

آشکارسازی سیگنال بر اساس پردازش موازی

مبتنی بر جی‌پی‌یو در شبکه‌های حس‌گری صوتی

دارای زیرساخت

حامد صادقی^{۱*} و امیر اخوان^۲

^۱ آزمایشگاه مخابرات بی‌سیم، دانشکده برق و کامپیوتر، دانشگاه تربیت مدرس، تهران، ایران

^۲ گروه مهندسی بیوشکی، دانشگاه صنعتی همدان، همدان، ایران

چکیده

الگوریتم فیشر، یکی از معروف‌ترین و پرکاربردترین روش‌های آشکارسازی آرایه‌ای سیگنال‌های صوتی بسامد پایین در شبکه‌های حس‌گری دارای زیرساخت است؛ اما یکی از مشکلات عمدۀ در بهکارگیری این الگوریتم، زمان طولانی انجام پردازش در آن است که در عمل، پیاده‌سازی بلادرنگ آشکارساز را با مشکل مواجه می‌سازد. در این مقاله، چگونگی پیاده‌سازی الگوریتم فیشر را با استفاده از واحد پردازش گرافیک (جی‌پی‌یو) به منظور تحقق محاسبات سریع و انجام پردازش‌های نزدیک به زمان واقعی، ارائه می‌کنیم. به خصوص به منظور بهبود هرچه بیشتر سرعت محاسبات، الگوریتم آشکارسازی با استفاده از روش پردازش موازی (مبتنی بر جی‌پی‌یو) پیاده‌سازی شده است. نتایج شبیه‌سازی‌ها، ارتقای قابل ملاحظه سرعت آشکارساز فیشر را نشان می‌دهند که باعث بهبود کارآیی شبکه حس‌گری صوتی خواهد شد.

واژگان کلیدی: شبکه حس‌گری، پردازش آرایه‌ای، شکل دهنده پرتو، پردازش موازی، جی‌پی‌یو.

Signal Detection Based on GPU-Assisted Parallel Processing for Infrastructure-based Acoustical Sensor Networks

Hamed Sadeghi^{1*} & Amir Akhavan²

¹ Wireless Communication Laboratory, ECE Department, Tarbiat Modares University, Tehran, Iran

² Department of Biomedical Engineering, Hamedan University of Technology, Hamedan, Iran

Abstract

Nowadays, several infrastructure-based low-frequency acoustical sensor networks are employed in different applications to monitor the activity of diverse natural and man-made phenomena, such as avalanches, earthquakes, volcanic eruptions, severe storms, super-sonic aircraft flights, etc. Two signal detection methods are usually implemented in these networks for the purpose of event occurrence identification, which are the progressive multi-channel correlator (PMCC) and the so-called Fisher detector. But, the Fisher method is more important and applicable in low signal-to-noise (SNR) ratio conditions, which is of a special interest in acoustical monitoring networks. Unfortunately, an important disadvantage of this algorithm is its relative high detection-time; which limits its application for real-time detection scenarios. This disadvantage is fundamentally due to a beam forming process in Fisher algorithm, which requires doing complete search in a slowness-network, constructed from possible incoming wave front directions and speeds. To address this issue, we propose a method for implementation of this beam forming on a graphics processing unit (GPU), in order

* Corresponding author

* نویسنده عهده‌دار مکاتبات



to realize a fast-computing and/or near real-time signal processing technique. In addition, we also propose a parallel-processing algorithm for further enhancement of the performance of this GPU-based Fisher detector. Simulation results confirm the performance improvement of Fisher detector, in terms of required processing time for acoustical signal detection applications.

Keywords: Sensor network, array processing, beamforming, parallel processing, GPU.

این روش بهدلیل پایداری مناسب در مقابله نویفه و قابلیت پیاده‌سازی پهن‌باند به‌طور گستردۀ برای تعیین جهت و آشکارسازی سیگنال‌های صوتی و لرزه‌ای به کار می‌رود [5]-[8]. روش فیشر برای نخستین بار در سال ۱۹۵۷ توسط ملتمن و بایلی⁷ پیشنهاد شد [9]. پس از آن این الگوریتم به‌منظور آشکارسازی سیگنال‌های لرزه‌ای استفاده شد [10]. این آشکارساز که بر اساس آنالیز واریانس نسبت سیگنال به نویفه⁸ استخراج می‌شود، شامل دو بخش شکل‌دهی پرتو⁹ و آشکارسازی مبتنی بر نسبت فیشر (به‌منظور ارزیابی محتوای سیگنال پرتوها) است. نسبت فیشر در واقع نسبت واریانس سیگنال به واریانس نویفه است که در هر دو حوزه زمان و پس‌آمد قابل پیاده‌سازی است. یکی از ویژگی‌های برجسته این روش آشکارسازی، مقاومبودن آن در مقابل نویفه است. این خاصیت در پردازش داده‌های مربوط به زلزله و شرایطی که نسبت توان سیگنال به نویفه پایین است، اهمیت زیادی دارد. شایان ذکر است که آشکارساز فیشر در کاربردهای مختلفی همچون تشخیص فعالیت‌های آشفسانی و ردیابی مسیر شهاب‌های آسمانی نیز به کار می‌رود [11]-[12]. این روش در مقایسه با آشکارساز معروف و پرکاربرد همبستگی چند کاناله پیش رو¹⁰، موسوم به پی‌ام‌سی‌سی¹¹، توانایی بیشتری برای آشکارسازی سیگنال در نسبت سیگنال به نویفه‌های پایین دارد. در مقابل، الگوریتم فیشر به‌دلیل جستجوی صفحه گندی به زمان بیشتری برای پردازش داده نیاز دارد.

در روش آشکارساز فیشر تخمین پارامترهای موج دریافتی بر مبنای اختلاف زمانی بین ثبت‌های صورت‌گرفته در حس‌گرها انجام می‌شود. از آنجایی که این اختلاف زمانی، به چیدمان حس‌گرهای آرایه و جهت ورود سیگنال بستگی دارند، در این روش ثبت‌های حس‌گرها به‌ازای یک

⁷ Melton and Bailey

⁸ Signal-to-Noise Ratio (SNR)

⁹ Beamforming

¹⁰ Progressive Multi-Channel Correlation (PMCC)

۱- مقدمه

یکی از انواع شبکه‌های حس‌گری¹، شبکه‌های حس‌گری صوتی مبتنی بر زیرساخت² (IASN) است که بیشتر به‌منظور آشکارسازی و جهت‌یابی سیگنال‌های گسیل‌شده از منبع مولد صدا به کار می‌رود [1]-[4]. در IASN پس از دریافت سیگنال‌های ارسالی از حس‌گرها در مرکز هم‌جوشی³ (FC)، وقوع رخداد با استفاده از الگوریتم‌های پردازش آرایه‌ای جستجو می‌شود. این الگوریتم‌ها شامل دو مرحله اصلی آشکارسازی⁴ و سپس تخمین جهت و سرعت ورود موج است. اصول پردازش سیگنال در این الگوریتم‌ها بر مبنای استفاده از مفهوم همدوسی⁵ امواج ساطع شده از منبع صوت و نیز همبستگی متقابل بین سیگنال‌های خروجی حس‌گرها است. به عبارت دیگر، از آنجایی که در صورت حضور سیگنال مورد نظر در موج دیس‌های ثبت‌شده، همبستگی بین موج دیس‌های خروجی حس‌گرها افزایش می‌یابد، لذا پارامتر همبستگی به عنوان یک معیار مناسب برای جدا کردن نویفه از سیگنال هدف، می‌تواند استفاده شود. از طرفی به‌منظور تخمین اختلاف زمان ورود سیگنال به حس‌گرهای متفاوت از همبستگی متقابل بین سیگنال‌های حس‌گرها مختلف می‌توان استفاده کرد. روشن است که مقادیر اختلاف زمانی، به هندسه و فاصله نسبی حس‌گرها از یکدیگر و نیز پارامترهای موج صوتی وابسته است. پس از تخمین مقادیر اختلاف زمانی، پارامترهای انتشار موج از جمله راستای ورود موج و سرعت آن محاسبه می‌شود.

یکی از روش‌های پرکاربرد آشکارسازی سیگنال و تخمین پارامترهای موج در IASN، الگوریتم فیشر⁶ است.

¹ Sensor Networks (SNs)

² Infrastructure-based Acoustic Sensor Network (IASN)

³ Fusion Center (FC)

⁴ Detection

⁵ Coherency

⁶ Fisher

بسیار حائز اهمیت است. به عنوان جمع‌بندی، این روش یک الگوریتم تلفیقی شکل‌دهی پرتوی سریع مبتنی بر آمارگان فیشر خواهد بود که به طور همزمان، هم حضور سیگنال هدف را آشکارسازی کرده و هم آن را می‌تواند جهت‌یابی کند. ساختار مقاله بدین شرح است: در بخش ۲، مفاهیم اولیه پردازش موازی داده، ساختار پردازنده گرافیکی و شیوه برنامه‌نویسی آن شرح داده شده است. همچنین در این بخش مروری مختصر بر آشکارسازی سیگنال‌های بسامد-پایین با استفاده از نسبت فیشر انجام می‌شود. در بخش ۳ روش پیشنهادی آشکارسازی سریع منابع صوتی بسامد پایین و چگونگی پیاده‌سازی الگوریتم آن ارائه شده است. نتایج شبیه‌سازی‌ها و کارآیی روش پیشنهادی در بخش ۴ بررسی می‌شوند. در انتها نتیجه‌گیری کلی مقاله و مراجع در بخش ۵ و ۶ ارائه شده است.

۲- آشکارسازی سریع سیگنال‌های صوتی

در همین‌اواخر استفاده از واحد پردازنده گرافیکی (جی‌پی‌و^۷) مورد توجه پژوهش‌گران حوزه محاسبات و پردازش سریع قرار گرفته است. مقالات منتشرشده در این زمینه حاکی از آن است که به کارگیری این پردازنده‌ها به‌منظور تسريع الگوریتم‌های دارای قابلیت موازی‌سازی، موفق بوده است [۱۴]-[۱۹]. به طور خاص، در حال حاضر یکی از زمینه‌های پژوهشی فعال در حوزه‌هایی که نیازمند پردازش سریع هستند و قابلیت موازی شدن در الگوریتم‌های مورد استفاده آن‌ها وجود دارد، به کارگیری پردازنده‌های گرافیکی با معماری کودا^۸ است [۲۰،۲۱]. در این مقاله پیشنهاد می‌شود، از این پردازنده‌ها به‌منظور پردازش سریع سیگنال‌های صوتی دریافتی از IASN استفاده شود. به عبارت دیگر در این مقاله، الگوریتم فیشر با استفاده از پردازش موازی مبتنی بر جی‌پی‌و پیاده‌سازی شده است. نتایج مقاله نشان می‌دهد که با این روی کرد، نقطه ضعف آشکارساز فیشر، که زمان برآوردن جستجو در شبکه گندی است، برطرف شده است.

پیش از پرداختن به الگوریتم پیشنهادی، ابتدا برخی مفاهیم پایه‌ای معرفی می‌شوند.

⁷ Graphics Processing Unit (GPU)

⁸ Compute Unified Device Architecture (CUDA)

جهت و سرعت فرضی برای ورود سیگنال، هم‌تراز می‌شوند؛ سپس همدوسی آن‌ها با استفاده از نسبت فیشر محاسبه می‌شود. این کار به‌طور مداوم به‌ازای جهت‌ها و سرعت‌های مختلف ورود سیگنال صدا صورت می‌پذیرد تا درنهایت، جهت و سرعتی که بیشینه مقدار همدوسی بین ثبت‌ها را ایجاد کند، به عنوان جهت و سرعت موج انتخاب شود. یک روش جستجوی سامان‌مند^۱ برای تخمین جهت و سرعت موج، استفاده از روش شکل‌دهی پرتو است که در آن از یک شبکه گندی^۲ استفاده می‌شود. اندازه هر بردار در صفحه گندی، معکوس سرعت ظاهری^۳ است و زاویه آن با شمال زاویه سمت‌پشتی^۴ نامیده می‌شود. جستجوی بردار گندی در تمامی جهت‌ها، تضمین می‌کند که یکی از پرتوها جهت تقریبی ورود سیگنال به آرایه را نشان دهد. واضح است که بسته به تعداد نمونه‌ها در صفحه گندی، میزان تفکیک‌پذیری^۵ زاویه و سرعت تخمینی موج در این شبکه تحت تاثیر قرار خواهد گرفت. از سوی دیگر با افزایش تعداد نمونه‌ها، زمان بیشتری نیز برای محاسبات مورد نیاز است و این مانع از پیاده‌سازی بلادرنگ آشکارسازی سیگنال‌های صوتی می‌شود.

از آنجایی که مشکل اصلی روش فیشر، انجام جستجوی سنگین^۶ در فضای شبکه گندی و درنتیجه، غیر زمان واقعی‌بودن سامانه است، در این مقاله یک روش سریع‌سازی الگوریتم فیشر پیشنهاد می‌شود که دارای کاربردهای گسترده‌ای در سامانه‌های نظارتی خواهد بود. روش کار بدین صورت است که ابتدا برخی محاسبات اولیه و نیز پنجره‌گذاری بر روی داده‌های ورودی در سی‌پی‌و انجام شده و سپس برای هر پنجره، نسبت‌های آمارگان فیشر به‌ازای هر کدام از عناصر شبکه گندی، به‌طور موازی توسط جی‌پی‌و محاسبه شوند. درنهایت، بردار گندی متناظر با بیشینه نسبت فیشر، انتخاب شده و با استفاده از آن پارامترهای موج برای آن پنجره زمانی تخمین زده می‌شوند. در این مقاله نشان داده خواهد شد که روش پیشنهادی باعث کاهش زمان آشکارسازی منبع شد، که این امر برای سامانه‌های نظارتی

¹ Systematic

² Slowness

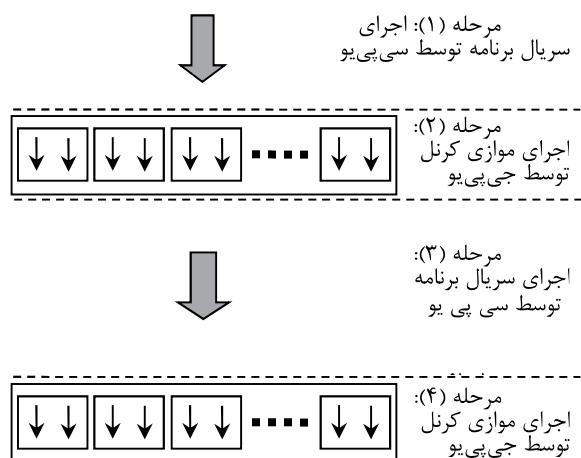
³ Apparent velocity

⁴ Back azimuth (BAZ)

⁵ Resolution

⁶ Exhaustive search

کاربردها، ابداع شد. با استفاده از این معماری، جی‌پی‌یو می‌تواند علاوه بر محاسبات گرافیکی، محاسبات همه‌منظوره را نیز انجام دهد. شرکت ان‌ویدیا علاوه بر ارائه سختافزاری که بتواند محاسبات گرافیکی و همه‌منظوره را انجام دهد، زبان برنامه‌نویسی جدیدی به نام کودا^۵ را نیز ارائه کرد. این نوع برنامه‌نویسی که در قالب محاسباتی غیرهمگن^۶ انجام می‌شود، امکان بهره‌گیری همزمان از مزایای سی‌پی‌یو و جی‌پی‌یو را فراهم می‌سازد. در این روش برنامه‌نویسی، بخش سری کد توسط سی‌پی‌یو اجرا شده و هسته موازی برنامه، موسوم به کرنل، توسط جی‌پی‌یو پردازش می‌شود. مدل اجرایی این روش برنامه‌نویسی در (شکل-۱) نمایش داده شده است. با توجه به این شکل، سی‌پی‌یو کدهای مربوط به مراحل ۱ و ۳ را به صورت سریال و جی‌پی‌یو کدهای مربوط به مراحل ۲ و ۴ را به صورت موازی اجرا می‌کند. هنگامی که یک تابع کرنل به صورت موازی راهاندازی^۷ شود (مرحله ۲ و ۴)، تعداد زیادی thread بر روی جی‌پی‌یو اجرا می‌شود.



(شکل-۱): مدل برنامه‌نویسی کودا-سی.
(Figure-1): CUDA-C programming model.

مجموعه تمام threadهایی که توسط یک تابع کرنل ایجاد می‌شود، گرید^۸ نامیده می‌شود. هنگامی که اجرای تمام threadهای یک کرنل پایان یابد، اجرای برنامه بر روی میزبان (سی‌پی‌یو) آغاز شده و تا زمان اجرای کرنل بعدی ادامه می‌یابد.

⁵ CUDA C

⁶ Heterogeneous computing

⁷ Launch

⁸ Grid

۲-۱- موازی‌سازی داده‌ها

در بسیاری از کاربردها حجم بالای داده، زمان پردازش نرم‌افزارهای موجود را در رایانه‌های سری بسیار طولانی می‌کند. هم‌چنین بسیاری از این نرم‌افزارها پدیده‌های فیزیکی موجود در دنیای واقعی را مدل می‌کنند. به عنوان نمونه، تصاویر و فریم‌های ویدئویی، نمایش لحظه‌ای دنیای واقعی هستند که در آن بخش‌های مختلف تصویر، پدیده‌های فیزیکی مستقل را به طور همزمان نشان می‌دهند. چنین استقلالی، پایه‌ی موازی‌سازی داده‌ها است. یک مثال ساده برای درک موازی‌سازی داده‌ها، ضرب دو ماتریس بسیار بزرگ در یکدیگر است که در آن هر درایه ماتریس حاصل ضرب، به صورت مستقل از سایر درایه‌ها قابل محاسبه است. به عبارت دیگر عملیات ریاضی لازم را برای محاسبه تمام درایه‌های ماتریس خروجی به صورت موازی می‌توان انجام داد؛ بنابراین ضرب دو ماتریس با ابعاد بالا قابلیت زیادی در موازی‌سازی داده دارد. موازی‌سازی داده در کاربردهای واقعی می‌تواند بسیار پیچیده‌تر باشد.

۲-۲- واحد پردازنده گرافیکی با معماری کودا

جی‌پی‌یو یک پردازنده موازی است که برای کاربردهای گرافیکی ابداع شده است؛ اما در سال‌های اخیر به عنوان یک پردازنده عمومی نیز به کار گرفته می‌شود. تفاوت عمدی و ساختاری بین جی‌پی‌یو و سی‌پی‌یو^۹ در این است که معماری سختافزاری جی‌پی‌یو تعداد بیشتری آلیو^{۱۰} را دربردارد و در مقابل، اجزای کمتری برای کش^{۱۱} و واحد کنترل دارد. این امر باعث می‌شود تا جی‌پی‌یو قابلیت محاسباتی بالایی داشته باشد. به طور خاص، جی‌پی‌یو توانمندی بیشتری برای اجرای موازی محاسبات ریاضی، که در آن تعداد زیادی عملیات مشابه، روی تعداد زیادی داده انجام می‌شود، دارد

[22]

همان‌طور که بیان شد، نسل‌های اولیه جی‌پی‌یو تنها قادر به اجرای محاسبات گرافیکی بودند؛ اما از اوخر سال ۲۰۰۷، معماری کودا توسط شرکت ان‌ویدیا^{۱۲} به منظور بهره‌گیری از قابلیت‌های محاسباتی جی‌پی‌یو در سایر

¹ Central Processing Unit (CPU)

² Arithmetic Logic Unit (ALU)

³ Cache

⁴ NVIDIA

$$x_{nt} = \bar{x} + \alpha_t + \epsilon_{nt} \quad (4)$$

با توجه به آنچه در بالا ذکر شد، تغییرات کل ثبت‌ها عبارت است از:

$$V_T = \sum_{t=1}^T \sum_{n=1}^N (x_{nt} - \bar{x})^2 \quad (5)$$

$$V_T = V_W + V_B, \quad (6)$$

$$V_W = \sum_{t=1}^T \sum_{n=1}^N (x_{nt} - \bar{x}_t)^2 \quad (7)$$

$$V_B = N \sum_{t=1}^T (\bar{x}_t - \bar{x})^2 \quad (8)$$

که در آن V_B معیاری از توان سیگنال هدف و V_W معیاری از توان نویه در سیگنال ثبت شده است. با توجه به این که آشکارساز فیشر یک آشکارساز آماری است، توجه به توزیع آماری متغیرهای V_B و V_W بسیار مهم است. می‌توان نشان داد که امید ریاضی این متغیرها به صورت زیر است:

$$E(V_W) = T(N-1)\sigma^2 \quad (9)$$

$$E(V_B) = (T-1)\sigma^2 + N \sum_{t=1}^T \alpha_t^2 \quad (10)$$

که در آن σ^2 واریانس نویه در هنگام ثبت سیگنال‌های حس‌گرهای مختلف است. با توجه به روابط بالا، $\frac{V_W}{T(N-1)}$ تخمینی بدون بایاس از واریانس از واریانس α_t است. در مقابل عبارت $\frac{V_B}{T-1}$ تنها در شرایط H_0 و H_1 معتبر است. در شرایط H_0 که α_t صفر است، تخمینی بدون بایاس از واریانس از α_t است. با استفاده از روابطه بالا نسبت فیشر به صورت تقسیم دو واریانس تعریف می‌شود:

$$F = \frac{V_B/(T-1)}{V_W/T(N-1)} \quad (11)$$

روشن است در صورت صحیح بودن فرض H_0 مقدار خروجی آشکارساز، به طور تقریبی برابر یک و در غیر این صورت بیشتر از یک خواهد بود. با ساده‌سازی رابطه بالا آزمون فرض به صورت زیر در می‌آید:

$$F = \frac{T(N-1) \sum_{t=1}^T (\sum_{n=1}^N x_{nt})^2 - \frac{1}{T} (\sum_{t=1}^T \sum_{n=1}^N x_{nt})^2}{N(T-1) \frac{\sum_{t=1}^T \sum_{n=1}^N x_{nt}^2 - \frac{1}{T} \sum_{t=1}^T (\sum_{n=1}^N x_{nt})^2}{\sum_{t=1}^T (\sum_{n=1}^N x_{nt})^2}} \quad (12)$$

با توجه به نحوه تعریف V_B و V_W می‌توان نشان داد، توزیع احتمال آشکارساز فیشر در شرایط H_0 ، یک توزیع اف مرکزی با پارامترهای ۱ و $v_1 = T(N-1)$ و $v_2 = T$ است.

۲-۳- آماره آشکارساز فیشر

در پردازش بلادرنگ سیگنال‌های صوتی، نه تنها آشکارسازی باید به اندازه کافی سریع باشد، بلکه باید روی داد به صورت خودکار و بدون نیاز به مفسر تشخیص داده شود. یک راه برای انجام این کار استفاده از آشکارساز آماری فیشر است که به طور گستردگی در این حوزه استفاده می‌شود. آشکارساز فیشر از همدوسي بین سیگنال‌های دریافتی توسط حس‌گرهای مختلف در یک آرایه صوتی بهره می‌گیرد. در ادامه به معرفی مختصر این آشکارساز می‌پردازیم. فرض کنیم که یک آرایه مشکل از N حس‌گر، تغییرات فشار جوی را در بازه زمانی $T = 1, \dots, T$ می‌کند، سیگنال‌های ثبت شده توسط حس‌گرهای در یک ماتریس ذخیره می‌شوند، به گونه‌ای که هر سطر شامل سیگنال یک حس‌گر بوده و هر ستون، نمونه‌های زمانی سیگنال‌های حس‌گرها را در یک زمان مشخص نشان می‌دهد. در این صورت یک ماتریس $N \times T$ به دست می‌آید. ثبت‌های صورت گرفته در محور زمان پنجره‌گذاری می‌شوند و پردازش آرایه‌ای بر روی پنجره‌ها، به طور جداگانه صورت می‌گیرد؛ این بازه‌های زمانی بین^۱ نامیده می‌شوند. متوسط هر ستون از این ماتریس به صورت زیر نمایش داده می‌شود [۷]:

$$\bar{x}_t = \frac{1}{N} \sum_{n=1}^N x_{nt} \quad (1)$$

که در آن x_{nt} نمونه به صفحه متناظر با زمان t و حس‌گر n است. داده‌های ثبت شده (حاوی نویه) به صورت زیر مدل می‌شوند:

$$x_{nt} = \bar{x}_t + \epsilon_{nt} \quad (2)$$

که در آن ϵ_{nt} انحراف هر نمونه از مقدار متوسط است. نویه در این مدل به صورت یک متغیر گوسی با متوسط صفر و واریانس σ^2 در نظر گرفته می‌شود. متوسط کل نمونه‌ها عبارت است از:

$$\bar{x} = \frac{1}{NT} \sum_{t=1}^T \sum_{n=1}^N x_{nt} \quad (3)$$

از آن جایی که انحراف هر ستون از متوسط کل را به صورت $\bar{x}_t - \bar{x}$ می‌توان به دست آورد، لذا در این حالت مدل داده به صورت زیر خواهد بود:

^۱ Bin



دارد، انتظار می‌رود که با موازی‌سازی پردازش، زمان اجرای الگوریتم به طور قابل توجهی کاهش یابد. نسبت فیشر متناظر با هر بردار کندی، در یک ماتریس ذخیره می‌شود. پس از تکمیل این ماتریس توسط جی‌پی‌بو، نتایج حاصل دوباره به سی‌پی‌بو بازگردانده می‌شود. در ادامه، بیشینه این نسبت‌ها توسط سی‌پی‌بو محاسبه می‌شود و پس از آن که بیشینه نسبت فیشر به‌ازای تمامی پنجره‌ها محاسبه شد، به تعداد پنجره‌ها ضریب خواهیم داشت، $[F_1, F_2, \dots, F_N]$. بیشینه این مقادیر مربوط به پنجره‌های است که بیشترین توان سیگنال منبع صوت را در خود جای داده است و بیشترین همبستگی را بین داده‌های ثبت شده توسط حس‌گرهای آرایه دارد. بردار کندی متناظر با این نقطه، سرعت ظاهری و راستای منبع صوت را تخمین می‌زند.

مهم‌ترین پارامترهای مورد استفاده در پیاده‌سازی الگوریتم فیشر عبارتند از: آستانه نسبت فیشر، تفکیک‌پذیری نمونه‌برداری در شبکه بردار کندی، طول پنجره‌های زمانی، مقدار هم‌پوشانی پنجره‌ها و نرخ نمونه‌برداری. مقدار آستانه (ترشولد) آشکارساز فیشر توسط احتمال هشدار غلط^۱ قابل تحمل در سامانه محاسبه می‌شود. تعیین آستانه در گام‌های اولیه نیاز به نظرات محققین خبره در این حوزه دارد تا با استفاده از آن و توزیع F^2 آشکارساز فیشر، آستانه مناسب مشخص شود. تفکیک‌پذیری نمونه‌برداری از شبکه کندی به طور مستقیم بر تفکیک‌پذیری جستجوی سرعت و راستای موج تخمینی اثر می‌گذارد؛ از طرفی هرچه شبکه کندی با تفکیک‌پذیری بالاتری نمونه‌برداری شود، بار محاسباتی آشکارسازی افزایش می‌یابد. انتخاب طول پنجره‌های زمانی مناسب، نقش مؤثری در خروجی آشکارساز فیشر در هر پنجره دارد. اگر پنجره زمانی بسیار کوچک در نظر گرفته شود، به‌طوری که مقدار آن کمتر از زمان سیر موج از حس‌گر مرجع تا حس‌گر n باشد، در این صورت بین این دو حس‌گر در پنجره موجود، همبستگی وجود نخواهد داشت و نمی‌توان از فزونی اطلاعات این حس‌گرهای استفاده کرد؛ بنابراین در نظر گرفتن پنجره‌های زمانی بسیار کوچک به تشخیص یک بردار کندی اشتباہ منجر خواهد شد. از سوی دیگر هرچه پنجره‌های زمانی بزرگ‌تر باشد، تغییرات مرحله در آن‌ها بیشتر است و همبستگی بین سیگنال حس‌گرهای کاهش

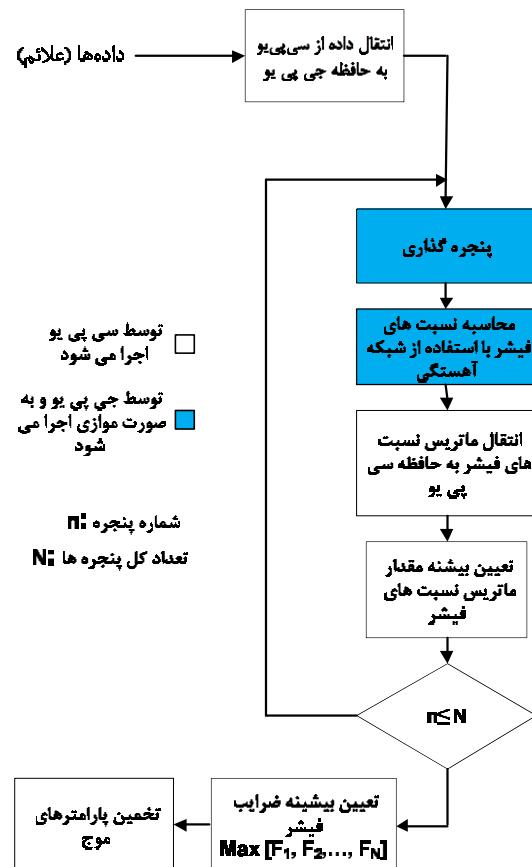
¹ False alarm (FA)

² F distribution

در شرایط H_1 ، یک توزیع اف غیرمرکزی با پارامتر غیرمرکزی $\lambda_{nc} = v_1 \cdot snr$ است.

۳- روش پیشنهادی پیاده‌سازی الگوریتم فیشر

برای درک بهتر روش پیشنهادی پیاده‌سازی الگوریتم فیشر، روند اجرا در (شکل-۲) نمایش داده شده است. همان‌طور که مشاهده می‌شود، الگوریتم با انتقال داده‌ها از سی‌پی‌بو به حافظه سراسری جی‌پی‌بو آغاز و پس از این انتقال مراحل بعدی در جی‌پی‌بو انجام می‌شود؛ در مرحله پنجره‌گذاری، مجموعه دادگان به پنجره‌های زمانی با طول معین تقسیم و در هر پنجره، نسبت فیشر به‌ازای تمامی بردارهای کندی محاسبه و پارامترهای سیگنال (سرعت ظاهری و زاویه سمت‌پشتی) برای بردار کندی متناظر با بزرگ‌ترین نسبت فیشر، تعیین می‌شوند.



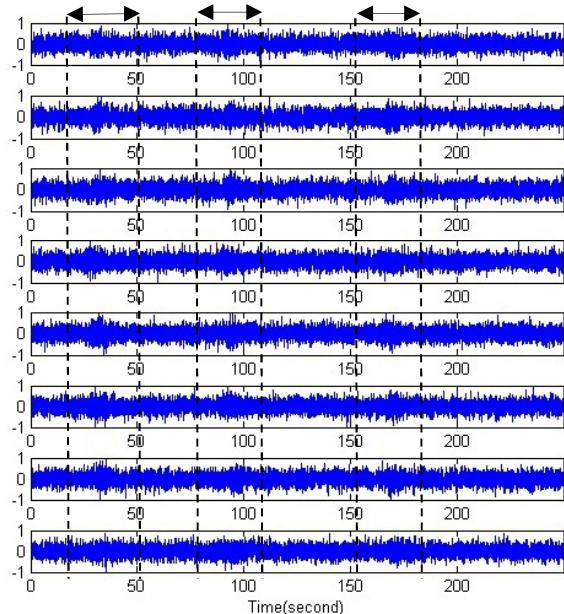
(شکل-۲): روندنا و مراحل پیاده‌سازی الگوریتم فیشر.

(Figure-2): Flowchart and implementation steps of Fisher algorithm.

لازم به توضیح است که در روش پیشنهادی، محاسبه نسبت فیشر به‌ازای بردارهای کندی مختلف به صورت موازی انجام می‌شود. بدلیل آن که تعداد زیادی بردار کندی وجود



می‌شود. این چینش بهدلیل وجود تقارن نسبی نسبت به راستاهای مختلف انتخاب شده است. پاسخ آرایه مربوط به این چینش بهازای بسامدهای ۱ و ۲ هرتز در شکل (۵) نمایش داده شده است. دو دایره رسم شده در این شکل محدوده پاسخ آرایه برای بسامدهای ۱ و ۲ هرتز را نشان می‌دهند. توجه به این نکته ضروری است که طراحی چیدمان آرایه نقش مهمی در عملکرد الگوریتم دارد. این چیدمان باید بر اساس بازه بسامدی پردازش و تفکیک پذیری زاویه‌ای مورد نیاز طراحی شود.



(شکل-۳): سیگنال شبیه‌سازی شده حس‌گرهای آرایه با سه مؤلفه از سه راستای ۹۰، ۶۰ و ۱۲۰ درجه. موقعیت زمانی حضور سیگنال منبع در هر حس‌گر بهصورت تقریبی با خطچین نمایش داده شده است.

(Figure-3): Simulated signals of array sensors from three different directions of 90, 60, and 120 degrees. Approximate time-domain locations of source signals are presented by dotted lines.

شبکه کُندی بهکاربرده شده در این پژوهش در شکل (۶) مشاهده می‌شود. هر برداری کندی نماینده یک سرعت ظاهری و جهت موج دریافتی است. فرض می‌شود که موج ورودی از تمام جهات و با سرعت‌های ظاهری مختلف می‌تواند وارد شود. برای مقادیر پیش‌فرض برداری کُندی، فرض شده است که نمونه‌های شبکه بهصورت یکنواخت انتخاب شده‌اند (یعنی فاصله بین هر دو سرعت متوازی، یکسان است). تعداد صد نمونه از هر سطر و ستون این شبکه دریافت شده و درمجموع دههزار بردار کندی را تشکیل می‌دهند. البته بهدلیل حصول وضوح بیشتر، تمام نمونه‌ها در شکل (۶) نمایش داده نشده‌اند. این عدد (۱۰۰×۱۰۰) برابر با

می‌یابد. در این حالت، بیشینه نسبت فیشر در پنجره‌ها کاهش می‌یابد. در بخش مربوط به شبیه‌سازی مقادیر پارامترهای تنظیم شده ارائه می‌شود.

(جدول-۱): پارامترهای سیگنال مرجع.

(Table-1): Reference signal parameters.

نام پارامتر	مقدار
بسامد مرکزی	۳ هرتز
بسامد نمونه‌برداری	۴۰ هرتز
فاکتور تضعیف	۰.۳
همپوشانی پنجره‌ها	۸۰٪
طول پنجره	۱۰۲۴ نمونه

۴- نتایج شبیه‌سازی‌ها و بحث

در این بخش پس از ارائه مدل سیگنال و چینش آرایه به بررسی نتایج پیاده‌سازی الگوریتم فیشر بر روی سی‌بی‌پو و جی‌پی‌يو می‌پردازیم. در انتهای مقایسه سرعت اجرای این الگوریتم در دو معماری سی‌بی‌پو و جی‌پی‌يو ارائه می‌شود.

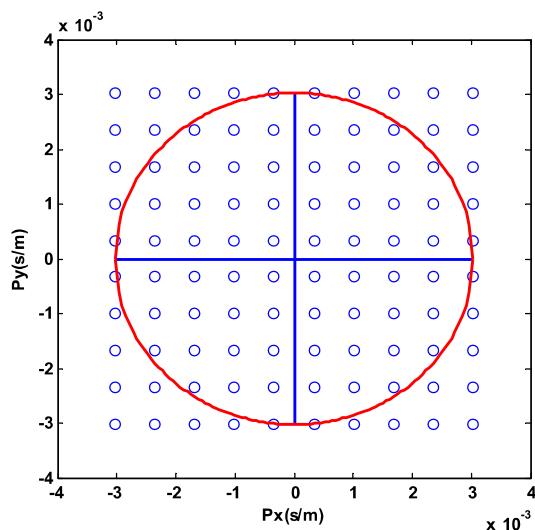
۴-۱- مدل سیگنال و چینش آرایه

مدل سیگنال استفاده شده در شبیه‌سازی‌های این بخش از رابطه زیر پیروی می‌کند:

$$s(t) = ate^{-at} \sin(2\pi ft) \quad (13)$$

که از سه بخش سینوسی، نمایی و خطی تشکیل شده است. a بسامد مرکزی سیگنال و α فاکتور تضعیف نام دارد. نرخ تضعیف سیگنال مرجع، توسط پارامتر a قابل تنظیم است. پارامترهای سیگنال مرجع شبیه‌سازی شده در (جدول-۱) نمایش داده شده است. بهمنظور شبیه‌سازی سیگنال خروجی هر حس‌گر، سه منبع صوت در راستاهای سیگنال ۹۰، ۶۰ و ۱۲۰ درجه بهترتیب با سرعت‌های انتشار موج ۳۶۰، ۳۴۰ و ۳۹۰ متر بر ثانیه در نظر گرفته شده است. لازم به ذکر است که منظور از سرعت انتشار، سرعت ظاهری انتشار موج نسبت به آرایه است. با توجه به زوایا و سرعت‌های انتخاب شده و همین‌طور موقعیت مکانی حس‌گرهای سیگنال خروجی هر یک از حس‌گرها بهازای نسبت سیگنال به نویه ۵-۵ دسی‌بل بهصورت شکل (۳) در می‌آید. موقعیت زمانی حضور سیگنال‌های منبع (بهازای سه راستای ذکر شده) بهطور تقریبی توسط خطچین نمایش داده شده است.

آرایه صوتی در نظر گرفته شده در این مقاله، متشکل از ۸ حس‌گر است. پیکربندی آرایه در شکل (۴) مشاهده



(شکل-۶): نمایش شبکه کنندی. نقاط داخل دایره سرعت‌های بیش تراز ۳۳۰ متر بر ثانیه را مشخص می‌کند.

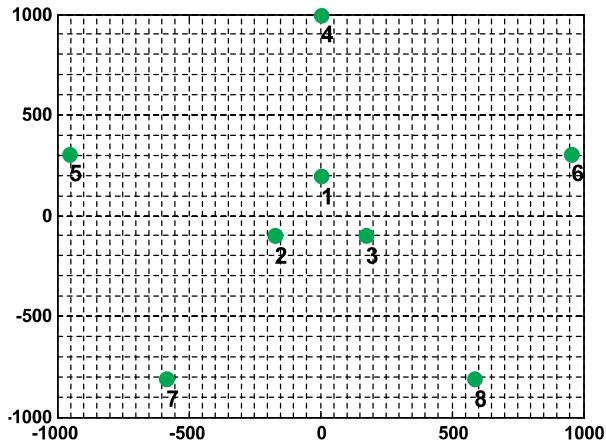
(Figure-6): Slowness-network illustration. The points inside the circle represent the source wave velocities higher than 330 m/s.

پردازنده مورد استفاده در این پژوهش، یک سی‌پی‌یو چهار هسته‌ای ۱۵ شرکت اینتل با بسامد ساعت ۳ گیگاهرتز است. از آنجایی که سرعت اجرای الگوریتم به قابلیت محاسباتی جی‌پی‌یو مورد استفاده نیز بستگی دارد، زمان اجرای الگوریتم با استفاده از دو پردازنده گرافیکی متفاوت یکی با مدل ۶۳۰ Geforce GT، قابلیت محاسباتی ۲/۱ و تعداد کل هسته ۹۶ (جی‌پی‌یو شماره ۱) و دیگری Geforce GTX 650 TI Boost با قابلیت محاسباتی ۳/۰ و تعداد کل هسته ۷۶۸ (جی‌پی‌یو شماره ۲) به دست آمده و مقایسه شده است. در شبیه‌سازی‌های این قسمت از نرم‌افزار Visual Studio 2010 و CUDA toolkit 5.0 استفاده شده است. در ادامه نتایج پیاده‌سازی آشکارساز فیشر بر روی سی‌پی‌یو و جی‌پی‌یو تشریح می‌شود.

۴-۲- نتایج تخمین پارامتر موج

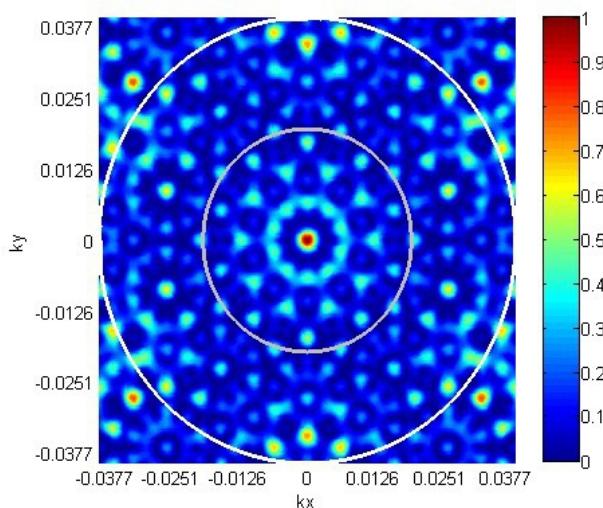
پس از شبیه‌سازی سیگنال‌های دریافتی در محل حسنگرهای الگوریتم یک بار بدون اعمال پردازش موازی و بار دیگر با اعمال پردازش موازی با استفاده از جی‌پی‌یو پیاده‌سازی شد. در این بخش نتایج پیاده‌سازی الگوریتم فیشر بر روی سی‌پی‌یو و جی‌پی‌یو ارائه می‌شود. در شکل (۷) و شکل (۸) خروجی آشکارساز فیشر به ترتیب بر روی سی‌پی‌یو و جی‌پی‌یو نمایش داده شده است. هر یک از این شکل‌ها سه

تعداد کل رشته‌های ایجادشده در کرنل موازی جی‌پی‌یو است. دایره قرمز در شکل (۶) مجموعه بردارهای کنندی با سرعت ۳۳۰ متر بر ثانیه را نشان می‌دهد. با توجه به اینکه کمینه سرعت ظاهری موج ۳۳۰ متر بر ثانیه فرض می‌شود، جستجوی شبکه کنندی تنها شامل نقاط داخل این دایره می‌شود. با افزایش تعداد نمونه‌های بردار کنندی در داخل دایره می‌توان دقت الگوریتم را افزایش داد. این مسئله زمان پردازش را افزایش می‌دهد. به عبارت دیگر، با به کارگیری پردازش موازی دقت الگوریتم را بهبود می‌توان داد و در عین حال، پردازش نیز به صورت بلادرنگ انجام شود.



(شکل-۴): پیکربندی آرایه صوتی (فاصله بر حسب متر).

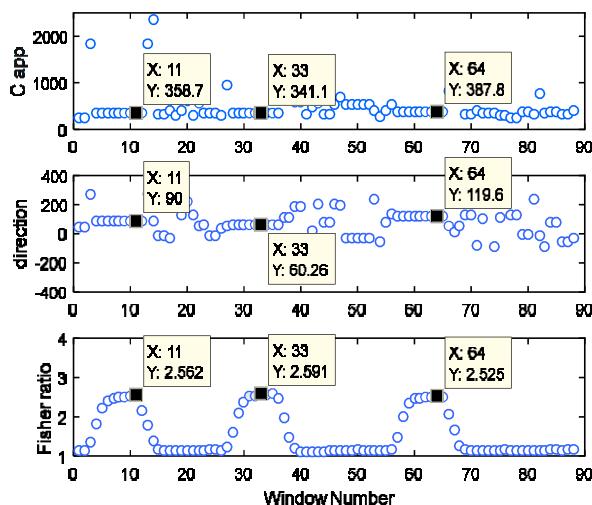
(Figure-4): Acoustic array configuration (distances are in meters).



(شکل-۵): پاسخ آرایه شکل (۴) به ازای

بسامدهای ۱ و ۲ هرتز.

(Figure-5): Array response for the frequencies of 1 and 2 Hz.



(شکل-۸): نتیجه آشکارساز فیشر با استفاده از جی‌پی‌يو شماره ۱ بهازی سیگنالی با سه مؤلفه از سه راستای ۹۰، ۶۰ و ۱۲۰ درجه و سرعت‌های ۳۶۰، ۳۴۰ و ۳۹۰ متر بر ثانیه.

(Figure-8): The results of Fisher detector implementation using the GPU no.1 for a signal received from three sources with directions 90, 60, and 120 with velocities 340, 360, and 390 m/s, respectively.

(جدول-۲) مقایسه پارامترهای تخمین زده شده توسط سی‌پی‌يو و جی‌پی‌يو.

(Table-2): Comparison between the estimated parameters from CPU and GPU implementations.

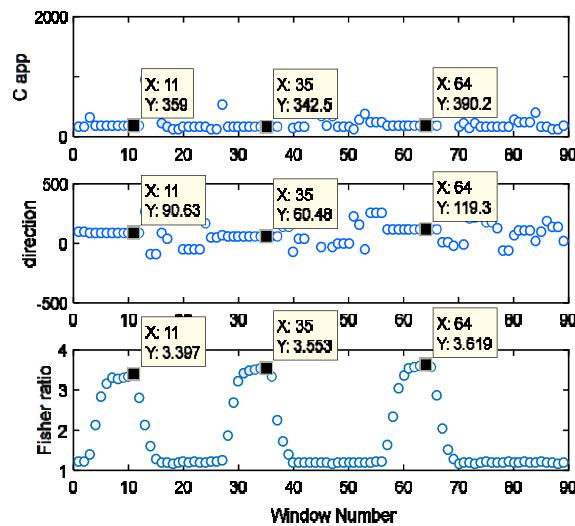
جی‌پی‌يو	سی‌پی‌يو	پارامتر
119.6	119.30	راستای تخمینی (درجه)
60.26	60.48	
90.00	90.63	سرعت تخمینی (متر بر ثانیه)
387.8	390.2	
341.1	342.5	
358.7	359	

۴-۳ مقایسه سرعت پیاده‌سازی بر روی سی‌پی‌يو و جی‌پی‌يو

در این بخش به مقایسه سرعت پیاده‌سازی الگوریتم فیشر بر روی سی‌پی‌يو و جی‌پی‌يو می‌پردازیم. جدول (۳) نتایج مقایسه زمان لازم برای اجرای الگوریتم بهازی شبکه‌های کنده با ابعاد 50×50 ، 100×100 ، 150×150 ، 200×200 و 250×250 را نشان می‌دهد. همان‌گونه که انتظار می‌رود با افزایش تعداد نمونه‌های شبکه کنده حجم بیشتری از محاسبات به صورت موازی انجام شده و درنتیجه نسبت بهبود سرعت جی‌پی‌يو به سی‌پی‌يو افزایش می‌یابد. در این جدول نتایج بهازی هر دو جی‌پی‌يو ذکر شده در قبل ارائه شده است. با توجه به این که ظرفیت اجرای موازی

نمودار دارد. منحنی پایین، خروجی آماره آزمون^۱ آشکارساز فیشر را بهازی پنجره‌های زمانی مختلف نمایش می‌دهد. پنجره‌های متوالی دارای همپوشانی ۸۰٪ هستند. منحنی وسط و بالا به ترتیب راستا و سرعت تخمین زده شده در هر پنجره زمانی را نشان می‌دهد.

همان‌طور که در منحنی‌های مربوط به نسبت فیشر (منحنی پایین) در شکل (۷) و شکل (۸) مشاهده می‌شود، الگوریتم فیشر حضور سه سیگنال را در پنجره‌های زمانی با شماره‌های تقریبی ۱۱، ۳۵ (و ۳۳ در سی‌پی‌يو) و ۶۴ آشکارسازی کرده است. مقادیر راستا و سرعت تخمینی متناظر با هر یک از این سه پنجره در جدول (۲) ارائه شده است. همان‌طور که مشاهده می‌شود، پارامترهای تخمین زده شده توسط جی‌پی‌يو با دقت قابل قبولی، تخمین‌های محاسبه شده توسط سی‌پی‌يو را دنبال می‌کنند که پیاده‌سازی صحیح کد را نشان می‌دهد. شایان ذکر است که تفاوت اندک میان مقادیر تخمینی توسط سی‌پی‌يو و جی‌پی‌يو، ناشی از مواردی همچون دقت متفاوت اعداد با ممیز‌شناور^۲ در محاسبات انجام شده توسط سی‌پی‌يو و جی‌پی‌يو و همچنین نوفره کانال تصادفی تولیدی در شبیه‌سازی‌ها است.



(شکل-۷): نتیجه آشکارساز فیشر با استفاده از سی‌پی‌يو بهازی سیگنالی با سه مؤلفه از سه راستای ۹۰، ۶۰ و ۱۲۰ درجه و سرعت‌های ۳۶۰، ۳۴۰ و ۳۹۰ متر بر ثانیه.

(Figure-7): The results of Fisher detector implementation using CPU for a signal received from three sources with directions 90, 60, and 120 with velocities 340, 360, and 390 m/s, respectively.

¹ Test Statistic

² Floating-point

۵-نتیجه‌گیری

آشکارساز فیشر یکی از مهم‌ترین ابزارهای پردازشی سیگنال‌های بسامد پایین در حوزه تخمین پارامترهای موج (راستا و سرعت) است. در این مقاله با استفاده از پردازش موازی مبتنی بر جی‌پی‌یو به پیاده‌سازی آشکارساز فیشر پرداختیم. نتایج شبیه‌سازی‌ها نشان می‌دهد که با استفاده از جی‌پی‌یو سرعت اجرای این الگوریتم را تا حدود پنجاه برابر نسبت به سی‌پی‌یو می‌توان افزایش داد. البته این بهبود با استفاده از جی‌پی‌یوهای با قدرت محاسبات بالاتر قابل ارتقا است. نتایج بیان گر آن است که پردازش موازی مبتنی بر جی‌پی‌یو امکان تخمین پارامترهای موج و آشکارسازی سیگنال را (با استفاده از الگوریتم‌های موجود در این حوزه) به صورت بلادرنگ تسهیل می‌کند.

6-References

۶-مراجع

- [1] V. L. Zimmer, N. Sitar, "Detection and location of rock falls using seismic and infrasound sensors," *Engineering Geology*, vol. 193, pp. 49-60, Apr. 2015.
- [2] R.D. Costley, W. G. Frazier, et al., "Frequency-wavenumber processing for infrasound distributed arrays," *Journal of Acoustical Society of America*, vol. 134, no. 4, pp. EL307-EL311, 2013.
- [3] S. Havens, H.-P. Marshall, et al., *Real-Time Avalanche Detection for High Risk Areas*, Research Report, Transportation Department, Idaho University, Dec. 2014, available online at: <https://itd.idaho.gov/highways/research/archived/reports/RP219Final12312014.pdf>.
- [4] W. W. Arrasmith, E. R. Coots, J. V. Olson, and E. A. Skowbo, "Analyzing infrasound and seismic signals emanating from a waterborne system using canonical modeling and analysis methods", International Journal of Modeling and Optimization, vol. 4, no. 3, pp. 176-181, Jun. 2014.
- [5] M. Charbit, I. Che, and A. Le Pichon, "Asymptotic distribution of GLRT versus Fisher distribution for infrasonic detection," *Geophysical Research Abstracts*, vol. 15, EGU2013-3690, 2013.
- [6] J. Park, B. W. Stump, C. Hayward, Detection of regional infrasound signals using array data: testing, tuning and physical interpretation, *Journal of Acoustical Society of America*, vol. 140, no. 1, pp. 240-259, Jul. 2016.

محاسبات در سخت‌افزارهای جی‌پی‌یو محدود است، می‌توان مشاهده کرد که با افزایش بیش تر ابعاد شبکه کندی نرخ افزایش سرعت کاهش می‌یابد. در جدول (۴) تأثیر طول پنجره در افزایش سرعت پیاده‌سازی بررسی شده است. در این جدول زمان اجرا بهازای پنجره با طول‌های ۲۵۶، ۵۱۲ و ۱۰۲۴ نمونه ارائه شده است. همان‌طور که مشاهده می‌شود، افزایش طول پنجره تغییر محسوسی در سرعت اجرای الگوریتم توسط جی‌پی‌یو ندارد؛ دلیل این این است که اگرچه با کاهش طول پنجره، تعداد تکرار حلقه موجود در شکل (۲) افزایش می‌یابد؛ ولی محاسبات مورد نیاز برای پردازش داده‌های هر پنجره به همان نسبت کاهش می‌یابد؛ بنابراین با تغییر طول پنجره سرعت پیاده‌سازی الگوریتم توسط جی‌پی‌یو به طور تقریبی ثابت باقی می‌ماند. این در حالی است که با کاهش طول پنجره تعداد تکرار جستجوی شبکه کندی افزایش یافته و زمان اجرای برنامه توسط جی‌پی‌یو بیشتر می‌شود.

(جدول-۳): نتایج مقایسه زمان لازم برای پیاده‌سازی الگوریتم فیشر بر روی سی‌پی‌یو و جی‌پی‌یو بهازای طول پنجره ۱۰۲۴ و هم‌بوشانی٪۸۰ (اعداد بر حسب میلی‌ثانیه است).

(Table-3): Results of time-consumptions measurement for the CPU and GPU implementations of Fisher detector with window length of 1024 samples and overlap rate of 80% (times are in milliseconds).

نسبت افزایش سرعت ۲	زمان جی‌پی‌یو شماره ۲	نسبت افزایش سرعت ۱	زمان جی‌پی‌یو شماره ۱	زمان سی‌پی‌یو	ابعاد شبکه کندی
18.7	1.0	4.8	3.9	18.5	50×50
33.7	2.2	8.7	8.5	74.1	100×100
39.8	4.2	10.3	16.2	164.9	150×150
45.8	6.5	10.7	27.7	299.2	200×200
48.7	9.5	10.9	42.4	460.3	250×250

(جدول-۴) نتایج زمانی پیاده‌سازی الگوریتم فیشر روی سی‌پی‌یو و جی‌پی‌یو بهازای پنجره با طول‌های مختلف (ابعاد شبکه کندی ۱۰۰×۱۰۰ می‌باشد).

(Table-4): Results of time-consumptions measurement for the CPU and GPU implementations of Fisher detector with different window lengths (for slowness network size of 100×100).

نسبت افزایش سرعت ۲	زمان جی‌پی‌یو شماره ۲	نسبت افزایش سرعت ۱	زمان جی‌پی‌یو شماره ۱	زمان سی‌پی‌یو	طول پنجره
49.8	2.8	14.7	9.5	139.4	L = 256
41	2.5	12.1	8.5	102.5	L = 512
33.7	2.2	8.6	8.6	74.1	L = 1024



حامد صادقی مدرک کارشناسی خود را در رشته الکترونیک در سال ۱۳۸۴ از دانشگاه شاهد اخذ کرد؛ سپس دوره های کارشناسی ارشد و دکترای خود را در دانشگاه تربیت مدرس از سال ۱۳۸۷ تا ۱۳۹۲ به اتمام رساند. ایشان در زمینه های مخابرات بی سیم، پردازش سیگنال های صوتی و تئوری آشکارسازی سیگنال، پژوهش های خود را پیش می برد.
نشانی رایانمه ایشان عبارت است از:

h.sadeghi.2015@ieee.org



امیر اخوان مدرک کارشناسی مخابرات خود را در سال ۱۳۸۷ از دانشگاه صنعتی اصفهان، کارشناسی ارشد خود را در سال ۱۳۹۰ از دانشگاه تربیت مدرس و دکترای مهندسی پزشکی خود را در سال ۱۳۹۶ از دانشگاه صنعتی امیرکبیر دریافت کرد. ایشان هم اکنون استادیار دانشگاه صنعتی همدان بوده و پژوهش های خود را در زمینه های پردازش سیگنال های صوتی و سیگنال های حیاتی به پیش می برد.
نشانی رایانمه ایشان عبارت است از:

amir.akhavan@aut.ac.ir

- [7] S. J. Arrowsmith, R. Whitaker, S. R. Taylor, "Regional monitoring of infrasound events using multiple arrays: application to Utah and Washington State", *Geophysical Journal International*, pp. 291–300, Jul. 2008.
- [8] S. J. Arrowsmith, R. Whitaker, C. Katz, C. Hayward, "The F-detector revisited: An improved strategy for signal detection at seismic and infrasound arrays", *Seismological Society of America*, vol. 99, no. 1, pp. 449–453, 2009.
- [9] B. Melton, and L. Baily, "Multiple signal correlators," *Geophysics*, XXII (3), pp. 565-588, 1957.
- [10] R. R. Blandford, "An automatic event detector at the Tonto Forest seismic observatory", *Geophysics*, vol. 39, pp. 633, 1974.
- [11] S. Angelis *et al.*, "Detecting hidden volcanic explosions from Mt. Cleveland Volcano, Alaska with infrasound and ground-coupled airwaves," *Geophysical Research Letters*, vol. 39, 2012.
- [12] L. G. Evers and H. W. Haak, "Tracing a meteoric trajectory with infrasound", *Geophysical Research Letters*, vol. 30, no. 24, Dec. 2003.
- [13] Y. Cansi, "An automatic seismic event processing for detection and location: the PMCC method", *Geophysical Research Letters*, vol. 22, pp. 1021-1024, 1995.
- [14] J. Nickolls, W.J. Dally, "The GPU computing era," *IEEE Micro Magazine*, vol. 30, pp. 56-69, 2010.
- [15] H. Chen, S. Saïghi, L. Buhry, and S. Renaud, "Real-time simulation of biologically realistic stochastic neurons in VLSI," *IEEE Transactions on Neural Networks*, vol. 21, no. 9, pp. 1511–1517, Sep. 2010.
- [16] S. U. Gjerald, R. Brekken, T. Hergum, J. D'hoog "Real-time ultrasound simulation using the GPU," *IEEE Transactions on Ultrasonic Ferroelectrics and Frequency Control*, vol. 59, pp. 885-892, 2012.
- [17] Y. Dai, *et al.*, "Real-time visualized freehand 3D ultrasound reconstruction based on GPU," *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, pp. 1338-1345, 2010.
- [18] C. Richter, S. Schops, and M. Clemens, "GPU acceleration of finite difference schemes used in coupled electromagnetic/ thermal field simulations", *IEEE Transactions on Magnetics*, vol. 49, no. 5, pp. 1649-1652, May 2013.
- [19] W. Rodrigues, *et al.*, "Accelerating atomistic calculation of quantum energy eigenstates on graphic cards", *Computer physics Communications*, pp. 2510-2518, May 2014.
- [20] A. Artu, "Parallel wavelet-based clustering

