GPU-based real-time beamforming for large arrays of optical wireless acoustic sensors

Gabriel Pablo Nava¹*, Hoang Duy Nguyen¹, Yusuke Hioka², Yutaka Kamamoto¹, Takashi G. Sato¹, Yoshifumi Shiraki¹ and Takehiro Moriya¹

¹NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation, 3–1, Morinosato Wakamiya, Atsugi, 243–0198 Japan
²Department of Mechanical Engineering, University of Auckland, Private Bag 92019, Auckland 1142, New Zealand

(Received 30 January 2015, Accepted for publication 21 July 2015)

Abstract: Recent optical wireless acoustic sensors have demonstrated the possibility to simultaneously sense massive numbers of audio channels in real time. Although this technology has enabled the deployment of large-scale applications, it raises new challenges from the computational perspective. In this regard, Graphics Processing Units provide significant parallel computational power. However, not all the existent algorithms are GPU-implementable in a straightforward way. This paper discusses signal processing schemes and implementation strategies to achieve real-time broadband beamforming using a single GPU card. The experiments introduced here, show our prototype implementation handling over 120 audio channels in real time. The experimental results further highlight the particular advantages of using a video camera-based approach to improve the beamforming performance.

Keywords: Optical wireless sensors, Broadband beamforming, GPU computing

PACS number: 43.60.+d [doi:10.1250/ast.36.489]

1. INTRODUCTION

Over the years, acoustic beamforming has found a broad range of applications including noise reduction, hands-free audio interfaces, sound source localization, and acoustic field mapping, to mention some. Along its study, it has been well known that the performance of a beamforming array improves as the array size increases [1,2]. However, the construction of large microphone arrays has been hindered by the cost and complexity of the hardware in the case of wired microphones, and by constraints on the available radio frequency bandwidth when wireless solutions (e.g. [3,4]) are considered, leaving post-processing filters [5–8] as the only choice to achieve optimum array performance with a limited number of microphones.

During the past decade, acoustic sensors with optical output have already started to emerge as an alternative to allow for the deployment of arrays with higher degree of scalability [9,10]. Such sensors rely on free-space optical communication to convey information of the acoustic measurement from the sensor to the processing central.

Particularly, those array systems based on the combination of light emitting diodes (LEDs) and video cameras offer the possibility of capturing the audio signals from multiple sensors at the same time. Indeed, a multichannel audio system with such capability has already been demonstrated in recent literature [11,12]. The prototype described in [11] consists of an array of optical wireless acoustic sensors (OWAS) and a high speed video camera. The OWAS device, such as the one illustrated in Fig. 1, is fundamentally made of a digital microphone, an FPGA-based processor, and a 4 x 4 matrix of LEDs. The digital microphone samples the acoustic waves at the rate $f_s = 16$ kHz, and the field programmable gate array (FPGA) converts the samples into parallel binary data encoded in 16-bit PCM format. Each bit drives one LED into ON or OFF state at the same rate. The encoding process takes place in a period of a few microseconds, therefore the transmission of audio samples through optical signals can be considered to be in real time. On the receiver side, the high speed camera records images of the optical signals from a number of OWAS at the rate $f_c = f_s$ frames per second. The recorded images are immediately transferred to a Graphics Processing Unit (GPU) server where all the
images are simultaneously analyzed with massive parallel processing to decode the optical signals and recover the multichannel audio originally transmitted by the OWAS array. Details of the image decoding process have been published elsewhere [12].

On one hand, the parallel channel transmission offered by the combination of LEDs and video cameras effectively makes an OWAS array highly scalable within the limits of the field of view of the camera. Moreover, the positional information retrievable from the images adds further flexibility to deploy complex array geometries. But on the other hand, such a massive audio recording paradigm raises new challenges from the computational perspective. In order to sustain a real-time factor not only at the audio acquisition level but also in the subsequent multichannel signal treatment, namely beamforming, GPU-based parallel computing turns out to be a promising solution. Hence, this paper is devoted to introduce parallel signal processing strategies (Sect. 3), together with fully detailed GPU implementations (Sect. 4), to achieve real-time acoustic beamforming with large arrays. The validation of these implementations will be shown with experiments, together with a discussion on the advantages of using arrays of OWAS (Sect. 6). But first, let us elaborate on the base acoustic array model in the following section.

2. BASE SIGNAL AND ARRAY MODEL

Consider a planar array of \( N_1 \times N_2 \) identical audio sensor elements arranged as depicted in Fig. 2. \( N_1 \) and \( N_2 \) are the numbers of microphones in the \( \hat{x} \) and \( \hat{y} \) directions, and \( d_1 \) and \( d_2 \) represent the inter-element spacing in the two orthogonal directions. Microphone elements are denoted by the indexes \( (n_1, n_2) \), where \( 0 \leq n_1 \leq N_1 - 1 \) and \( 0 \leq n_2 \leq N_2 - 1 \), respectively. Under far-field assumptions first, planar wavefronts from a broadband source of interest impinge on the reference microphone \((0,0)\) from a direction \((\theta_d, \phi_d)\), where \(\theta_d\) and \(\phi_d\) denote the elevation and azimuth angles, respectively. The particular case where the source is located at a near-field distance to the microphone array will be addressed in the appendix.

Let the broadband signal received on the \( m \)th array element be denoted by

\[
x_m(t) = s_m(t) + n_m(t),
\]

where \(m = \{0, \ldots, N_1 N_2 + n_2, \ldots, N_1 N_2 - 1\} \), \(s_m(t)\) is the desired noiseless signal and \(n_m(t)\) represents the cumulative effect of noise and interference observed at the output of the \( m \)th sensor.

Each array element is followed by an additive network of transversal filters with \( J \) weighting coefficients and tap spacing \( T_s = 1/f_s \), as shown in Fig. 3. The output signal of this Finite Impulse Response (FIR) filter network constitutes the beamformer output \( y(t) \). If \( x_{m,i}(t) = x_m(t - iT_s) \), and \( w_{m,i} \) represents the filter weight for the \( m \)th channel at the \( i \)th tap \((i = 0, \ldots, J - 1)\), then

\[
y(t) = \sum_{m=0}^{N_1 N_2 - 1} \sum_{i=0}^{J-1} x_{m,i} w_{m,i},
\]

which can be expressed in the matrix form [13]:

\[
y(t) = \mathbf{w}^T \mathbf{x}(t),
\]

where \(\mathbf{w}\) and \(\mathbf{x}(t)\) are \(N_1 N_2 J \times 1\) dimensional vectors defined as

\[
\mathbf{w} = [\mathbf{w}_0^T \ldots \mathbf{w}_{J-1}^T],
\]

\[
\mathbf{x}(t) = [\mathbf{x}_0^T(t) \ldots \mathbf{x}_{J-1}^T(t)].
\]

Here, \((\cdot)^T\) denotes the transpose operator and each \(N_1 N_2 \times 1\) dimensional vector \(\mathbf{w}_i\) and \(\mathbf{x}_i(t)\) is defined as

\[
\mathbf{w}_i = [w_{0,i} \ldots w_{N_1 N_2 - 1,i}]^T,
\]

\[
\mathbf{x}_i(t) = [x_{0,i}(t) \ldots x_{N_1 N_2 - 1,i}(t)]^T.
\]
Assuming that plane waves arrive on the broadside of the sensor array, the Linearly Constrained Minimum Variance (LCMV) beamformer is formulated as follows:

$$w_{LCMV} = \arg \min_w w^H R_s w \text{ subject to } C^H w = f, \tag{8}$$

where $R_s = E[x^T x]$ is the correlation matrix of the input signals, $C \in \mathbb{R}^{N_c \times J}$ and $f \in \mathbb{R}^J$ denote the constraints matrix and response vector, respectively, as defined in \cite{13,14}. The well-known solution to the minimization problem (8) is given by the Wiener solution \cite{15}:

$$w_{opt} = R_s^{-1} C (C^H R_s^{-1} C)^{-1} f, \tag{9}$$

and the adaptive version of (9) is given by the stochastic constrained Least Means Squares (LMS) algorithm:

$$w(0) = g, \tag{10}$$
$$w(n + 1) = P(w(n) - \mu (n)x(n)) + g, \tag{11}$$

where $g \in \mathbb{R}^{N_c J \times 1}$ and $P \in \mathbb{R}^{N_c J \times N_c J}$ are defined as

$$g = C (C^T C)^{-1} f, \tag{12}$$
$$P = I_{N_c J \times J} - C (C^T C)^{-1} C^T, \tag{13}$$

and $\mu$ is the scaling factor which controls the steady-state performance of the LCMV beamformer. The choice of $\mu$ is a tradeoff between convergence time and misadjustment from the Wiener solution, as addressed extensively in \cite{13}. A sufficient condition for upper bounded steady-state misadjustment is $\mu < 2/(3E[x^T x])$.

On the basis of this fundamental array model, we are ready to build up an efficient scheme for parallel processing in a real-time beamforming implementation.

### 3. PROPOSED SCHEMES FOR PARALLEL BEAMFORMING PROCESSING

#### 3.1. Steering the Beamforming Array

The beamforming design criterion given by (9) assumes planar wavefronts incoming from the broadside of the array, i.e. $\theta_d = 0^\circ$. In order to electrically steer the array in a different direction, microphone signals must be pre-aligned before they enter the FIR network. Classically, this can be accomplished by introducing delay lines immediately after the microphones (e.g. \cite{16–18}). However, implementing delay filters for each channel adds a substantial computational burden to the overall deployment, especially for big arrays. Instead, the steering process may directly be integrated into the beamforming-filter design itself by redefining a series of linear constraints in the equation $C^H w = f$ of (8). This idea has already been explored and validated previously in \cite{19}, from which elements of the methodology have been adapted here to facilitate a parallel implementation on a GPU. Therefore, the theory of the linearly constrained filter design, updated from \cite{19}, is developed in the following paragraphs.

Recalling the filter structure of Fig. 3, the output of the beamformer can be alternatively expressed as follows:

$$y(t) = \sum_{m=0}^{N_c N_f - 1} \sum_{i=0}^{J-1} w_{m,i} x_{i}(t - \tau_m(\theta_d, \phi_d) - IT_s), \tag{14}$$

which in the frequency domain translates to

$$Y(f) = \sum_{i=0}^{J-1} \sum_{m=0}^{N_c N_f - 1} w_{m,i} e^{-j2\pi f \tau_m(\theta_d, \phi_d)} X_0(f) e^{-j2\pi f T_s}, \tag{15}$$

where

$$\tau_m(\theta_d, \phi_d) = \frac{\sin \theta_d d_1 \cos \phi_d + n_2 d_2 \sin \phi_d}{c} \tag{16}$$

denotes the cumulative delay from the reference to the $m$th microphone at the desired look direction ($\theta_d, \phi_d$), and $c$ is the speed of sound.

The space-frequency transfer function $H(f, \theta_d, \phi_d)$ is then obtained as

$$H(f, \theta_d, \phi_d) = \frac{Y(f)}{X_0(f)} = \sum_{i=0}^{J-1} w_{i}^T \Delta(f, \theta_d, \phi_d) e^{-j2\pi f T_s}, \tag{17}$$

where

$$\Delta(f, \theta_d, \phi_d) = \left[ e^{-j2\pi f \tau(\theta_d, \phi_d)} \right]. \tag{18}$$

In this case, the response vector $f$ of Eq. (8) becomes

$$f_i = w_i^T \Delta(f, \theta_d, \phi_d).$$

This vector can be thought of as the weighting coefficients of a global FIR filter which integrates the filters of each branch $m$. Consequently, in
order to have a distortionless signal at the beamformer output, the following condition should be satisfied:
\[ H(f) = e^{i2\pi f(J-1)/2}, \forall f_k \in [f_{ld}, f_{hd}], \]
where \( k = \{1, \ldots, K_d\} \), \( f_k \) is a discrete frequency bin, \( f_{ld} \) and \( f_{hd} \) represent the low and high edges of the frequency band of interest, respectively, and \( K_d \) denotes the number of discrete frequencies between \( f_{ld} \) and \( f_{hd} \).

Similarly, in the case where one or multiple directional interferences are present, one may suppress the sound from those directions by imposing a null response
\[ H(f) = e, \forall f_k \in [f_{lv}, f_{hv}], \]
where \( k = \{1, \ldots, K_v\} \), \( e \) is an arbitrarily small number, and \( f_{lv} \), \( f_{hv} \) and \( K_v \) are predefined parameters.

In so doing, the constraints matrix \( \hat{C} \in \mathbb{C}^{N_c \times J} \) and response vector \( F \in \mathbb{C}^{K \times 1} \) can be redefined as follows for \( \Gamma \) interference sources impinging on the array from directions \( (\theta_y, \phi_y) \) \( (\gamma = 1, \ldots, \Gamma) \), with \( K = K_d + \Gamma K_v \ll J \) being the total number of discrete frequencies:
\[ \hat{C} = [C_d C_1 \cdots C_y \cdots C_T], \]
\[ F = [f_d^T f_1^T \cdots f_y^T \cdots f_T^T]^T, \]
where \( C_d \in \mathbb{C}^{N_c \times K_d}, C_y \in \mathbb{C}^{N_c \times \Gamma K_v}, f_d \in \mathbb{C}^{K_d \times 1} \) and \( f_y \in \mathbb{C}^{\Gamma K_v \times 1} \) are defined respectively as
\[ C_d = [c_d(f_1) \cdots c_d(f_{K_d})], \]
\[ C_y = [c_y(f_1) \cdots c_y(f_{K_v})], \]
\[ f_d = [e^{-i2\pi f_1 \gamma_1} \cdots e^{-i2\pi f_{K_d} \gamma_1}]^T, \]
\[ f_y = [e^{i \gamma_1}]^T, \]
\[ c(f_k) = \begin{bmatrix} e^{-i2\pi f_k \gamma_1} \\ \vdots \\ e^{-i2\pi f_k \gamma_{\Gamma K_v}} e^{i(2\pi f_k \gamma_1 (0,0) + J(1-1)T_1)} \end{bmatrix}. \]

The reformulation of the LCMV beamforming design in (8) follows
\[ \min_w \|R_{xy}w^H w \| \text{ subject to } \hat{C}^H w = F. \]

With this scheme, the adaptive algorithm (Eqs. (10)–(11)) does not need to be redefined. Further notice that the computation of the elements in the matrices \( C \) and \( F \) is highly parallelizable, thus making this array-steering scheme computationally efficient and amenable for a GPU implementation.

### 3.2. Complexity Reduction

In the case of the conventional Frost beamformer, the size of the constraints matrix \( C \) is \( N_c N_s J \times J \), whereas in the suggested steering scheme introduced above it has been reduced to \( N_c N_s J \times K \), with \( K \ll J \). Despite this extra gain in computational efficiency, the initialization of the filter coefficients using Eqs. (12)–(13) still requires the computation of a Moore-Penrose pseudoinverse involving the constraint matrix \( C \), which impedes parallelization. To address this issue, the following matrix simplifications are proposed, which lead to equations that totally dispense with the need for any matrix inversions.

Consider the truncated Singular Value Decomposition (SVD) of the constraint matrix \( C \) as follows:
\[ C_K = U_K \Sigma_K V_K^H, \]
where \( U_K \in \mathbb{C}^{N_c \times J}, V_K \in \mathbb{C}^{K \times K} \) are the left and right singular vectors of \( C_K \), respectively, and \( \Sigma_K = \text{diag}(\sigma_k) \) denotes the vector of eigenvalues sorted in descending order, such that \( |\sigma_1| > \cdots > |\sigma_K| \).

Substituting \( C \) for \( C_K \) in Eqs. (12)–(13) and performing matrix simplifications yields the redefinition of the iteration parameters as follows:
\[ g = \hat{U}_K^H F, \]
\[ P = I_{N_c \times N_s} - \hat{U}_K \hat{U}_K^H, \]
where \( \hat{U}_K \in \mathbb{C}^{N_c \times J} \) and \( \hat{U}_K \in \mathbb{C}^{N_c \times J \times N_s N_l} \) are defined respectively as
\[ \hat{U}_K = [\sigma_1^{-1} U_1 \cdots \sigma_K^{-1} U_K], \]
\[ \hat{U}_K = [U_K \theta_{N_c N_s - K}], \]
and \((\cdot)^H\) denotes the Hermitian transpose.

With these expressions, the proposed beamforming scheme no longer requires any explicit matrix inversion and the algorithm can now harness the full parallel processing capabilities of a GPU device.

### 4. IMPLEMENTATION ALGORITHMS

#### 4.1. Notation

Figure 4 shows the overall block diagram of the GPU processes involved in the optical receiver proposed in [12], starting from the image frames acquired by the video camera, and up to the decoding and enhancement of the multichannel audio signals. As stated in Sect. 1, this paper focuses particularly on the blocks involved in the beamforming process (gray boxes), and the following sections will address the algorithms for their parallel implementation. The detailed parallelization architecture of the other blocks related to the image processing and audio signal retrieval are described extensively in [12].

Note that for the sake of coherence and clarity, the same terminology and built-in variable names as those employed by NVIDIA’s CUDA (Compute Unified Device Architecture) programming model will be utilized. Also, for brevity purposes, the following notation will be employed for kernel launch parameters:
\[ \ll (\text{gridDim}.x, \text{gridDim}.y), (\text{blockDim}.x, \text{blockDim}.y) \gg \]
where \text{gridDim}.x and \text{gridDim}.y designate the number of
blocks in the grid, and, \( \text{blockDim.x} \) and \( \text{blockDim.y} \) the number of threads per block in the \( x \) and \( y \) directions, respectively. Similarly, variables written in typewriter font will designate iterators, for instance \( k \) and \( n \) whose ranges are \([0, 1, \ldots, K-1] \) and \([0, 1, \ldots, N-1] \), where \( K \) and \( N \) are the lengths of their corresponding targets.

### 4.2. Data Pre-conditioning

The \( N_1 \times N_2 \) audio channels decoded by the image processing blocks (white boxes of Fig. 4 which are described in [12]), are stored in an \( L \times N_1 N_2 \) matrix \( \tilde{x} \), where \( L \) is the length of the signal, as follows:

\[
\tilde{x} = \begin{pmatrix}
x_0(t = 0) & \cdots & x_{N_1 N_2 - 1}(t = 0) \\
\vdots & \ddots & \vdots \\
x_0(t = L - 1) & \cdots & x_{N_1 N_2 - 1}(t = L - 1)
\end{pmatrix}.
\]  

(34)

In order to perform beamforming on those \( L \times N_1 N_2 \) samples in one-pass, as in Eq. (3), the tapped-delay line (Fig. 3) needs to be filled in an appropriate way before the data can be processed concurrently on the GPU. To accomplish that, the kernel architecture shown in Fig. 5 can be implemented in a grid of \( \ll (N_1 N_2, L), (J, 1) \gg \) 2-dimensional blocks and threads. Each thread executes the following assignments to assemble a matrix \( X \) of size \( N_1 N_2 J \times L \):

\[
tid = n_2 + n_1 N_2 + i N_1 N_2 + 1 N_1 N_2 J
\]

(35)

\[
X[tid] = \begin{cases} 
\tilde{x}[1 + (n_2 + n_1 N_2) L - 1] & 1 \leq i \\
\tilde{x}_c[1 + (n_2 + n_1 N_2) L - 1 + L] & \text{else}
\end{cases}
\]

(36)

where \( \tilde{x}_c \) is a copy of the previously processed block of data. The input data \( X \) is now ready to be processed by the beamforming kernel.

### 4.3. Filter Generation

The filter generation process is split into several steps detailed in the next subsections. A flowchart summarizing the different operations is shown in Fig. 6.

#### 4.3.1. Handling complex data structures

Consider a complex matrix \( A \in \mathbb{C}^{M \times N} \). Its real decomposition \( A_{\mathbb{R}^{2M \times 2N}} \) is obtained as follows:

\[
A_{\mathbb{R}^{2M \times 2N}} = \begin{pmatrix}
\Re(A) & \Im(A) \\
-\Im(A) & \Re(A)
\end{pmatrix}.
\]

(37)

where \( \Re(A) \) and \( \Im(A) \) denote the real and imaginary part of the matrix \( A \), respectively. From this point, the subsequent matrix operations can be carried out with standard floating-point computations. Hereafter, a “\( \mathbb{R}^{2M \times 2N} \)” subscript will systematically be appended to the name of a complex matrix to denote its real decomposed version.

#### 4.3.2. Constraints matrix generation

The constraints matrix \( C \) of Eq. (21) can be generated on the GPU in real-time based on the specifications provided by the user during the application runtime. In this case, the constraints-generation kernel can take as input parameters the user-specified the steering directions of the desired source and also of an interference. The kernel can then be implemented using a grid comprising \( \ll (J, K), (N_1, N_2) \gg \) 2D-blocks whose threads are indexed as

\[
tid_1 = n_2 + n_1 N_2 + i N_1 N_2 + 1 N_1 N_2 J
\]

(38)

\[
tid_2 = tid_1 + N_1 N_2 JK
\]

(39)
\[ f = f_l + \frac{f_h - f_l}{K - 1} \]
\[ \text{expr} = \sin \theta \left( \frac{n_1 \cos \phi + n_2 \sin \phi}{c} \right) + iT_s \]
\[ C_C[tid_1] = \cos(-2\pi f \cdot \text{expr}) \]
\[ C_C[tid_2] = \sin(-2\pi f \cdot \text{expr}) \]

4.3.3. SVD decomposition

Upon computation of the real decomposed matrix \( C_R \), its SVD factors can readily be obtained as in Eq. (29) using the single-precision kernels available in CUBLAS libraries [20].

4.3.4. Response vector generation

The generation of the response vector \( F \) from Eq. (22) is straightforward. It can readily be implemented with the following kernel using a grid of \( K \) threads:
\[ \text{expr} = -\pi f (J - 1) \]
\[ F_C[k] = \cos(\text{expr}) \]
\[ F_C[k + K] = \sin(\text{expr}) \]

4.3.5. Filter coefficients generation

This kernel performs the matrix-matrix product of Eq. (30) using the standard CUBLAS routine for matrix-matrix multiplications (SGEMM). The output of this kernel represents the initial filter coefficients to be used in the beamforming. Note that if the input data is real valued, only the real part of the filter is needed.

4.4. Beamforming Filtering

With the data preconditioning and filter generations schemes presented above, the beamforming filtering process boils down to a straightforward matrix-vector product.

\[ y = X^T w. \]

The parallel architecture that implements the beamforming filtering is shown in Fig. 7. At the core of the kernel is the matrix-vector multiplication implemented with the efficient CUBLAS routine SGEMV. The output of this kernel represents the desired beamforming output \( y(t) \) of the FIR structure in Fig. 3.

5. NUMERICAL EXAMPLE

As an example of the beamforming design introduced above, a numerical case is shown in this section. Our goal is to compute the beamforming response of a linear array formed by 30 microphones attached to a filter structure.
Fig. 8 Angle-frequency response of a beamforming filter design to steer a linear array of 30 microphones at a target source located in the direction $\theta_{\text{tar}} = 80^\circ$ and to suppress an interferer at $\theta_{\text{int}} = 30^\circ$.

(Fig. 7) with $J = 127$ taps. During the design, the following settings are considered: $K_d = K_f = 45$ discrete frequencies linearly spaced between 100 Hz and $F_s = 16,000$ Hz, and inter-microphone distance $d_2 = 3.5$ cm. Furthermore, let us assume that a target source is located at an arbitrary direction $\theta_{\text{tar}} = 80^\circ$, and an interferer at $\theta_{\text{int}} = 30^\circ$, in which a flat and a null response is desired respectively.

Figure 8 illustrates the resulting angle-frequency response computed with the proposed far-field beamforming scheme. Observe that the plot shows a flat constant response computed with the proposed far-field beamforming.

The beamforming algorithms introduced in this paper have been implemented in a computing server with GPU capabilities. The test-bed setup consists of a dual-Xeon CPU computer equipped with an NVIDIA Tesla C2070 GP-GPU card (General Purpose Graphics Processing Unit). To this server, a high speed video camera is attached via an image-acquisition PCI card. The video camera captures and streams intensity images to the server at 16 kilo frames per second (kfps). Furthermore, the optical signals captured by the video camera are transmitted by 120 optical wireless acoustic sensors arranged in a grid of $4 \times 30$, as illustrated in Fig. 9. Each one of these sensor devices is similar to the one described in Fig. 1, which takes samples of the acoustic waves at 16 kHz and broadcasts them in digital form through the LEDs at the same rate. Details of this system are described in [11,12], and a demonstration of its performance can be seen in [21]. This section will focus on showing the performance of the proposed beamforming framework.

6. EXPERIMENTAL INVESTIGATIONS
6.1. System Setup
The beamforming algorithms introduced in this paper have been implemented in a computing server with GPU capabilities. The test-bed setup consists of a dual-Xeon CPU computer equipped with an NVIDIA Tesla C2070 GP-GPU card (General Purpose Graphics Processing Unit). To this server, a high speed video camera is attached via an image-acquisition PCI card. The video camera captures and streams intensity images to the server at 16 kilo frames per second (kfps). Furthermore, the optical signals captured by the video camera are transmitted by 120 optical wireless acoustic sensors arranged in a grid of $4 \times 30$, as illustrated in Fig. 9. Each one of these sensor devices is similar to the one described in Fig. 1, which takes samples of the acoustic waves at 16 kHz and broadcasts them in digital form through the LEDs at the same rate. Details of this system are described in [11,12], and a demonstration of its performance can be seen in [21]. This section will focus on showing the performance of the proposed beamforming framework.

6.2. Beamforming Performance
6.2.1. Beam scanning
The task in this experiment is to steer the sensor array of Fig. 9 through the proposed beamforming scheme so as to scan the half space in front of the array, with the purpose of visualizing the presence of two white-noise acoustic sources located 3 meters away from the array in the directions $A(260^\circ, 85^\circ)$ and $B(20^\circ, 45^\circ)$, where the angle pair $(0^\circ \leq \phi_d \leq 360^\circ, 0^\circ \leq \theta_d \leq 90^\circ)$ are the azimuth and elevation as defined in the coordinate system of Fig. 2. The experiment was setup in a room whose average reverberation time is approximately 213 ms measured based on the $T_{60}$ criterion.

By steering the array with steps of 2 degrees in both directions and computing the output power of the beamforming, it is possible to construct an acoustic map that can reveal the locations of the sound sources. Such acoustic imaging technique has been employed here to generate the acoustic map shown in Fig. 10, in which the two sound sources become clearly visible at approximately the expected directions. Note that in Fig. 10, the following coordinate transformations have been used for the sake of easy visualization:

$$u = \sin \theta_d \cos \phi_d,$$

$$v = \sin \theta_d \sin \phi_d.$$  

As expected, the acoustic map reveals a higher spatial resolution (shaper beam) on the horizontal plane as the dimension of the array on that direction is considerably higher than that in the vertical direction. In order to increase the resolution on the axis $v$, more microphones in the vertical dimension should be added to the array.

6.2.2. Improvement by using positional information
While increasing the size of the array leads to improvements in the output SNR, in practice, building a large array introduces other sources of errors that may degrade the performance of the beamforming. One of them
is the error originated from the mismatch between the idealized array geometry and that of the actual sensor positions. Moreover, due to the additive nature of the beamforming filter network, the mismatching error is linearly cumulative as the number of sensors in the array grows. As an example, Fig. 11 shows the cumulative error derived from the mismatch measurements (Fig. 12) corresponding to an idealized 4 × 30 array and the actual setup of Fig. 9.

Therefore, knowing the exact position of the sensors is as important as the filtering processing itself. Here is where a video camera-based approach comes handy. Within the limits of the pixel resolution, the positional information of the sensors can be accurately retrieved from the recorded images, thus allowing us to boost the performance of the beamforming by correcting the mismatches between the idealized and the experimental arrays.

Continuing with the example of Figs. 11–12, let us discuss our next experiment. We investigate the SNR performance of the beamforming in the case of an idealized geometry model, and when positional information of the sensors is used. For this purpose, the SNR of the signals at one of the input channels is measured and compared to that of the signals at the output, for different array sizes.

The results of this experiment are presented in Fig. 13 from which two observations can be drawn. The first one is the typical behavior of a linear performance gain obtained by doubling the number of sensors in a given direction. In Fig. 13, this is mainly evidenced in the SNR levels corresponding to the linear array of sizes from 1 × 2 to 1 × 30. Adding a few rows of sensors (sizes from 2 × 30 to 4 × 30) does not yield considerable improvements because the test sources are located nearly on the same (horizontal) plane, and the array size is much smaller on the vertical direction. Thus, the response of the beamforming is dominated by the largest dimension of the array. To get a higher SNR for the vertical plane, more sensors in that direction should be added.

On the other hand, adding more sensors introduces positional errors which are reflected as a down-performance in the output SNR (diamond dot line in Fig. 13), leading us to our second observation: the agreement between the cumulative error plot of Fig. 11 and the difference of SNR performance in Fig. 13 before and after correcting the sensor positions with the aid of the video camera.

If the number of microphones was extrapolated (in the order of hundreds or more), one may expect a substantial
improvement when positional information is used. At this point, the advantages of deploying a beamforming array based on an optical-wireless multichannel acquisition approach [11] become apparent.

One should keep in mind that the accuracy of the position estimation depends strongly on the camera resolution. Although techniques to detect LEDs close to imaging-sensor resolution have been introduced [12], this is a topic of ongoing research.

6.3. Real-time Performance Analysis

This section addresses the real-time performance of the proposed beamforming scheme and implementation algorithms.

A common metric to assess the processing power of a system is the real-time factor (RTF), which is defined by the ratio between the processing time $T_{\text{proc}}$ taken to process a data block of duration $T_{\text{data}}$ at a constant rate:

$$\text{RTF} := \frac{T_{\text{proc}}}{T_{\text{data}}}.$$  \hfill (50)

Furthermore, a system is said to achieve real-time performance if its real-time factor is $\text{RTF} \leq 1.0$. Having this definition, let us setup an experiment to estimate the timing profiles for processing data blocks of 256 samples of audio sampled at 16 kHz, i.e. $T_{\text{data}} = 16 \text{ ms}$.

The first scenario is a conventional broadband beamformer executing in a single-threaded CPU implementation. The second is the proposed beamforming architecture deployed in a single GP-GPU card (NVIDIA Tesla C7020). In both scenarios, the data was analyzed for several array sizes, starting from 5 microphones, up to 210.

From the results, the processing time taken to generate the initial filter coefficients, involving all the steps described in Sect. 4.3, can be seen in Fig. 14. For arrays of a few microphones, the time profile of the GPU implementation is dominated by the time of data transfers, thus providing a modest speed gain compared to that of the CPU. However, as the array size grows, it becomes clear that the proposed GPU implementation outperforms its CPU counterpart, thus providing a faster response to the array steering.

Upon computation of the initial filter, the multichannel filtering (Sects. 4.2 and 4.4) starts and continues streaming the output signal. At this point, achieving $\text{RTF} \leq 1.0$ is crucial in order to sustain an overall real-time performance. Figure 15 shows the time profile of the beamforming filtering for both implementations. The horizontal dashed line indicates the limit for achieving real-time processing ($\text{RTF} = 1.0$). Note that the processing time taken by the proposed algorithms is well below that limit, reaching approximately $\text{RTF} = 0.6$ when handling 210 audio channels. In contrast, the processing time taken by the CPU version is over two orders of magnitude compared to that of the GPU framework, showing that a CPU-based realization is inadequate for a massive-sensor array deployment.

Finally, a demonstration of the actual system is available online at [21].

7. CONCLUSIONS

In this paper, parallel signal processing strategies to achieve real-time broadband beamforming with massive microphone arrays were introduced. The algorithms developed here are particularly suitable for optical wireless audio acquisition approaches [11,12]. Such multichannel acquisition systems have been shown promising for the deployment of large acoustic sensor arrays, and therefore, massive parallel signal processing architectures become imperative. In this regard, the experimental results presented here have shown that the proposed beamforming processing framework offers a realizable solution for real-
time scenarios with large acoustic arrays. The experiments have further demonstrated the advantages of exploiting the visual information available from the video-camera in order to improve the beamforming quality. On the other hand, the factors that constraint the scalability are the amount of memory available in current commercial GPU cards, and the limited bandwidth of the PCI interface that interconnects them. Nevertheless, the rapid evolution of the GPU devices, and the emerging GPU-enabled cloud computing technologies, will eventually relax those constraints, thus motivating further development of the fundamental signal processing frameworks proposed in this paper.

REFERENCES

[1] H. L. Van Trees, *Optimum Array Processing: Detection, Estimation, and Modulation Theory* (Wiley-Interscience, New York, 2002).

[2] W. Liu and S. Weiss, *Wideband Beamforming: Concepts and Techniques* (Wiley, West Sussex, UK, 2010).

[3] Panasonic Corporation, “Digital wireless microphone system” (2014), online: http://panasonic.biz/it/ramsa/wirelessmic/dwm/.

[4] Shure Incorporation, “Wireless microphone systems” (2014), online: http://www.shure.com/americas/products/wireless-systems.

[5] H. Yoon and H. Ko, “Microphone array post-filter using input/output ratio of beamformer noise power spectrum,” *Electron. Lett.*, 43, 1003–1005 (2007).

[6] I. A. McCowan and H. Bourlard, “Microphone array post-filter based on field noise coherence,” *IEEE Trans. Speech Audio Process.*, 11, 709–716 (2003).

[7] Y. Hioka, K. Kobayashi, K. Furuya and A. Kataoka, “Enhancement of sound sources located within a particular area using a pair of small micropho arrays,” *IEICE Trans. Fundam.*, E91-A, 561–574 (2008).

[8] R. Monzingo and T. Miller, *Introduction to Adaptive Arrays* (John Wiley & Sons, New York, 1980).

[9] C. Seagrave and E. Benjamin, “Seeing sound: Sound sensor array with optical outputs,” *Proc. AES 114th Convention* (2010).

[10] M. Kurihara, Y. Honji, J. Fujimori, Y. Oikawa and Y. Yamasaki, “Examination of sound field visualization method using small visualization devices built with memsASIC and led,” *Proc. 2nd Meet. Acoustic Imaging, Acoustical Society of Japan* (2009) (in Japanese).

[11] G. Pablo Nava, Y. Kamamoto, S. G. Takashi, Y. Shiraki, N. Harada and T. Moriya, “Simultaneous acquisition of massive numbers of audio channels through optical means,” *Proc. AES 135th Convention*, pp. 247–254 (2010).

[12] G. Pablo Nava, Y. Kamamoto, S. G. Takashi, Y. Shiraki, N. Harada and T. Moriya, “Image processing techniques for high speed camera-based free-field optical communication,” *Proc. 2013 IEEE Int. Conf. Signal and Image Processing Applications (ICSIAP)*, pp. 384–389 (2013).

[13] O. L. Frost, “An algorithm for linearly constrained adaptive array processing,” *Proc. IEEE*, 60, 926–935 (1972).

[14] L. J. Griffiths and C. W. Jim, “An alternative approach to linearly constrained adaptive beamforming,” *Antennas Propag.*, 30, 27–34 (1982).

[15] S. Haykin, *Adaptive Filter Theory* (Prentice Hall, New Jersey, 1996).

[16] L. C. Godara, “Application of the fast Fourier transform to broadband beamforming,” *J. Acoust. Soc. Am.*, 98, 230–240 (1995).

[17] L. C. Godara and M. R. S. Jahromi, “Limitations and capabilities of frequency domain broadband constrained beamforming schemes,” *IEEE Trans. Signal Process.*, 47, 2386–2395 (1999).

[18] L. C. Godara and M. R. S. Jahromi, “Convolution constraints for broadband antenna arrays,” *IEEE Trans. Antennas Propag.*, 55, 3146–3154, (2007).

[19] R. Ebrahimi and S. Seydnejad, “Elimination of pre-steering delays in space-time broadband beamforming using frequency domain constraints,” *IEEE Commun. Lett.*, 17, 769–772 (2013).

[20] NVIDIA CUDA Toolkit, “CUDA Basic Linear Algebra Subprograms” (2014), online: http://docs.nvidia.com/cuda/cublas/

[21] NTT Communication Science Laboratories, “Demonstration of the real-time acquisition of multichannel audio signals through optical means” (2014), online: nms://icsflash.kecl.ntt.co.jp/cslab/ml/pablo/vasdemo1.wmv.

APPENDIX: NEAR-FIELD BEAMFORMING

When the broadband source of interest is close to the microphone array, the far-field assumptions are no longer valid and spherical wavefronts (instead of planar wavefronts) as well as signal attenuation have to be considered. The typical rule of thumb is that the far-field assumptions are no longer valid when the following expression no longer holds true:

\[ R \geq \frac{\max[(N_1 - 1)d_1, (N_2 - 1)d_2]^2F_s}{c}, \]  

where \( R \) is the distance from the signal source to the centre of the microphone array and \( \max[(N_1 - 1)d_1, (N_2 - 1)d_2] \) denotes the maximal spatial extension of the planar microphone array.

In presence of a near-field source, the differences in distances between the source and each sensor can be significant, resulting in phase misalignment across sensors. The difference in propagation time to each microphone with respect to the reference microphone \((n_1, n_2) = (0, 0)\) is given by:

\[ \tau_m(l_{01}, l_{02}) = \frac{\| \tilde{p}_x - \tilde{p}_m \|}{c}, \]  

\[ \tilde{p}_m = R(\sin l_{01}\cos l_{02}, \sin l_{01}\sin l_{02}, \cos l_{02}) \]  

is the position of the source from the reference microphone and \( n = (n_1d_1, n_2d_2, 0) \) is the position of the \( n \)th sensor in the 3-dimensional vector space defined in Fig. 2.

In addition, the wavefront amplitude decays at a rate proportional to the distance traveled. The microphone attenuation factors \( \alpha_m(l_{01}, l_{02}) \), with respect to the amplitude of the reference microphone, are given by:

\[ \alpha_m(l_{01}, l_{02}) = \frac{R}{\| \tilde{p}_x - \tilde{p}_m \|}. \]
A near-field beamformer can then be designed by substituting (16) by (A-2) and by introducing the attenuation factors (A-3) into (27).

**Gabriel Pablo Nava** received his M.S. and Ph.D. degrees from the Department of Information Science and Technology at the University of Tokyo, in 2004 and 2007 respectively. He continued his research work there as postdoctoral fellow for one year. In April 2008, he joined NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation, where he is currently conducting research and development on Room Acoustics, Multichannel Audio, Microphone Array Signal Processing, and Digital Image Processing for the next generation ultrarealistic teleconferencing systems.

**Hoang Duy Nguyen** received his B.S. and M.S degrees in Electrical and Telecommunications Engineering from the University of Brussels, Belgium, in 2011 and 2013, respectively. From 2012 to 2013 he was a researcher in the Smart Systems and Energy Technology (SSET) department of the Interuniversity Micro-Electronis Centre (IMEC), Leuven, Belgium, where he wrote his thesis on the development of high-throughput, near-optimal MIMO detection schemes for parallel programmable architectures. In 2014 he joined NTT Communication Science Laboratories as an intern engineer, and carried out experimental research and development on free-field optical communication. Since 2015 he is working as an Application Support Engineer at The MathWorks Ltd. Cambridge, UK.

**Yusuke Hioka** received his B.E., M.E., and Ph.D. degrees in engineering in 2000, 2002, and 2005 from Keio University, Yokohama, Japan. From 2005 to 2012, he was with the NTT Cyber Space Laboratories (now NTT Media Intelligence Laboratories), Nippon Telegraph and Telephone Corporation (NTT). From 2010 to 2011, he was also a visiting researcher at Victoria University of Wellington, New Zealand. In 2013 he moved to New Zealand and was appointed as a Lecturer at the University of Canterbury, Christchurch. Then in 2014, he joined the Department of Mechanical Engineering, the University of Auckland, Auckland, where he is currently a Senior Lecturer. His research interests include microphone array signal processing and room acoustics. He is a Senior Member of IEICE and a Member of ASJ and IEICE.

**Yutaka Kamamoto** received the B.S. degree in applied physics and physico-informatics from Keio University, Kanagawa, in 2003 and the M.S. and Ph.D. degrees in information physics and computing from the University of Tokyo in 2005 and 2012, respectively. Since joining NTT Communication Science Laboratories in 2005, he has been studying signal processing and information theory, particularly lossless coding of time-domain signals. He additionally joined NTT Network Innovation Laboratories, where he developed the audio-visual codec for ODS (Other Digital Stuff or Online Digital Source) from 2009 to 2011. He has contributed to the standardization of coding schemes for MPEG-4 Audio lossless coding (ALS), ITU-T Recommendation G.711.0 Lossless compression of G.711 pulse code modulation, and 3GPP Enhanced Voice Services (EVS). He is a member of the Information Processing Society of Japan (IPSJ), Acoustical Society of Japan (ASJ), the Institute of Electronics, Information and Communication Engineers (IEICE), and IEEE.

**Takashi G. Sato** received his B.S., M.S. and Ph.D. degrees in Information Science and Technology from the University of Tokyo, in 2003, 2005 and 2008. He is currently a researcher at NTT Communication Science Laboratories. His research interests include bioengineering, tactile and audio interfaces using psychophysiological and neural measurement as feedback information. He is a member of the IEEE and the Society of Instrument and Control Engineers (SICE).

**Yoshifumi Shiraki** received the B.A. degree in natural science from International Christian University in 2008 and the M.S. degree in Computational Intelligence and Systems Science from Tokyo Institution of Technology in 2010, respectively. Since joining NTT Communication Science Laboratories in 2010, he has been studying signal processing, particularly distributed compressed sensing. He is a member of the IEICE and the IEEE.

**Takehiro Moriya** received the B.S., M.S., and Ph.D. degrees in mathematical engineering and instrumentation physics from the University of Tokyo in 1978, 1980, and 1989, respectively. Since joining the Nippon Telegraph and Telephone Public Corporation (now NTT) in 1980, he has been engaged in research on medium to low bitrate speech and audio coding. In 1989, he worked at AT&T Bell Laboratories, NJ, USA, as a visiting researcher. Since 1990, he has contributed to the standardization of coding schemes for the Japanese Public Digital Cellular system, ITU-T G.729, G.711.0, ISO/IEC MPEG MPEG-4 General Audio, MPEG-4 ALS and 3GPP EVS. He is NTT Fellow, IEEE Fellow and a member of IPSJ, IEICE, AES, and ASJ.