام تمرکز روی هر زاویه، نوفه‌های جهت‌دار موجود در زوایای دیگر به‌شدت تضعیف می‌شوند.

۳- واحد پردازنده گرافیکی

ویژگی اصلی GPUها تعداد هسته‌های پردازشی آنها است؛ شکل (۲) تفاوت اصلی میان معماری داخلی GPU و CPU را نشان می‌دهد.

همان‌طورکه در شکل مشخص است، GPU از هسته‌های پردازشی بسیار بیشتری نسبت به یک CPU تشکیل‌شده است و این ویژگی، قابلیت پردازش موازی را به GPUها می‌دهد. اگرچه CPUها از هسته‌های پردازشی قوی‌تری بهره‌مند هستند، ولی در شرایطی که بخواهیم عملیاتی با محاسبات کم را به‌صورت مکرر انجام دهیم، استفاده از GPU و پردازش موازی از نظر زمان و سرعت انجام عملیات بسیار به‌صرفه‌تر است.



(شکل-۲): مقایسه معماری داخلی CPU و GPU

(Figure-2): CPU/GPU interior architecture comparison

۳-۱- مدل برنامه‌نویسی CUDA

کودا بستری برای محاسبات موازی همه‌منظوره است که رابط‌های برنامه‌نویسی را در اختیار برنامه‌نویسان قرار می‌دهد و به طراحان نرم‌افزار اجازه می‌دهد تا از توانایی‌های GPUهای ساختهٔ شرکت انویدیا در جهت محاسبات همه‌منظوره روی واحد پردازش گرافیکی استفاده کنند. این بستر به‌گونه‌ای طراحی شده است که با زبان‌های برنامه‌نویسی C، C++، Fortran و MATLAB کار می‌کند [23].

شیوه برنامه‌نویسی کودا به‌گونه‌ای است که امکان برنامه‌نویسی روی هردو پردازندهٔ GPU و CPU را به برنامه‌نویس می‌دهد؛ قسمت‌های سریال برنامه بر روی CPU اجرا می‌شود، سپس برنامه به GPU منتقل شده و پس از اجرای عملیات مربوط به قسمت‌های موازی، دوباره به CPU بازمی‌گردد. بخش‌های موازی برنامه در یک تابع هسته[1] نوشته می‌شوند و هر هسته توسط تعداد زیادی نخ[2] به‌صورت موازی اجرا می‌شود. سازماندهی نخ‌ها در کودا توسط بلوک‌ها و گرید[3]ها انجام می‌شود. هر بلوک شامل یک آرایه سه‌بعدی از نخ‌ها است؛ بنابراین هر نخ در هر بلوک توسط موقعیت مکانی (x، y و z) شناخته می‌شود. هر گرید نیز یک آرایه دوبعدی از بلوک‌ها است و همه بلوک‌ها در یک گرید دارای ابعاد یکسانی هستند. در فراخوانی هر هسته، برنامه‌نویس می‌تواند تعداد بلوک‌ها و تعداد نخ‌ها در هر بلوک را برای اجرای آن هسته انتخاب نماید.

کودا برای تعریف توابع، سه کلیدواژه مهم دارد. کلیدواژه __global__ به این معنی است که تابع معرفی شده یک تابع هسته در کودا است؛ این تابع بر روی دستگاه[4] اجرا می‌شود و تنها از میزبان[5] می‌تواند یک شبکه از نخ‌ها را بر روی دستگاه ایجاد کند. کلیدواژه __device__ نیز یک تابع دستگاه را نشان می‌دهد؛ این تابع در دستگاه اجرا می‌شود و تنها از یک تابع هسته یا یک تابع دستگاه دیگر می‌توان آنرا فراخوانی کرد و درنهایت کلیدواژه __host__ نشان‌دهندهٔ یک تابع میزبان است که در میزبان اجرا می‌شود و تنها ازطریق یک تابع میزبان دیگر قابل فراخوانی می‌باشد.

[1] Kernel
[2] Thread
[3] Grid
[4] Device
[5] Host

۴- پیاده‌سازی کمینه‌واریانس روی GPU

یک ویژگی مهم الگوریتم کمینه‌واریانس قابلیت موازی‌سازی آن است. به‌این‌ترتیب که می‌توان توان خروجی شکل‌دهنده‌پرتو در هر زاویه و فرکانس خاص را به‌طور مستقل محاسبه کرد؛ درنتیجه قادر خواهیم بود به کمک GPU و پردازش موازی، سرعت انجام محاسبات را افزایش دهیم. گفتنی است، اگرچه پیاده‌سازی الگوریتم کمینه‌واریانس در حوزه زمان دارای پیچیدگی محاسباتی کمتری نسبت به پیاده‌سازی آن در حوزه فرکانس است؛ ولی با اجرای الگوریتم در حوزه زمان اطلاعات فرکانسی تا حد زیادی از بین می‌رود. این در حالی است که با استفاده از پیاده‌سازی به‌کاربرده‌شده در این مقاله، نتیجه جهت‌یابی منبع صوت در همه مؤلفه‌های فرکانسی به‌صورت هم‌زمان در دسترس خواهد بود؛ بنابراین در شرایطی که طیف فرکانسی منبع صوت زیر آب به‌طور دقیق مشخص نباشد با استفاده از پیاده‌سازی روش کمینه واریانس در حوزه فرکانس می‌توان علاوه‌بر اطلاعات مربوط به راستای منابع صوت به اطلاعات فرکانسی آنها نیز دست یافت.

در این مقاله برای پردازش موازی الگوریتم کمینه‌واریانس از مدل برنامه‌نویسی کودا و زبان C++ استفاده می‌شود. شکل (۳) شمای کلی ساختار طراحی‌شده در این مقاله را به‌منظور پیاده‌سازی الگوریتم کمینه‌واریانس نشان می‌دهد. همان‌طورکه اشاره شد، در کودا ابتدا برنامه روی CPU اجرا می‌شود تا دسترسی لازم به قسمت‌های موازی فراهم شود. در این بخش ابتدا بازه فرکانسی و زاویه‌ای مورد نظر برای جهت‌یابی منبع سیگنال تعیین، سپس سیگنال‌های همه حس‌گرها به صورت یک ماتریس به حافظه گلوبال GPU منتقل می‌شود. در این مقاله پیاده‌سازی الگوریتم کمینه‌واریانس توسط GPU در پنج بخش متوالی انجام می‌شود.

در نخستین بخش، پس از انتقال سیگنال‌ها به GPU به‌کمک جعبه‌ابزار cuFFTW در کودا، سیگنال به حوزه فرکانس منتقل می‌شود.

در بخش دوم به‌وسیله یک تابع هسته، ماتریس کواریانس مکانی طبق رابطه (۴) محاسبه می‌شود. با توجه به اینکه ماتریس کواریانس، یک ماتریس هرمیتی است؛ به جهت کاهش محاسبات اضافی، تنها درایه‌های بالامثلثی این ماتریس محاسبه می‌شود. در این هسته به‌ازای هر مؤلفه‌فرکانسی[6]، یک بلوک و به هر بلوک، به اندازه تعداد حس‌گرها، نخ اختصاص داده می‌شود؛ به این شکل که هر

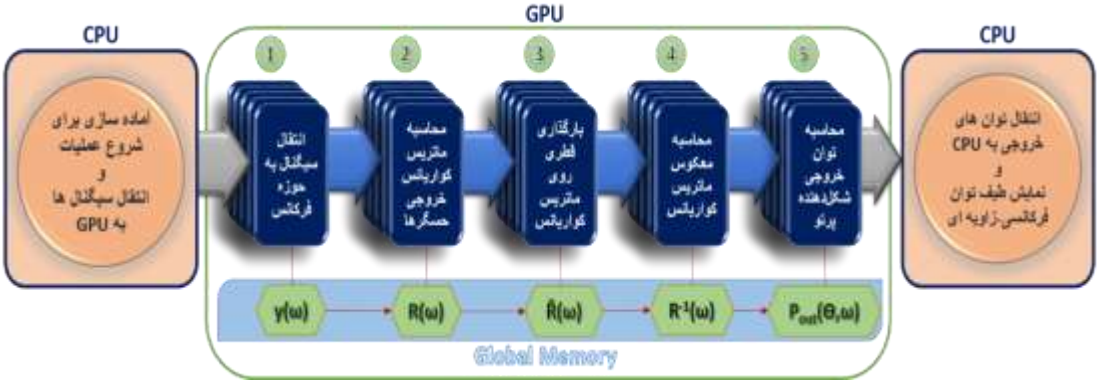
[6] Frequency Bin

نخ، یک ستون از ماتریس کواریانس و یک بلوک از نخ‌ها، کل ماتریس کواریانس مربوط به یک فرکانس خاص را محاسبه می‌کند.

در بخش سوم، طبق رابطه (۸) در یک تابع هسته به بارگذاری قطری روی ماتریس کواریانس پرداخته می‌شود. به این هسته به تعداد مؤلفه‌های‌فرکانسی، نخ اختصاص داده و هر نخ، بارگذاری قطری را روی ماتریس کواریانس مربوط به یک فرکانس انجام می‌دهد.

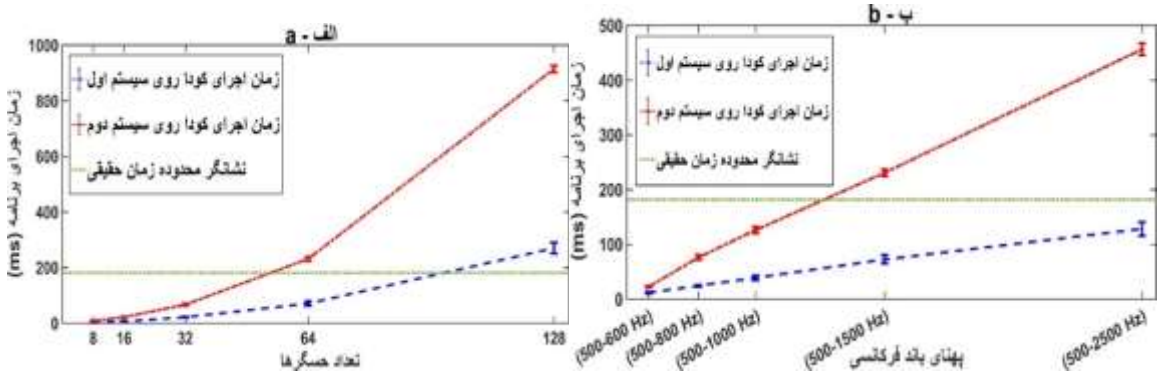
بخش چهارم مربوط به محاسبه معکوس ماتریس کواریانس مکانی است. در این مقاله از جعبه‌ابزار cuBLAS به‌منظور محاسبه معکوس ماتریس کواریانس مکانی استفاده شده است. دلیل استفاده از این جعبه‌ابزار، دارابودن قابلیت معکوس‌گیری به‌صورت گروهی، از تعداد زیادی ماتریس با ابعاد یکسان و همچنین قابلیت معکوس‌گیری از ماتریسی با درایه‌های مختلط است. در این جعبه ابزار از روش LU برای محاسبه معکوس ماتریس‌ها استفاده می‌شود. این کار در دو مرحله انجام می‌شود [27]. در مرحله نخست تمامی ماتریس‌های داده‌شده به جعبه‌ابزار، به ماتریس‌های بالامثلثی و پایین مثلثی به‌صورت $R = LU$ تجزیه می‌شوند و در مرحله دوم نیز، با حل معادله رابطه (۱۰) ماتریس‌های معکوس محاسبه می‌شود.

$$LUR^{-1} = I \qquad (۱۰)$$



(شکل-۳): نمودار بلوکی پیاده‌سازی کمینه‌واریانس روی واحد پردازنده گرافیکی

(Figure-3): Block diagram of the MVDR implementation on the GPU



(شکل-۴): نمایش تغییرات زمان اجرای برنامه با کودا روی سیستم اول و کودا روی سیستم دوم

الف) تغییرات زمان اجرای برنامه با تغییر تعداد حسگرها

ب) تغییرات زمان اجرای برنامه با تغییر بازه‌فرکانسی

(Figure-4): Displaying the changes in the program's runtime with CUDA on GPU-1 and CUDA on GPU-2
a) Changes in the program's runtime by changing the number of sensors
b) Changes in the program's runtime by changing the frequency range

در بخش پنجم نیز از یک تابع هسته برای محاسبه توان خروجی شکل‌دهنده پرتو استفاده می‌شود. از آنجایی که هدف نهایی، جستجو برای منبع سیگنال در همه فرکانس‌های موجود در بازه مدنظر و همه زوایای صفر تا ۳۵۹ درجه است؛ به تعداد (۳۶۰*تعداد مؤلفه‌های فرکانسی) نخ به این هسته اختصاص داده می‌شود و هر ۵۱۲ نخ در یک بلوک قرار می‌گیرد. در این هسته، هر نخ توان سیگنال خروجی مربوط به یک فرکانس خاص و یک زاویه خاص را محاسبه می‌کند. برای محاسبه توان خروجی، پس از محاسبه بردار جهت‌دهی، از رابطه (۹)

استفاده می‌شود. باتوجه به اینکه توان خروجی، یک عدد حقیقی است، محاسبات مربوط به قسمت‌های موهومی حذف می‌شود. در انتها نتیجه جهت‌یابی به‌صورت یک ماتریس سه‌بعدی به CPU منتقل می‌شود.

۵- نتایج و بحث

نتایج به‌دست‌آمده در این مقاله به دو بخش تقسیم می‌شوند. در بخش اول نتایج مربوط به داده‌های شبیه‌سازی ارائه می‌شود و در بخش دوم نتایج مربوط به داده‌های واقعی ثبت‌شده ارائه می‌شود.

۵-۱- نتایج داده‌های شبیه‌سازی

در این مقاله به‌منظور مقایسه زمان لازم برای جهت‌یابی صوت به روش کمینه‌واریانس، از دو مدل سخت‌افزار متفاوت با مدل‌های زیر استفاده شده است:

CPU-1: Intel Core i7 7700HQ
GPU-1: NVIDIA GeForce GTX 1060
CPU-2: Intel Core i7 6700HQ
GPU-2: NVIDIA GeForce GTX 950m

زمان لازم برای جهت‌یابی صوت بر روی هر سگمنت داده، وابسته به عواملی چون سخت‌افزار مورد استفاده، تعداد حس‌گرها و پهنای‌باند فرکانسی (متناسب با آن تعداد مؤلفه‌های فرکانسی) است. در نمودارهای شکل (۴) تأثیر این عوامل بر روی زمان اجرای الگوریتم جهت یابی با استفاده از داده‌های شبیه‌سازی نشان داده شده است. نمودار "الف" زمان لازم برای پردازش یک پنجره ۱۸۲ میلی‌ثانیه‌ای برحسب تعداد حس‌گرهای آرایه به‌ازای پهنای باند یک کیلوهرتز (در بازه‌فرکانسی ۵۰۰-۱۵۰۰ هرتز) را نمایش می‌دهد. گفتنی است که طول زمانی پنجره در داده‌های شبیه‌سازی به‌دلیل شباهت با داده‌های واقعی این‌گونه انتخاب شده است.

با توجه به طول هر پنجره داده، تا هنگامی که زمان مورد نیاز برای پردازش هر پنجره کمتر از ۱۸۲ میلی‌ثانیه باشد، الگوریتم دارای قابلیت اجرای زمان‌حقیقی است. با افزایش تعداد حس‌گرها عرض لوب اصلی کاهش یافته و دقت جهت‌یابی نیز افزایش می‌یابد؛ پس تا زمانی که از حالت زمان‌حقیقی (۱۸۲ میلی‌ثانیه) خارج نشویم، می‌توانیم به‌منظور افزایش دقت، تعداد حس‌گرها را بیشتر کنیم؛ بنابراین با توجه به نتایج شکل (۴-الف)، GPU-1 و GPU-2 به‌ترتیب می‌توانند ۶۴ حس‌گر و ۳۲ حس‌گر را به‌صورت زمان حقیقی پردازش کنند. در شکل (۴-ب) تاثیر پهنای باند پردازش به‌ازای یک آرایه ۶۴ حس‌گره نمایش داده شده است. روشن است که با افزایش پهنای باند فرکانسی، زمان اجرای برنامه افزایش می‌یابد. افزایش پهنای‌باند فرکانسی سامانه این امکان را فراهم می‌کند که منابع مختلف با محتوای فرکانسی متفاوت جهت‌یابی شوند. با توجه به شکل (۴-ب)، GPU-2 تنها تا پهنای‌باند پانصد هرتز را می‌تواند به‌صورت زمان‌حقیقی پردازش کند و GPU-1 تا انتهای نمودار از محدوده زمان‌حقیقی خارج نشده است. این پردازنده تا پهنای‌باند سه‌هزار هرتز را نیز می‌تواند به‌صورت زمان‌حقیقی اجرا کند.

سناریوی دوم در حین چرخش منبع صوت حول یک نقطه، غیر از مرکز آرایه می‌باشد؛ در این سناریو جهت‌یابی در دو بازه فرکانسی مختلف انجام شده است. نمودارهای شکل (۶) نتایج خروجی در این سناریو را نشان می‌دهند. همان‌طورکه در این شکل مشاهده می‌شود با افزایش فرکانس رزولوشن زاویه‌ای بهبود می‌یابد، درحالی‌که ابهام ناشی از الیاسینگ[1] مکانی نیز افزایش می‌یابد.

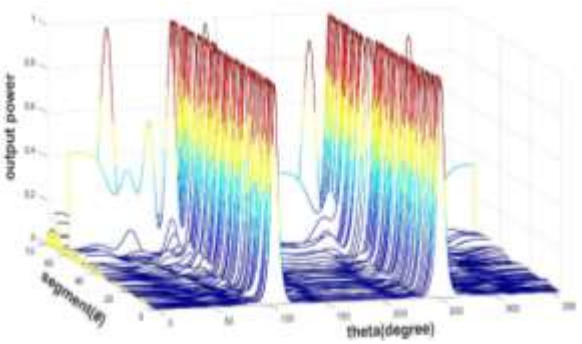
شکل (۷) نتایج جهت‌یابی در سناریوی سوم و در دو بازه فرکانسی مختلف را نشان می‌دهد. در این حالت منبع صوت حول مرکز آرایه روی دایره‌ای به شعاع یک کیلومتر می‌چرخد. به‌منظور مقایسه زمان اجرای برنامه در جهت‌یابی داده‌های واقعی، پردازش داده‌های ثبت شده در سناریوی سوم در هردو سخت‌افزار موجود انجام شده است. جدول (۱) زمان متوسط صرف‌شده برای پردازش داده‌های ثبت‌شده در سناریوی سوم (بازه فرکانسی ۲۵۰۰-۳۰۰۰ هرتز) و همچنین زمان لازم برای پردازش پنج مرحله اصلی برنامه را به‌طور مجزا نشان می‌دهد. این زمان‌ها برای سه حالت مختلف اجرا در متلب روی CPU-1، اجرا با کودا روی GPU-1 و GPU-2 گزارش شده است. گفتنی است که نتایج به‌دست‌آمده در جدول (۱) مربوط به زمان متوسط پردازش ۱۱۴۶ پنجره ۱۸۲ میلی‌ثانیه‌ای و همچنین زمان متوسط پردازش یک پنجره است. همانطورکه مشخص است، سرعت اجرا در کودا چندین برابر بیشتر از سرعت اجرا در متلب است و در هر مرحله، مقدار نسبت برتری متفاوت است. به‌منظور تشخیص بهتر نسبت برتری سرعت اجرای کودا به متلب و همچنین کودای سیستم نخست به کودای سیستم دوم، نمودارهای شکل (۸) ارائه شده است.

[1] Aliasing

۲-۵- نتایج داده‌های واقعی

داده‌های واقعی در این مقاله از یک آرایه هشت حس‌گره ثبت شده است. داده‌های این سامانه سونار با فرکانس ۱۰ کیلوهرتز نمونه‌برداری و در ادامه این داده‌ها به پنجره‌های بدون هم‌پوشانی ۱۸۲ میلی‌ثانیه‌ای (۱۸۲۰ نمونه) تقسیم و الگوریتم جهت‌یابی بر روی هر پنجره اعمال می‌شود. داده‌های سونار این مقاله در سه سناریوی مختلف ثبت شده است. در سناریوی نخست منبع صوت زیر آب ثابت بوده و در سناریوی دوم و سوم منبع صوت (شناور) در حال حرکت روی یک دایره بوده است. در ادامه این بخش نتایج جهت‌یابی با استفاده از روش پیاده‌سازی شده در مقاله ارائه می‌شود.

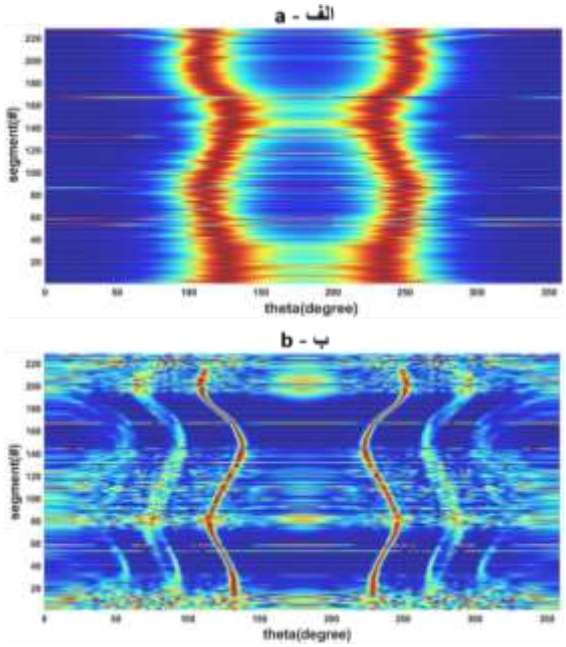
شکل (۵) نتیجه اجرای الگوریتم روی یک منبع صوت ثابت (سناریوی نخست) در فاصله و زاویه‌ای معین از آرایه را نشان می‌دهد. تخمین زاویه منبع سیگنال صوت در زمان‌های مختلف با توجه به بیشترین دامنه توان خروجی به‌دست می‌آید. علت وجود دو پیک[1] متقارن در خروجی، خطی‌بودن آرایه حس‌گرها است.



(شکل-۵): نمودار آبشاری نتیجه جهت‌یابی در سناریوی نخست (جستجو در تمامی زوایا و بازه فرکانسی ۹۵۰-۱۰۵۰ هرتز)

(Figure-5): Waterfall chart of the DOA result in first mode

GPU-1 و GPU-2 به‌ترتیب دارای ۱۲۸۰ و ۶۴۰ هسته پردازشی هستند. نمودار شکل (۸) نشان می‌دهد که سرعت اجرای برنامه با GPU-1 نزدیک به ۲ تا ۲/۵ برابر بیشتر از سرعت اجرا با GPU-2 است و این موضوع، تأثیر تعداد هسته‌های یک GPU بر روی سرعت انجام محاسبات به‌صورت موازی را نشان می‌دهد. زمان‌برترین بخش برنامه، مربوط به بخش محاسبه توان خروجی شکل‌دهنده‌پرتو است و باقی بخش‌ها در مقایسه با بخش آخر، زمان بسیار کمتری را به خود اختصاص داده‌اند. همچنین بیشترین نسبت سرعت اجرای کودا به متلب نیز در همین بخش است. علت این موضوع قابلیت موازی‌سازی زیاد در محاسبات مربوط به مرحله محاسبه توان خروجی شکل‌دهنده پرتو است. با وجود اختلاف بین زمان‌های اجرا در این دو مدل GPU، اما هردو سخت‌افزار توانستند به‌کمک روش پیاده‌سازی ارائه‌شده در این مقاله به‌صورت زمان‌حقیقی داده‌های سونار را جهت‌یابی کنند (زمان مورد نیاز برای پردازش هر پنجره داده با استفاده از هر دو GPU کمتر از ۱۸۲ میلی‌ثانیه شده است).

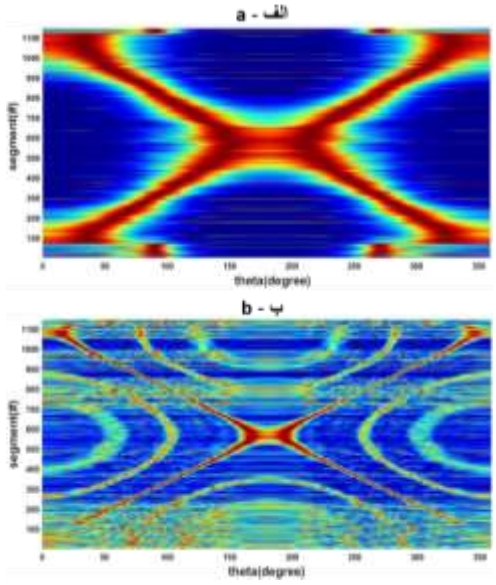


(شکل-۶): نمودار آبشاری نتیجه جهت‌یابی در سناریوی دوم

الف) جستجو در تمامی زوایا و بازه فرکانسی ۱۰۰-۳۰۰ هرتز

ب) جستجو در تمامی زوایا و بازه فرکانسی ۲۵۰۰-۳۵۰۰ هرتز

(Figure-6): Waterfall chart of the DOA result in second mode
a) Search in all angles and frequency range of 100-300 Hz
b) Search in all angles and frequency range of 2500-3500 Hz
(Search in all angles and frequency range of 950-1050 Hz)



(شکل-۷): نمودار آبشاری نتیجه جهت‌یابی در سناریوی سوم

الف) جستجو در تمامی زوایا و بازه فرکانسی ۱۰۰-۵۰۰ هرتز

ب) جستجو در تمامی زوایا و بازه فرکانسی ۲۵۰۰-۳۰۰۰ هرتز

(Figure-7): Waterfall chart of the DOA result in third mode
a) Search in all angles and frequency range of 100-500 Hz
b) Search in all angles and frequency range of 2500-3000 Hz

[1] Peak

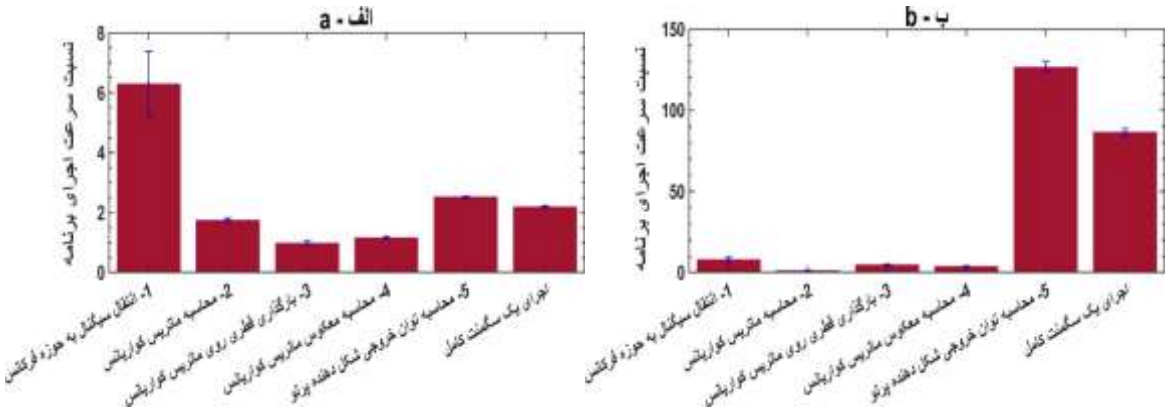
۶- نتیجه‌گیری

پیچیدگی محاسباتی زیاد الگوریتم کمینه‌واریانس مانع از اجرای زمان حقیقی جهت‌یابی صوت در آرایه‌های با تعداد حس‌گر زیاد می‌شود. این در حالی است که یکی از پارامترهای مهم هر سامانه سونار قابلیت جهت‌یابی زمان حقیقی است. برای افزایش سرعت اجرای الگوریتم کمینه واریانس می‌توان از پیاده‌سازی سخت‌افزاری (استفاده از FPGA و یا DSP) و یا برنامه‌نویسی موازی استفاده کرد. اگرچه پیاده‌سازی سخت‌افزاری سرعت بالاتری نسبت به برنامه‌نویسی موازی دارد اما منجر به کاهش انعطاف‌پذیری برنامه خواهد شد؛ بنابراین در این مقاله از رویکرد دوم (برنامه‌نویسی موازی) به‌منظور پیاده‌سازی الگوریتم کمینه‌واریانس استفاده شده است. زمان‌برترین عملیات پردازشی موجود در روش کمینه‌واریانس محاسبه توان خروجی شکل‌دهنده پرتو با استفاده از بردار وزن بهینه است؛ زیرا این عملیات به ازای هر زاویه و هر مؤلفه‌فرکانسی یک‌بار تکرار می‌شود. نتایج ارائه‌شده در این مقاله نشان می‌دهد که با استفاده از کارت گرافیکی موجود و مدل برنامه‌نویسی کودا می‌توان سرعت اجرای الگوریتم را نسبت به برنامه‌نویسی سریال تا حدود هشتاد برابر افزایش داد (شکل-۸). همچنین تأثیر عواملی چون تعداد حس‌گرها در آرایه، پهنای باند فرکانسی و ویژگی‌های سخت‌افزاری، بر روی سرعت اجرای برنامه بررسی شد.

(جدول-۱): زمان لازم برای جهت‌یابی صوت در داده‌های ثبت‌شده در سناریوی سوم (بازه فرکانسی ۲۵۰۰-۳۰۰۰ هرتز) روی متلب در سیستم اول، کودا در سیستم اول و کودا در سیستم دوم (میلی‌ثانیه)

(Table-1): The program's run time in third mode (frequency range of 2500-3000 Hz) on MATLAB in CPU-1, CUDA in GPU-1 and CUDA in GPU-2 (ms)

مرحله پردازش شده	زمان اجرای متلب روی CPU-1 (ms)		زمان اجرای کودا روی GPU-1 (ms)		زمان اجرای کودا روی GPU-2 (ms)	
	۱۱۹۶ پنجره	یک پنجره	۱۱۹۶ پنجره	یک پنجره	۱۱۹۶ پنجره	یک پنجره
انتقال سیگنال به حوزه فرکانس	268.6±4.1	0.234±0.003	34.4±6.15	0.030±0.005	212.2±26.69	0.185±0.023
محاسبه ماتریس کواریانس	364.8±9.8	0.318±0.008	328.4±14.45	0.287±0.013	576±14.70	0.503±0.013
بارگذاری قطری روی ماتریس کواریانس	698.5±18.8	0.609±0.016	145.3±9.49	0.127±0.008	141.9±6.88	0.354±0.006
محاسبه معکوس ماتریس کواریانس	1338.3±46.9	1.168±0.041	350.9±15.80	0.306±0.014	405.7±16.88	0.354±0.015
محاسبه توان خروجی شکل‌دهنده پرتو	225529.7±4165.1	196.797±3.634	1779.8±21.76	1.553±0.019	4477.1±20.78	3.907±0.018
مجموع مراحل	228199.9±4230.3	199.127±3.691	2638.9±28.38	2.303±0.025	5848.3±121.9	5.103±0.106



(شکل-۸): نمایش نسبت سرعت اجرای برنامه طبق اطلاعات جدول (۱)

الف) نسبت سرعت اجرای کودا روی سیستم نخست به سرعت اجرای کودا روی سیستم دوم

ب) نسبت سرعت اجرای کودا روی سیستم نخست به سرعت اجرای متلب روی سیستم نخست

(Figure-8): Displaying the ratio of the program execution speed according to information of table (1)
a) The ratio of the execution speed in CUDA on GPU-1 to that of GPU-2
b) The ratio of the execution speed in CUDA on GPU-1 to execution speed in MATLAB on CPU-1

۷- مراجع 7- References

[۱] یزدانی سیدحمید، ابوطالبی حمیدرضا. شکل‌دهی وفقی و هوشمند پرتو در آرایه‌های میکروفونی Ad-hoc با استفاده از خوشه‌بندی و رتبه‌بندی

میکروفون‌ها. پردازش علائم و داده‌ها. ۱۳۹۴; ۱۲ (۳) :۶۸-۵۷

[1] S. H. Yazdani, H. R. Abutalebi, "Adaptive and Smart Beamforming in Ad-hoc Microphone Arrays by Clustering and Ranking of the Microphones," *JSDP*. 2015; 12 (3) :57-68.

[۲] مجیدیان سینا، حدادی فرزان. تخمین جهت منابع با استفاده از زیرفضای کرونکر. پردازش علائم و داده‌ها. ۱۳۹۷; ۱۵ (۱) :۴۰-۲۹

[2] S. Majidian, F. Haddadi, "Direction of Arrival (DOA) Estimation Using Kronecker Subspace," *JSDP*. 2018; 15 (1) :29-40.

[3] J. Benesty, J. Chen, and Y. Huang, "Conventional Beamforming Techniques," in *Microphone Array Signal Processing*, vol. 1, Berlin, Germany: Springer, 2008.

[4] F. Vignon, and M. R. Burcher, "Capon beamforming in medical ultrasound imaging with focused beams," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 55, no. 3, pp. 619–628, 2008.

[5] F. Yan, M. Jin, and X. Qiao, "Low-Complexity DOA Estimation Based on Compressed MUSIC and Its Performance Analysis," *IEEE Transactions on Signal Processing*, vol. 61, no. 8, pp. 1915–1930, 2013.

[6] X. Wu, W. Zhu, and J. Yan, "Direction of Arrival Estimation for Off-Grid Signals Based on Sparse Bayesian Learning," IEEE Sensors Journal, vol. 16, no. 7, 2016.

[7] J. Dai, and H. Cheung, "Sparse Bayesian Learning Approach for Outlier-Resistant Direction-of-Arrival Estimation," *IEEE Transactions on Signal Processing*, vol. 66, no. 3, pp. 744–756, 2018.

[8] J. Capon, "High-resolution frequency-wavenumber spectrum analysis," *Proceedings of the IEEE*, vol. 57, no. 8, August 1969.

[9] J. I. Buskenes, J. P. Asen, C. I. C. Nilsen, and A. Austeng, "An Optimized GPU Implementation of the MVDR Beamformer for Active Sonar Imaging," *IEEE J. Ocean. Eng.*, vol. 40, no. 2, pp. 441–451, 2014.

[10] M. Sasso, and C. Cohen-Bacrie, "Medical Ultrasound Imaging Using the Fully Adaptive Beamformer," *Philips Research*, no. 4, pp. 489–492, 2005.

[11] H. Sun, M. Lu, and T. D. Abhayapala, "Ultrasound imaging using modal domain frequency smoothed MVDR beamforming," in *2017, 11th Int. Conf. Signal Process. Commun. Syst. ICSPCS 2017 - Proc.*, no. 1, pp. 1–5, Janua 2017.

[12] I. K. Holfort, F. Gran, and J. A. Jøensen, "Broadband minimum variance beamforming for ultrasound imaging," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 56, no. 2. pp. 314–325, 2009.

[13] J. P. Åsen, J. I. Buskenes, C. I. C. Nilsen, A. Austeng, and S. Holm, "Implementing capon beamforming on a GPU for real-time cardiac ultrasound imaging," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 61, no. 1, pp. 76–85, 2014.

[14] Y. S. Yoon, M. G. Amin, and F. Ahmad, "MVDR beamforming for through-the-wall radar imaging," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 47, no. 1, pp. 347–366, 2011.

[15] C. I. Nilsen and I. Hafizovic, "Beamspace adaptive beamforming for ultrasound imaging," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 56, no. 10, pp. 2187–2197, 2009.

[16] A. M. Deylami, and B. M. Asl, "A Fast and Robust Beamspace Adaptive Beamformer for Medical Ultrasound Imaging," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, 2017.

[17] J. F. Synnevåg, S. Holm, and A. Austeng, "A Low Complexity Data-dependent Beamformer," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 57, no. 2, 2010.

[18] K. Kim, S. Park, J. Kim, S. Park, and M. Bae, "A Fast Minimum Variance Beamforming Method Using Principal Component Analysis," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 61, no. 6, pp. 930–945, 2014.

[19] A. Jakobsson, S. L. Marple, Jr., and P. Stoica, "Computationally Efficient Two-Dimensional Capon Spectrum Analysis," *IEEE Trans. Signal Process.*, vol. 48, no. 9, pp. 2651–2661, 2000.

[20] Q. Huang, L. Zhang, and Y. Fang, "Performance analysis of Low-complexity MVDR beamformer in spherical harmonics domain," *Signal Processing*, vol. 153, pp. 153–163, 2018.

[21] B. M. Asl and A. Mahloojifar, "A low-complexity adaptive beamformer for ultrasound imaging using structured covariance matrix," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 59, no. 4, pp. 660–667, 2012.

[۲۲] صادقی حامد، اخوان بی‌تقصیر امیر. آشکارسازی سیگنال بر اساس پردازش موازی مبتنی بر جی‌پی‌یو در شبکه‌های حس‌گری صوتی دارای زیرساخت. پردازش علائم و داده‌ها. ۱۳۹۶; ۱۴ (۴) :۳۰-۱۹

[22] H. Sadeghi, A. Akhavan Bitaghsir, "Signal Detection Based on GPU-Assisted Parallel Processing for Infrastructure-based Acoustical Sensor Networks," *JSDP*. 2018; 14 (4) :19-30.

[23] D. B. Kirk, and W. W. Hwu, *Programming Massively Parallel Processors _ A Hands-On Approach*. Burlington, MA: Elsevier, 2010.

[24] R. Paridar, M. Mozaffarzadeh, M. Nasiriavanaki, and M. Orooji, "Double Minimum Variance Beamforming Method to Enhance Photoacoustic Imaging," *Journal of the Optical Society of America A*, pp. 1–9, 2018.

[25] A. Hakam, R. M. Shubair, and E. Salahat, "Enhanced DOA Estimation Algorithms Using MVDR and MUSIC," in *2013 International Conference on Current Trends in Information Technology (CTIT)*, Dubai, United Arab Emirates, 2013.

[۲۶] فتحی. یاسر، محلوجی‌فر. علی، محمدزاده اصل. بابک، "پیاده‌سازی بلادرنگ شکل‌دهی‌پرتو وفقی در تصویربرداری فراصدای پزشکی با استفاده از پردازش موازی به کمک واحدهای پردازنده ترسیمی (جی پی یو)"، *مجله انجمن مهندسی صوتیات ایران*، جلد ۱، شماره ۱، ۱۳۹۲.

[26] Y. Fathi, A. Mahloojifar, and B. M. Asl, "Real-time implementation of adaptive beamformer in ultrasound imaging using parallel processing with GPU," *Journal of Iran Audio Engineering Society*, vol. 1, no. 1, 2013.

[27] Nvidia, "cuBLAS," *docs.nvidia.com*, Oct. 30, 2018. [Online]. Available:https://docs.nvidia.com/cuda/cublas/index.html. [Accessed: Sept. 19, 2018].

احسان ایمانی‌فر مدرک کارشناسی خود را در سال ۱۳۹۷ از دانشگاه صنعتی همدان در رشته مهندسی پزشکی، گرایش بیوالکتریک دریافت کرد؛ سپس دوره کارشناسی ارشد را در دانشگاه صنعتی امیرکبیر در رشته مهندسی پزشکی- بیوالکتریک از سال ۱۳۹۷ تا ۱۴۰۰ گذراند. علایق پژوهشی وی برنامه‌نویسی موازی، پردازش سیگنال، هوش مصنوعی و پردازش تصویر هستند.

نشانی رایانامه ایشان عبارت است از:

ehsanimanif@gmail.com

امیر اخوان مدرک کارشناسی خود را در سال ۱۳۸۷ در رشته برق - مخابرات از دانشگاه صنعتی اصفهان و سپس دوره کارشناسی ارشد را در دانشگاه تربیت مدرس در رشته مهندسی پزشکی- بیوالکتریک از سال ۱۳۸۸ تا ۱۳۹۰ گذراند. وی مدرک دکترای خود را از دانشگاه صنعتی امیرکبیر در سال ۱۳۹۶ دریافت کرد. ایشان هم‌اکنون استادیار دانشکده برق و کامپیوتر دانشگاه صنعتی اصفهان است. زمینه‌های پژوهشی مورد علاقه ایشان عبارتند از: پردازش چندکاناله سیگنال‌های صوتی و سیگنال‌های حیاتی، بهینه‌سازی، مسأله‌های معکوس و جهت‌یابی صوت.

نشانی رایانامه ایشان عبارت است از:

aakhavan@cc.iut.ac.ir

علی‌اصغر آبنیکی مقطع کارشناسی و کارشناسی ارشد خود را در رشته مهندسی برق گرایش الکترونیک در دانشگاه جامع امام حسین(ع) گذراند و در سال ۱۳۸۸ موفق به دریافت مدرک کارشناسی ارشد شد. ایشان هم اکنون دانشجوی سال آخر دکترا در رشته مهندسی دریا گرایش آکوستیک زیرآب در دانشگاه صنعتی شریف و در مرحله دفاع از رساله دکترا خود هستند. ایشان هم‌اکنون به‌عنوان پژوهش‌گر در یک مرکز پژوهشی در حوزه آکوستیک زیر آب مشغول هستند. زمینه‌های پژوهشی مورد علاقه ایشان عبارتند از: پردازش سیگنال صوتی زیرآب، الگوریتم‌های دسته‌بندی و استخراج ویژگی در سیگنال‌های صوتی زیرآبی و انتشار صوت زیرآب.

نشانی رایانامه ایشان عبارت است از:

aliasghar_abniki@yahoo.com