Coherent Spatio-Temporal Sensor Fusion on a Hybrid Multicore Processor System

Charlotte Kotas  Eduardo Ponce  Holly Williams  Jacob Barhen*

Abstract. We report on the development, implementation, and demonstration of a novel, massively parallel computational scheme for detection of a target radiating a random signal in the presence of noise. This scheme involves coherent spatio-temporal fusion of data streams from multiple sensors, leading to the derivation of LLR detection statistics. Since streaming multicore processors with multi-SIMT architectures open unprecedented opportunities for fast signal processing, our algorithms are implemented on an NVIDIA Tesla C2050 many-core processor. Results achieved to date demonstrate up to two orders of magnitude speedup over a parallel implementation on a conventional quad-core processor, on a per-target-kinematic-hypothesis basis.

Index Terms — sensor arrays, multicore processors, NVIDIA Tesla, CUDA FORTRAN, spatio-temporal LLR detection

I. Introduction

With the emergence of ever stealthier underwater targets, there is a continuing need to develop innovative approaches for near-real-time remote detection, classification, localization, and tracking (DCLT). In that context, new sensing arrays are being designed and built that will exploit the potential advantages of many novel technologies in complex maritime environments. These arrays are expected to provide mission data necessary for intelligence, surveillance and reconnaissance, for characterizing the environment, and for enabling other ad-hoc capabilities.

To fully exploit the information contained in data measured by these novel devices often requires the use of unconventional algorithms that exhibit growing computational complexity as function of the number of acoustic channels and the number of complex samples in each observation window. For example, signal whitening has long been recognized as an essential stage in processing sonar array data [1], and the Fast Fourier Transform (FFT) has played, for many years, a fundamental role in that context [2-4].

The unconventional twice whitening paradigm includes the inversion of the spatio-temporal covariance matrix for ambient noise. The computational challenge one then faces stems from the following considerations.

An array consisting of $N_s$ acoustic channels, producing $N_c$ complex samples per time window per channel, ($N_s \sim 10^3$, $N_c \sim 10^5$) yields a spatio-temporal covariance matrix of size $10^5 \times 10^5$ for diffuse ambient noise. Its inversion must be carried out for each observation window in real time [5].

This, in turn, implies processing throughput requirements that cannot readily be met with conventional computing hardware and software. In such a context, the emergence of streaming hybrid multicore processors with multi-SIMD or multi-SIMT architectures (e.g., IBM “Cell” [6], NVIDIA Tesla [7]) opens unprecedented opportunities for executing signal processing algorithms faster [8] and within a much lower energy budget. For ultra-low power applications, the Coherent Logix hx3100 [9] and the NVIDIA Tegra [7] devices are of potential interest.

Here, our objective is to report on the development, implementation, and demonstration of a novel, massively parallel computational scheme for Log Likelihood Ratio (LLR) based detection of a target radiating a random signal in the presence of noise. Our paper is organized as follows. Section II presents the basic formalism for coherent spatio-temporal fusion of data streams from multiple sensors, leading to the derivation of LLR detection statistics. Section III focuses on computationally tractable algorithms. Then, in Section IV, we discuss our specific implementation on the NVIDIA Tesla C2050 many-core processor and present results that demonstrate up to two orders of magnitude speedup over a sequential implementation on an Intel Xeon quad-core processor. Finally, conclusions reached so far are summarized in Section VI.

II. Spatio-Temporal Detection

We consider the space-time detection problem of finding a point source in motion that is radiating a Gaussian random signal against a background of Gaussian random noise [10, 11]. Let there be a receiver array consisting of $N_s$ sensors. Without loss of generality, we assume here that each sensor has a single acoustic channel, say pressure. These sensors are arranged in an arbitrary 3-D geometry. Moreover, each element of the array is, in general, assumed to be in motion (i.e., its location can vary with time), and this motion may be...
independent of the motion of any other element in the array. The signal radiated by the source, after sampling, is represented as a Gaussian random vector $\xi$. Conceptually, this vector may be defined in terms of $N_s$ complex random samples in time, and has the following statistics:

$$< \xi > = 0, \quad 0 \in \mathbb{C}^{N_s} \quad \text{with} \quad < \xi \xi^* > = P$$

(1)

Here, the brackets denote expectation, i.e., integration over a complex Gaussian probability distribution $p(\xi)$, the superscript $*$ refers to the Hermitian, and the diagonal elements of the complex matrix $P$ are real. Clearly, $\xi \in \mathbb{C}^{N_s}(0, P)$, with

$$p(\xi) = \frac{1}{\pi^{N_s/2}} \exp[-\xi^* P^{-1} \xi]$$

(2)

The noise vector $\eta$ has similar statistics.

$$< \eta, \eta^* > = H^i, \quad i = 1, \ldots, N_r$$

(3)

By definition, $(H^*)^* = H^i$. Also, $\eta \in \mathbb{C}^{N_s}(0, H^i)$, where $\text{dim}(H^i) = N_s \times N_r$ and

$$p(\eta) = \frac{1}{\pi^{N_r/2}} \exp[-\eta^* (H^*)^{-1} \eta]$$

(4)

We also permit the noise backgrounds at different sensors to be correlated by defining

$$< \eta_i, \eta_j^* > = H^{ij}, \quad (H)_{ij} = H^{ii} \quad i, j = 1, \ldots, N_r$$

(5)

Finally, we assume that the signal is uncorrelated with the background noise seen at any element in the array, so that

$$< \xi_i \xi^*_j > = 0, \quad 0 \in \mathbb{C}^{N_s \times N_r} \quad i = 1, \ldots, N_s$$

(6)

For single target detection, the problem can be simply stated as follows. Let $r(i)$ denote the vector of signal samples, $\xi(i)$ the unknown signal source, and $\eta(i)$ the additive noise, as received respectively at the $i$-th element of the sensor array. Given combined (stacked) complex array variables $r$, $\xi$, $\eta$, where, for example, $r = [r(i_1), r(i_2), \ldots, r(i_{N_r})]^T$ denotes a vector of size $N_r = N_s \times N_r$, we must choose between two hypotheses $H_0$ and $H_1$, such that

$$H_0 \quad (\text{source absent}): \quad r = \eta$$

(7)

$$H_1 \quad (\text{source present}): \quad r = Q s + \eta$$

(8)

In Eq (8), $Q = [Q^1, Q^2, \ldots, Q^{N_s}]^T$, $\text{dim}(Q) = N_r \times N_s$, and $s$ refers to the (unknown) spectrum of the source signal. Here, we have assumed that the signal received from the source at any sensor $i$ may be written as $Q^i s$, $\forall i$ [11]. $Q^i$ is an $N_s \times N_r$ complex matrix that is assumed to be non-random and known. It represents the net transfer function between the complex spectrum generated by the source and the time samples measured by array element $i$. Thus, it encompasses physical propagation effects that the environment and the geometry impose on the signal.

Using discrete approximations of the Green’s function, $Q^i$ may be constructed in a fashion that fully accounts for both source and receiver motion.

The log likelihood ratio (LLR) $\mathcal{L}$ is a quantity providing critical information for making a decision between two hypotheses. The corresponding statistical test uses the natural logarithm of the ratio of the probability densities of the observable $r$ under the two hypotheses $H_1$ and $H_0$, respectively [1, 11]. To compute $\mathcal{L}$, one must first derive expressions for these probability distributions, which involves the integration of complex Gaussian random processes. Specifically, in our context

$$p_r(r | H_0) = \frac{1}{\pi^{N_r/2}} \exp[-r^* H^{-1} r]$$

$$p_r(r | H_1) = \frac{1}{\pi^{N_r/2}} \exp[-\beta]$$

(9)

(10)

Then, after some algebra,

$$\mathcal{L} = \log \frac{p_r(r | H_1)}{p_r(r | H_0)} = z^T A^{-1} z - \log(|H|)$$

(11)

where $A = P^{-1} + Q^* H^{-1} Q$, and the full spatio-temporal covariance matrix of the noise is denoted as $H$. The spatial components refer to sensors, while temporal components refer to measured samples. Specifically, matrix $H$ has the following structure:

$$H = \begin{bmatrix}
H^1 & H^{12} & \cdots & H^{1N_s} \\
H^{12} & \ddots & \vdots & \vdots \\
\vdots & \ddots & \ddots & \vdots \\
H^{1N_s} & \cdots & \cdots & H^{N_s}\end{bmatrix}$$

(12)

In Eq (11), $z$ is a vector representing “twice whitened” data, obtained via spatio-temporal matched filtering [11]

$$z = Q^* H^{-1} r$$

(13)

This expression of $z$ (introduced in [5]) to avoid replica whitening during target search), is distinct from the conventional expression for whitening data, which is usually written $z = (r - Q s)^* H^{-1} (r - Q s) + s^* P^{-1} s$.

In a recent article [12], we have shown that by exploiting intrinsic features of the spatially additive noise in conjunction with the FFT, we can reduce this complexity to order $N_r \times (N_r)^3$. In a recent article [12], we have shown that by exploiting intrinsic features of the spatially additive noise in conjunction with the FFT, we can reduce this complexity to order $N_r \times (N_r)^3$. Given that $N_r \ll N_s$, this reduction in the cost of spatio-temporal data whitening was found to be very significant.

Since the problem under consideration is the detection of a target radiating a random signal against a background of random noise, direct evaluation of Eq (11) is of limited value. Instead, what need to be computed are the LLR detector statistics, i.e., the moments of $\mathcal{L}$ under the hypotheses $H_0$ and $H_1$. For instance, we need to calculate
<L>_{0} = \left\{ \int d \mathbf{r} L(\mathbf{r}) p_{i}(\mathbf{r} | \mathcal{H}_{0}) \right\} \\
= \frac{1}{\pi^{\nu}|H|} \left\{ \int d \mathbf{r} (x^{*} \mathbf{A}^{*} \mathbf{A} - \mathbf{H}^{*} \mathbf{H}) \exp[-r^{*} \mathbf{H}^{*} \mathbf{r}] \right\} \\
- \frac{1}{\int d \mathbf{r} \log_{2}(|\mathbf{B}|) \exp[-r^{*} \mathbf{H}^{*} \mathbf{r}]} 
\tag{14}
\]

After some non-trivial manipulations using Wick’s theorem [13], one obtains
\[
< L >_{0} = \frac{1}{\pi^{\nu}|H|} \sum_{i,j} (\mathbf{H}^{*} \mathbf{Q} \mathbf{A}^{*} \mathbf{Q}^{*} \mathbf{H})_{i,j} \mathbf{H}_{i,j} \]
\[
- \log_{2}(|\mathbf{P} \mathbf{A}|) \pi^{\nu}|H| 
\tag{15}
\]

Observe that the first term on the RHS of Eq (15) involves a Frobenius product of matrices, which could be expressed in terms of a trace operator. One can derive in similar manner expressions for other moments of L. However, it is clear from Eq (15) that computational tractability, particularly for problems of the size mentioned in our Introduction, becomes the primary challenge.

III. Tractable Algorithms

The fundamental idea that enabled advances in coherent spatio-temporal sensor fusion in terms of computational tractability is due to Polcari [11]. It is referred to as the Quasi-Static Approximation. In this paradigm, data streams from sensors in the array are divided into time slices (called “frames”). Within each frame, source and receiver are treated as stationary, but between frames source and receivers are assumed to change location by discrete frame-to-frame steps.

We begin by specifying the actual construction of Q'. Let R_{i} = |\mathbf{x}^{*} - \mathbf{x}^{i}| be the range between the source and receiver i in a given time frame. Assume that the source begins transmitting at time t_{0} = 0. The initial wave front reaches a detector element (sensor) i at source time t_{0} + m_{i} \Delta_{t}, where m_{i} = R_{i} / c_{0} \Delta_{t}. The detector will begin collecting N_{f} temporal samples indexed by m = 0, 1, ..., N_{f} - 1 starting at detector time t_{0} = 0. These will have been generated at the source at times \( (m - m_{i}) \Delta_{t} \) detector time coordinates, transformed there into the spectral domain, propagated via Q' to the detector, and finally recast back (again via Q') into the temporal domain. Thus,

\[
Q_{m_{i}k}^{i} = \frac{1}{4\pi R_{i} \sqrt{N_{f}}} e^{\frac{2\pi}{N_{f}}(m - m_{i})k} 
\tag{16}
\]

\[
s_{k}^{i} = \frac{1}{\sqrt{N_{f}}} \sum_{m_{i} = 1}^{N_{f}} e^{\frac{2\pi}{N_{f}}(m - m_{i})k} 
\tag{17}
\]

In the above expressions, both Q' and s are in detector time coordinates, and m_{i} is constant. If the source to receiver i range is constant over a time frame, Q' may be factored in terms of the inverse (unitary) Fourier Transform operator F (written in matrix form) and a diagonal matrix \( \Phi^{i} \), i.e.,
\[
Q' = F \Phi^{i} 
\tag{18}
\]

This can readily be seen. Splitting Eq (16), we get
\[
Q_{m_{i}k}^{i} = \frac{1}{\sqrt{N_{f}}} e^{\frac{2\pi}{N_{f}}(m - m_{i})k} 
\times \frac{1}{4\pi R_{i}} e^{-\frac{2\pi}{N_{f}}m_{i}k} 
\tag{19}
\]

The first term on the LHS is the \((m, k)\)-th element of the inverse FT, operator, whereas the second term is identified with the \((k, k)\)-th (diagonal) element of a new matrix labeled \( \Phi^{i} \), where
\[
\Phi^{i}_{kk} = \frac{1}{4\pi R_{i}} e^{-\frac{2\pi}{N_{f}}m_{i}k} \quad \text{dim} \Phi = N_{f} \times N_{f}. 
\tag{20}
\]

\( \Phi^{i}_{kk} \) is the phase delay associated with a frequency domain beamformer at frequency bin \( k \). The scalar amplitude term keeps the detector scaled by properly accounting for propagation loss.

We next decompose each sensor’s time series into a sequence of \( L_{f} \) “frames” consisting of \( N_{f} \) time samples, so that \( N_{f} = L_{f} N_{f} \). Then
\[
Q' = \left[ \begin{array}{c} Q_{1}^{i} \\
\vdots \\
Q_{L_{f}}^{i} 
\end{array} \right] 
\tag{21}
\]

where each \( Q_{l}^{i} (l = 1, ..., L_{f}) \) is an \( N_{f} \times N_{f} \) complex matrix. Note that \( N_{f} \) refers to the time dimension, whereas \( N_{f} \) refers to the frequency dimension of the transfer matrix.

One can reorganize \( Q \) to highlight the frame contributions in the quasi-static approximation. Let the transfer matrix for element \( i \) and frame \( l \) be denoted by the \( N_{f} \times N_{f} \) complex matrix \( Q_{q_{i}q_{j}}^{i} \), with
\[
Q_{q_{i}q_{j}}^{i,l} Q_{q_{j}q_{k}}^{i,l} = \left[ \begin{array}{c} Q_{q_{i}q_{j}}^{i} \\
\vdots \\
Q_{q_{i}q_{k}}^{i} 
\end{array} \right] 
\tag{22}
\]

Each of the \( L_{f} - 1 \) matrix blocks \( \Phi \) is of size \( N_{f} \times N_{f} \). The factorization of Eq (18) is then rewritten \( Q_{q_{i}q_{j}}^{i,l} = F \Phi_{q_{i}q_{j}}^{i,l} \) and
\[
(\Phi_{q_{i}q_{j}}^{i})_{kk}^{l} = \frac{1}{4\pi R_{i}(l)} e^{-\frac{2\pi}{N_{f}}m_{i}k} 
\tag{23}
\]

Note that in Eq (23) there is a different source-sensor \( i \) range for each frame \( l \). Then, the quasi-static steering vector at frequency bin \( k \) (a complex vector of \( N_{f} \) components) is defined as
\[
\Phi_{q_{i}q_{j}}^{i}(k) = \left[ \begin{array}{c} (\Phi_{q_{i}q_{j}}^{i})_{k}^{1} \\
(\Phi_{q_{i}q_{j}}^{i})_{k}^{2} \\
\vdots \\
(\Phi_{q_{i}q_{j}}^{i})_{k}^{N_{f}} 
\end{array} \right] 
\tag{24}
\]
To further reduce computational complexity one can introduce an orthogonal permutation matrix \( \Omega \) such that

\[
\begin{bmatrix}
\Phi_{\text{sp}}^{i,l}
\Phi_{\text{pp}}^{i,l}
\vdots
\Phi_{\text{pp}}^{N_s,N_s-1,l}
\end{bmatrix} = \Omega \begin{bmatrix}
\mathbf{\phi}_{\text{sp}}^0(0)
\mathbf{\phi}_{\text{sp}}^0(1)
\vdots
\mathbf{\phi}_{\text{sp}}^0(N_s-1)
\end{bmatrix}
\begin{bmatrix}
0
0
\vdots
0
\end{bmatrix}
\begin{bmatrix}
\mathbf{\phi}_{\text{pp}}^1(0)
\mathbf{\phi}_{\text{pp}}^1(1)
\vdots
\mathbf{\phi}_{\text{pp}}^1(N_s-1)
\end{bmatrix}
\begin{bmatrix}
0
0
\vdots
0
\end{bmatrix}
\begin{bmatrix}
\mathbf{\phi}_{\text{pp}}^{N_s,N_s-1}(0)
\mathbf{\phi}_{\text{pp}}^{N_s,N_s-1}(1)
\vdots
\mathbf{\phi}_{\text{pp}}^{N_s,N_s-1}(N_s-1)
\end{bmatrix}
\tag{25}
\]

This factorization can now be used to efficiently compute the LLR detection statistics.

**IV. Practical Details**

We now provide a step-by-step summary for the computation of \( \mathbf{z} \).

a) Given the data vector \( \mathbf{r} \) collected from all \( N_s \) channels of a sensor array (with \( \text{dim} \mathbf{r} = N_s N_s = N_s^2 \)), spatio-temporally “twice”-whiten \( \mathbf{r} \) via

\[ \mathbf{x} = \mathbf{H}^{-1} \mathbf{r} \]

where \( \mathbf{H}^{-1} \) is the \( N_s \times N_s \) full covariance matrix for the window under consideration. This is done once per integration window, using the most efficient implementation techniques (e.g., [5, 11]).

b) Partition \( \mathbf{x} \) into segments \( \mathbf{x}'; \) each segment corresponds to a sensor channel. Then, partition each \( \mathbf{x}' \) into \( L_f \) contiguous data frames, the length of each frame being \( N_f \) data samples.

c) For each frame, extract the corresponding \( N_f \) data samples for each channel and zero-pad to form \( N_s \) vectors of length \( N_s \). For example, for frame \( i \) and array channel \( j \) one would obtain the vector

\[
\begin{bmatrix}
\mathbf{x}_{i,j,t}^{N_f}
0
\end{bmatrix}
\]

where \( \text{dim} \mathbf{x}_{i,j,t}^{N_f} = N_f \), and \( \text{dim} \mathbf{0} = (L_f - 1) N_f \).

d) Compute \( N_s \)-dimensional \( N_s \)-point forward Fourier transforms for frame \( i \)

\[ \mathbf{y}_{i,t}^{N_f} = \mathbf{F}^{+} \begin{bmatrix}
\mathbf{x}_{i,t}^{N_f}
0
\end{bmatrix} \]

Repeat for all frames up to \( L_f \). Store (concatenate) \( \mathbf{y}_{i,t}^{N_f} \) for all \( i \) into the vector \( \mathbf{\tilde{y}} \) of dimension \( N_s \).

e) “Corner-turn” the FFT outputs. The goal is to group together (i.e., store contiguously) all \( \mathbf{y} \) - array elements at each frequency. This is done once per frame, in each integration window. Formally, the operation is achieved by multiplying the FFT outputs by the orthogonal permutation matrix \( \Omega^T \), i.e.,

\[ \mathbf{\Omega}_y^T = \begin{bmatrix}
\mathbf{F}^+ \begin{bmatrix}
\mathbf{x}_{i,t}^{N_f}
0
\end{bmatrix}
\vdots
\mathbf{F}^+ \begin{bmatrix}
\mathbf{x}_{i,t}^{N_f}
0
\end{bmatrix}
\end{bmatrix} = \begin{bmatrix}
\mathbf{y}_{i,t}^{N_f}(k = 0)
\vdots
\mathbf{y}_{i,t}^{N_f}(k = N_f - 1)
\end{bmatrix} = \mathbf{\tilde{y}} \]

Note that \( \mathbf{\tilde{y}} \) comprises \( N_s \) one-dimensional blocks, each of size \( N_f \), as shown in Figure 1.

f) The corner-turned FFT output vectors for each frame \( i \) in the integration window, \( \mathbf{\tilde{y}}_i \), are multiplied by the Hermitian of the amplitude and phase shading matrix \( \mathbf{\Phi}^+ \), i.e., by

\[
\mathbf{\Phi}^+ = \begin{bmatrix}
\mathbf{\phi}_{\text{sp}}^0(0)
\mathbf{\phi}_{\text{sp}}^0(1)
\vdots
\mathbf{\phi}_{\text{sp}}^0(N_s-1)
\end{bmatrix}
\begin{bmatrix}
0
\mathbf{\phi}_{\text{pp}}^1(0)
\vdots
\mathbf{\phi}_{\text{pp}}^1(N_s-1)
\end{bmatrix}
\begin{bmatrix}
0
0
\vdots
0
\end{bmatrix}
\begin{bmatrix}
\mathbf{\phi}_{\text{pp}}^{N_s,N_s-1}(0)
\mathbf{\phi}_{\text{pp}}^{N_s,N_s-1}(1)
\vdots
\mathbf{\phi}_{\text{pp}}^{N_s,N_s-1}(N_s-1)
\end{bmatrix}
\]

Since \( \mathbf{\Phi}^+ \) is function of a target kinematic or spectral hypothesis, this multiplication needs to be done for each hypothesized beam (there are \( N_b \) such beams).

g) Observe that summation over elements is automatically included in the matrix-vector multiplication \( \mathbf{\Phi}^+ \cdot \mathbf{\tilde{y}} \). Summation over frames has to be performed. All this has to be done for each hypothesized beam.

h) The vector \( \mathbf{z} \) has \( N_s \) components. Each component represents output at a particular frequency \( f_i \) with \( k = 0,...,N_s - 1 \). Of course, \( \mathbf{z} \) is different for each hypothesized beam.

In the following Section, we will implement these concepts to illustrate how the LLR detection statistics are obtained and efficiently computed on the many-core Tesla processor.
V. Implementation

Four parameters are of paramount importance when evaluating the relevance of emerging computational platforms for time-critical, embedded applications. They are: computational speed, communication speed (the I/O and inter-core data transfer rates), the power dissipated (usually measured in pico-Joules per floating point operation), and the processor footprint. For each of these parameters, one can compare the performance of an algorithm for different hardware platforms and software (e.g., compilers) tools. In that context, several multicore processors are of relevance for computationally intensive maritime sensing applications. These include the IBM Cell Broadband Engine, the Coherent Logix HyperX hx 3100, and the NVIDIA Tesla. In this study we exploit the capabilities of the latter.

The NVIDIA Tesla C2050 GPU [7] used in our system operates in conjunction with an Intel Xeon X5677 3.46 GHz CPU that accesses 24 GB of ECC DDR3 RDIM 1333 MHz memory with ECC. The operating system is the 64-bit edition of Windows 7. Our algorithms are programmed in FORTRAN (2003 standard), in CUDA FORTRAN (on the Tesla GPU), and in Assembly (timing routines) using a compiler provided by the Portland Group Inc [14]. NVIDIA GPUs exploit the CUDA architecture [15], which, for the Tesla (C2050, Fermi architecture), is built in terms of an array of 14 Streaming Multiprocessors (SMs). Each SM consists of 32 scalar processor cores. This results in a total of 448 cores for the Tesla. In addition, an SM includes two special units for transcendental functions, a multithreaded instruction unit, and 16 KB of shared memory. There are 3 GB of GDDR5 global RAM available to the GPU, and its clock runs at 1.15 GHz. From a computational perspective, the fundamental underlying paradigm is the concept of scalar thread. Threads are grouped in blocks that execute concurrently on one SM with zero overhead [15]. Massive parallelism is achieved in terms of a kernel grid comprising the thread blocks. As any thread block terminates, a new block is launched on the first available (vacated) SM.

Before computing the detection statistics, we make the following observations. The signal covariance matrix can be expressed in the frequency domain (notation $\Sigma_s$) in the following manner. By definition,

$$ P = F \Sigma_s F^T $$

Thus,

$$ \Omega^T F^T P F \Omega = \begin{bmatrix} \sigma_s^2(f_s,t_l) & 0 & \cdots & 0 \\ 0 & \sigma_s^2(f_s,t_l) & \cdots & \vdots \\ \vdots & \cdots & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_s^2(f_s,t_{l-1}) \end{bmatrix} $$

In the above expression $\sigma_s^2(f_s,t_l)$ is simply the expected beam signal power $<BSP(f_s,t_l)>$ in frequency bin $k$ and frame $l$.

A replica, in frequency bin $k$ and frame $l$, is identified with the corresponding quasi-static steering function. Thus,

$$ e(f_k,t_l) \equiv l \varphi_{qs}(k) = \begin{bmatrix} (\Phi_{qs})_{k,k}^{1/l} \\ (\Phi_{qs})_{k,k}^{2/l} \\ \vdots \\ (\Phi_{qs})_{k,k}^{N/l} \end{bmatrix} $$

Hence, the expected beam noise power $<BNP(f_k,t_l)>$ is given by

$$ <BNP(f_k,t_l)> = \left[ e^*(f_k,t_l) \Sigma^{-1}_n(f_k,t_l) e(f_k,t_l) \right]^{-1} $$

With these definitions, we can compute the expected signal-to-noise ratio. By definition,

$$ E_{k,l} = E(f_k,t_l) = <BSP(f_k,t_l) > <BNP(f_k,t_l)> $$

Thus,

$$ E_{k,l} = \sigma_s^2(f_k,t_l) \left[ e^*(f_k,t_l) \Sigma^{-1}_n(f_k,t_l) e(f_k,t_l) \right] $$

Now, let

$$ \delta(f_k,t_l) \equiv l \gamma = \Omega^T \gamma = \begin{bmatrix} \gamma_{1/l}^{x}(k = 0) \\ \vdots \\ \gamma_{N/l}^{x}(k = 0) \\ \vdots \\ \gamma_{1/l}^{y}(k = N_y - 1) \\ \vdots \\ \gamma_{N/l}^{y}(k = N_y - 1) \end{bmatrix} $$

The signal-plus-noise-to-noise ratio can then be expressed as

$$ S_{k,l} = \frac{|e^*(f_k,t_l) \Sigma^{-1}_n(f_k,t_l) \delta(f_k,t_l)|^2}{e^*(f_k,t_l) \Sigma^{-1}_n(f_k,t_l) e(f_k,t_l)} $$

With these fundamental building blocks, the LLR and the corresponding detection statistics are easily computed. As stated before, these computations assume: (1) quasi-stationary motion in a frame interval, (2) sequential frames statistically independent of each other for both signal and noise effects, and (3) short channel impulse responses. The LLR is then given by the following expression:

$$ \Lambda = \sum_{k=1}^{K} \sum_{l=1}^{L} \frac{E_{k,l}}{1 + E_{k,l}} S_{k,l}^{x} - \text{Log}_e (1 + E_{k,l}) $$
Finally, the formulae for resulting detection statistics are given below. For the first moments, we have

\[
\langle \mathcal{L} \rangle_0 = \sum_{k=1}^{k=N_f} \sum_{l=1}^{l=L_f} \left\{ \frac{E_{k,l}}{1 + E_{k,l}} - \log_e(1 + E_{k,l}) \right\} \tag{37}
\]

\[
\langle \mathcal{L} \rangle_1 = \sum_{k=1}^{k=N_f} \sum_{l=1}^{l=L_f} \left\{ E_{k,l} - \log_e(1 + E_{k,l}) \right\} \tag{38}
\]

For the second moments, one obtains

\[
\text{var}(\mathcal{L}|\mathcal{H}_0) = \sum_{k=1}^{k=N_f} \sum_{l=1}^{l=L_f} \left( \frac{E_{k,l}}{1 + E_{k,l}} \right)^2 \tag{39}
\]

\[
\text{var}(\mathcal{L}|\mathcal{H}_1) = \sum_{k=1}^{k=N_f} \sum_{l=1}^{l=L_f} E_{k,l}^2 \tag{40}
\]

The following parameters were used to test the massively parallel implementation. The sensor array under consideration included 176 active acoustic channels. There were 400 frequency bins partitioned into 8 frequency segments. A total of 180 infinite-range-directions hypotheses were explored. The sensor data stream was processed in terms of 30 seconds observation windows, each window partitioned into 30 non-overlapping frames. The data sampling rate was 1 KHz. The timing results reported hereafter refer to conventional beam-forming [1, 10].

In Table 1, three code versions are compared. MATLAB refers to a direct implementation in MATLAB running on the Intel Xeon host of the hybrid computational platform. OpenMP refers to a version implemented also on the Xeon processor, but parallelized using OpenMP (with number of threads equal to 4). Finally, GPU refers to the algorithms implementation on the Tesla GPU processor coded in CUDA FORTRAN. Computations were run in double precision.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATLAB</td>
<td>0.01060</td>
</tr>
<tr>
<td>OpenMP</td>
<td>0.00281</td>
</tr>
<tr>
<td>GPU</td>
<td>0.00021</td>
</tr>
</tbody>
</table>

**Table 1.** Mean time per target kinematic hypothesis

The mean times reported in Table 1 were calculated as follows. For the MATLAB and OpenMP codes, separate time estimates were initiated whenever a new replica generation was performed (i.e., once for each hypothesis and frequency segment), and thereafter computing the mean of these time estimates. For the massively parallel GPU version, timing estimates are started for each block of hypotheses, dividing by the number of hypotheses computed, and computing the mean of this time per hypothesis.

VI. Summary and Conclusions

To achieve the real-time and low power performance required for maritime sensing and other computationally demanding applications, many existing algorithms may need to be revised and adapted to the emerging revolutionary computing technologies. Novel hardware platforms of interest to naval applications include the IBM Cell, the Coherent Logix HyperX, and the NVIDIA Tesla (Fermi) devices.

In this article, we have developed, implemented, and demonstrated a novel algorithm for phase-coherent spatio-temporal fusion of data streams from multiple sensors, leading to the derivation of LLR detection statistics. Our implementation on the NVIDIA Tesla C2050 many-core processor indicates possible speedups of close to two orders of magnitude over a sequential and even host-multicore-parallel implementation.

The recent release by PGI of the high-performance CUDA FORTRAN compiler for the NVIDIA Tesla opens, via mixed language (C and FORTRAN) programming an optimal framework for ultra-fast future implementations. Such a framework would fully exploit the intrinsic array language, compiler optimization and numerical capabilities of FORTRAN 2003 (at the base of CUDA FORTRAN) in conjunction with the DMA and system capabilities of C.

Finally, we believe that in the longer term, the emergence of multicore devices will enable the implementation of novel, more powerful information-processing paradigms that could not be considered heretofore.

**Acknowledgments**

The authors would like to thank John Polcari for introducing them to his innovative approach for spatio-temporal target detection, Joshua Gershone for helpful discussions on code implementation, and to Mike Traweek for supporting this research effort.

**References**

7. www.nvidia.com
9. www.coherentlogix.com


---

Fig 1. Illustration of corner turning operation $\bar{\mathbf{y}} = \mathbf{\Omega}^T \mathbf{y}$ for $N_e = 2$, $N_f = 6$, $L_f = 3$ and $l = 3$. 

Matrix $\mathbf{\Omega}^T$

- $\square = 1$
- $\blacksquare = 0$

Vector $\mathbf{y}$

Components of: $\mathbf{y}^{1,l}$, $\mathbf{y}^{2,l}$

Vector $\bar{\mathbf{y}}$: Components grouped by array elements for increasing frequencies