Procedia Computer Science

Volume 51, 2015, Pages 201–210

ICCS 2015 International Conference On Computational Science

Real-Time Sound Source Localization on an Embedded GPU Using a Spherical Microphone Array

Jose A. Belloch[1]*, Maximo Cobos[2], Alberto Gonzalez[3], and Enrique S. Quintana-Ortí[1]

[1] Depto. de Ingeniería y Ciencia de Computadores, Universitat Jaume I, 12071–Castellón, Spain
jbelloch@uji.es, quintana@uji.es

[2] Computer Science Department, Universitat de València, 46100–Valencia, Spain
maximo.cobos@uv.es

[3] Depto. de Comunicaciones, Universitat Politècnica de València, 46022–Valencia, Spain
agonzal@dcom.upv.es

Abstract

Spherical microphone arrays are becoming increasingly important in acoustic signal processing systems for their applications in sound field analysis, beamforming, spatial audio, etc. The positioning of target and interfering sound sources is a crucial step in many of the above applications. Therefore, 3D sound source localization is a highly relevant topic in the acoustic signal processing field. However, spherical microphone arrays are usually composed of many microphones and running signal processing localization methods in real time is an important issue. Some works have already shown the potential of Graphic Processing Units (GPUs) for developing high-end real-time signal processing systems. New embedded systems with integrated GPU accelerators providing low power consumption are becoming increasingly relevant. These novel systems play a very important role in the new era of smartphones and tablets, opening further possibilities to the design of high-performance compact processing systems. This paper presents a 3D source localization system using a spherical microphone array fully implemented on an embedded GPU. The real-time capabilities of these platforms are analyzed, providing also a performance analysis of the localization system under different acoustic conditions.

Keywords: Embedded systems, Sound source localization, Audio processing, Microphone arrays

1 Introduction

The localization of sound sources in acoustic environments with noise and reverberation is a challenging task for applications arising in a wide range of areas. Applications such as acoustic-based surveillance, gaming, spatial sound and virtual reality can be highly improved

*Corresponding author. Thanks to projects TIN2011-23283, TEC2012-37945-C02-02, TEC2012-38142-C04-01, PROMETEO FASE II 2014/003, P1-1B2013-20 and EU FEDER

doi:10.1016/j.procs.2015.05.226

Figure 1: Eigenmike microphone, from MH Acoustics.

by using robust algorithms for source localization and tracking [8]. Moreover, other speech-related applications such as speech enhancement or source separation usually require to know the positions of the active sound sources in advance in order to perform successfully. Most localization methods make use of multiple microphones configured under different geometries. As a result, the vast majority of localization approaches need to process in real time the input signals captured by a set of microphones, requiring high computing power. The processing is usually based on the computation of the *Generalized Cross-Correlation* (GCC) [12, 9], which calculates the correlation function by using the inverse Fourier transform of the cross-power spectral density function multiplied by a proper weighting.

Many of the challenges that appear in multi-channel signal processing can be efficiently tackled by taking advantage of the computation capabilities of current Graphics Processing Units (GPUs). The CUDA platform [3] provides a computing framework that enables the use of GPUs in applications beyond computer graphics. For example, in [7] the authors already demonstrated the potential of GPUs for dealing with acoustic source localization problems by using distributed microphones in a room.

In this paper, we address the localization problem by targeting a compact spherical microphone array with 32 capsules using a mobile platform. Spherical microphone arrays have gained a lot of popularity in the last years, leading to commercial arrangements such as the well-known 32-channel *Eigenmike* (shown in Figure 1). However, although spherical microphone arrays are very important for its applications in sound field analysis, beamforming and spatial sound processing, their use requires managing computing platforms powerful enough to process in real time the signals acquired by such a high number of sensors. As a result, exploring and evaluating computationally intensive signal processing algorithms in state-of-the-art platforms becomes a necessary step for envisioning the applicability of these methods in real-time applications.

In the above context, GPUs are gaining widespread within today's embedded world of tablets and smartphones. This is the case of the Tegra K1 (TK1) Mobile Processor, which is a highly parallel data processing unit with low power consumption. The processor TK1 has an embedded GPU that presents a Kepler GPU architecture [3] and is programmed in the same way as those CUDA engines in high-end systems. Nowadays, the processor TK1 can be found on the Jetson development kit (DevKit) [2] and on Google's Nexus 9 tablet [1]. Our purpose in this paper is to evaluate the real-time capabilities of the embedded GPU on Tegra K1 as a portable sound source localization system requiring intensive signal processing, such as is the case for a 32-capsule spherical array. The paper does not only cover the computational analysis of the system, but also presents the adaptation of our former localization work to a spherical geometry. Thus, we provide a performance analysis of the system as well in terms of localization accuracy under different acoustic conditions.

2 Sound Source Localization

In this work, We select the *Steered Response Power - Phase Transform* (SRP-PHAT) algorithm because of its easy parallelization. However, it is necessary to present the adaptation of the algorithm for estimating the Direction of Arrival (DOA) of the source instead of its absolute position. This is due to the fact that localization will not be performed by means of a distributed set of microphones, but from an array of closely spaced sensors and our interest is to estimate the direction of the source with respect to the center of the array.

2.1 The SRP-PHAT Algorithm

Consider the output from a microphone l, $m_l(t)$, in a system composed of S microphones. Then, the SRP at the spatial point $\mathbf{x} = [x, y, z]^T$ for a time frame n of length T is defined as

$$P_n(\mathbf{x}) \equiv \int_{nT}^{(n+1)T} \left| \sum_{l=1}^{S} w_l m_l \left(t - \tau(\mathbf{x}, l)\right) \right|^2 dt, \tag{1}$$

where w_l is a weight and $\tau(\mathbf{x}, l)$ is the direct time of travel from location $\mathbf{x}$ to microphone l. DiBiase [11] showed that the SRP can be computed by summing the GCCs for all possible pairs of the set of microphones. In particular, the GCC for a microphone pair (k, l) is computed as

$$R_{m_k m_l}(\tau) = \int_{-\infty}^{\infty} \Phi_{kl}(\omega) M_k(\omega) M_l^*(\omega) e^{j\omega\tau} d\omega, \tag{2}$$

where τ is the time lag, * denotes complex conjugation, $M_l(\omega)$ is the Fourier transform of the microphone signal $m_l(t)$, and $\Phi_{kl}(\omega)$ is a combined weighting function in the frequency domain. The phase transform (PHAT) [12] has been demonstrated to be a very effective GCC weighting for time delay estimation in reverberant environments:

$$\Phi_{kl}(\omega) \equiv \frac{1}{|M_k(\omega) M_l^*(\omega)|}. \tag{3}$$

Taking into account the symmetries involved in the computation of Eq.(1) and removing some fixed energy terms [11], the part of $P_n(\mathbf{x})$ that changes with $\mathbf{x}$ is isolated as

$$P'_n(\mathbf{x}) = \sum_{k=1}^{S} \sum_{l=k+1}^{S} R_{m_k m_l}\left(\tau_{kl}(\mathbf{x})\right), \tag{4}$$

where $\tau_{kl}(\mathbf{x})$ is the *inter-microphone time-delay function* (IMTDF). This function is very important, since it represents the theoretical direct path delay for the microphone pair (k, l) resulting from a point source located at $\mathbf{x}$. The IMTDF is mathematically expressed as [10]

$$\tau_{kl}(\mathbf{x}) = \frac{\|\mathbf{x} - \mathbf{x}_k\| - \|\mathbf{x} - \mathbf{x}_l\|}{c}, \tag{5}$$

where c is the speed of sound, and $\mathbf{x}_k$ and $\mathbf{x}_l$ are the microphone location vectors.

In summary, the SRP-PHAT algorithm consists in evaluating the functional $P'_n(\mathbf{x})$ on a fine grid G with the aim of finding the point-source location $\mathbf{x}_s$ that provides the maximum value:

$$\hat{\mathbf{x}}_s = \arg\max_{\mathbf{x} \in G} P'_n(\mathbf{x}). \tag{6}$$

2.2 3D-DOA Estimation Using SRP-PHAT

When sound source localization is performed by means of an array of closely spaced microphones, it is usually preferred to locate the source in terms of its relative direction with respect to the array axis instead of its absolution position $\mathbf{x}_s$. To this end, it is assumed that the sound arriving to the microphones is a plane wave of direction given by the DOA vector $\mathbf{d}_s$, which is defined, according to the coordinate system shown in Figure 2, as:

$$\mathbf{d}_s = [\cos\theta_s \cos\phi_s,\ \sin\theta_s \cos\phi_s,\ \sin\phi_s]^T, \tag{7}$$

where θ_s and ϕ_s are the azimuth and elevation angles of the source. Taking into account the above plane wave approximation, the IMTDF of Eq.(5), as a function of a given direction vector $\mathbf{d}$, can be expressed as:

$$\tau_{kl}(\mathbf{d}) = \frac{(\mathbf{x}_k - \mathbf{x}_l)^T \mathbf{d}}{c}. \tag{8}$$

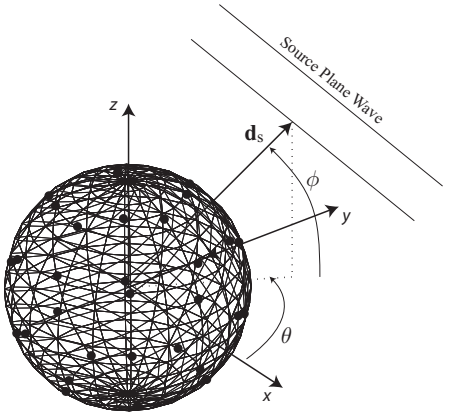

Figure 2: Coordinate system used for the spherical microphone array.

With this approximation, the SRP-PHAT algorithm can be applied over a spherical search grid G to estimate the source direction as the point of the grid accumulating the maximum value:

$$\hat{\mathbf{d}}_s = \arg\max_{\mathbf{d}\in G} P'_n(\mathbf{d}), \tag{9}$$

where

$$P'_n(\mathbf{d}) = \sum_{k=1}^{S} \sum_{l=k+1}^{S} R_{m_k m_l}\left(\tau_{kl}(\mathbf{d})\right), \tag{10}$$

is now evaluated over the direction vectors $\mathbf{d}$ corresponding to each point of the spherical grid.

2.3 Computational Cost

The SRP-PHAT algorithm is usually implemented by performing a frequency-domain processing of the input microphone signals. Given S microphones, the number of microphone pairs to process is $Q = S(S-1)/2$. For a DFT size of L (equal to the time-window size), an FFT takes $5L\log_2 L$ arithmetic operations that result from $\frac{L}{2}\log_2 L$ complex multiplications and $L\log_2 L$

complex additions. Note that one complex multiplication is equivalent to four real multiplications and one real addition, while a complex addition is equivalent to two real additions. In consequence, the signal processing cost for computing the GCC is given by:

- **DFT**: Compute S FFTs, then, $S \times 5L\log_2 L$.
- **Cross-Power Spectrum**: A complex multiplication for L points, resulting in $6L$ operations (4 real multiplications and 2 real additions). This is done for Q microphone pairs, resulting in a cost of $6QL$.
- **Phase Transform**: Magnitude of the L points of the GCC, which costs L operations. This is also done for Q pairs, resulting in QL operations.
- **IDFT**: The IDFT for Q pairs must be performed, which requires $Q5L\log_2 L$ operations.

Moreover, for each functional evaluation, the following parameters must be calculated:

- S Euclidean distances, $\|\mathbf{x}_k\|$, requiring 3 multiplications, 5 additions and 1 square root (≈ 12 operations): $20S$ operations.
- Q TDOAs, requiring 2 operations (1 subtraction and 1 division by c) per microphone pair: $2Q$ operations.
- The SRP requires truncating the TDOA values to the closest sample according to the system sampling frequency, multiplying the cross-power spectrum to obtain the phase transform for each microphone pair and adding up all the GCC values: $5Q$ operations.

As a result, the cost of the SRP-PHAT is given by:

$$Cost = \left(\frac{S+S^2}{2}\right) 5L\log_2 L + \frac{7S(S-1)}{2}L + \nu\left(20S + \frac{7S(S-1)}{2}\right) \text{ flops}, \tag{11}$$

where ν is the total number of functional evaluations. In the conventional full grid-search procedure, ν equals the total number of points of the grid G.

Particularizing the cost for the 32-channel *Eigenmike*, Eq.(11) boils down to:

$$Cost = 2640 \cdot L\log_2 L + 3472 \cdot L + 4112 \cdot N_\theta N_\phi \text{ flops}. \tag{12}$$

3 GPU Implementation

The embedded GPU in TK1 is based on the Kepler GPU architecture [3] and consists of one Streaming Multiprocessor (SMX) composed of 192 CUDA cores. The code to be executed on the GPU concurrently by multiple elementary processes, called threads, is written as kernel function. The threads are grouped into thread blocks. An important aspect to consider is that the threads inside the block can access to data in a coalesced way. Thus, all the samples of all the microphone signals are stored consecutively in memory.

Before starting the real-time processing, it is necessary to compute off-line the inter-microphone delays from Equation (5). To this end, we launch Kernel 1 (see Algorithm 1). Taking into account that the CUDA capability of Tegra K1 is 3.2, the maximum number of resident blocks per multiprocessor is 16. Thus, as Tegra K1 has only one SMX, we execute a kernel composed of 16 thread blocks, where each block is composed of 256 threads, and each thread is devoted to compute $\frac{QN_\theta N_\phi}{16\cdot 256}$ values. Each one of the computed values is stored

in a tridimensional matrix, denoted as **TDM**, of dimensions $(N_\theta \times N_\phi \times Q)$. $\mathbf{TDM}_{t,f,q}$ stands for the component of the matrix at row t, column f, and the q-pair of microphones. Note that $t \in [1, N_\theta]$, $f \in [1, N_\phi]$, and $q \in [1, Q]$, where Q comes from a pair combination $\in \{(1,2),(1,3),\cdots,(32,31)\}$. Figure 3 shows the computation of the matrix **TDM**.

We denote the input-data buffer as $\mathbf{m}_{\text{buff}_k}$, which is composed of L audio samples that have been captured by microphone k. The processing that is carried out with the $S = 32$ captured buffers follows the overlap-save technique with 50% overlap in the frequency domain. Thus, vector $M_{\text{buff}k}$ represents the FFT of vector $\mathbf{m}_{\text{buff}_k}$ and is composed of $2L$ complex elements, where $M_{\text{buff}k,e}$ denotes the complex element of position e inside complex vector $M_{\text{buff}k}$. In addition, the suffixes $.r$ and $.i$ distinguish the real and imaginary parts of a complex element, such as $M_{\text{buff}k,e}.r$. The processing to be carried out by this application in real time is summarized in Algorithm 1, where **GCC** is a matrix with dimensions $(Q \times 2L)$ with complex entries. The value $\mathbf{GCC}_{q,e}$ corresponds to the complex value that is indexed at row q and column e. Matrix **SRP** has the dimensions of the search grid G, i.e $(N_\theta \times N_\phi)$. The value Max_{srp} is the maximum computed value of the matrix **SRP**, and $\mathbf{pos}_{\theta,\phi}$ indicates the position of the Max_{srp}; i. e. the component θ and ϕ of the sound source position.

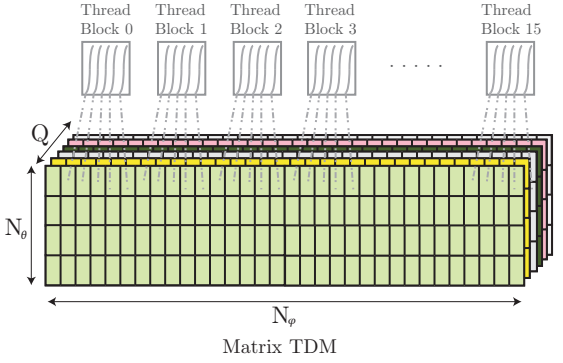

Figure 3: Tridimensional matrix denoted as **TDM** whose dimensions are $(N_\theta \times N_\phi \times Q)$.

Algorithm 1 invokes FFT and IFFT. NVIDIA provides a specialized FFT library [4] that we leveraged in our GPU routine to compute multiple 1-D FFTs and IFFTs. The $FIND$ function retrieves the position in the matrix **SRP** containing the maximum value Max_{srp}.

The **GCC** matrix is computed by means of Kernel 2 (see Algorithm 1). There, a GPU thread takes one value from each of two different rows that are at the same vector position. It conjugates one of the values and multiplies it by the corresponding value of the other row. The phase of the complex number obtained by the multiplication is stored in the corresponding position in the **GCC** matrix.

The accesses to the two rows by GPU threads are totally coalesced since consecutive threads access consecutive memory positions. Kernel 2 is limited by the instruction bandwidth since GPU-native functions `cosf`, `sinf`, and `atan2f` are used and all of them require several clock cycles. Kernel 2 computes $2LQ$ values of the **GCC** matrix by using 256-size thread blocks in a CUDA grid composed of 16 Thread Blocks. This implies launching 4096 threads, where each thread is responsible for computing $\frac{2LQ}{4096}$ values of the **GCC** matrix.

The computation of the matrix **SRP** storing the accumulated SRP values is carried out by kernel 3. This kernel also launches 16 Thread blocks composed of 256 threads. Each GPU thread is devoted to the computation of $\frac{QN_\theta N_\phi}{4096}$ elements of the matrix **SRP**. Note that the

Algorithm 1 Sound Source Localization using spherical microphone array

Input: S, $\mathbf{m}_{\text{buff}}$, **TDM**
Output: $\mathbf{pos}_{\theta,\phi}$

```
/*------NVIDIA CUFFT library------*/
for i = 1,...,S do                                   ▷FFT of each input-data buffer
    M_buffi = FFT(m_buffi).                          ▷S FFTs of size 2L.
end for
/*------Kernel 2------*/
for q = 1,...,Q do          ▷Computation of the GCC matrix for all microphone pairs in Eq.(2)
    l = index first microphone in pair q;
    k = index second microphone in pair q;
    for e = 1,...,2L do                  ▷Conjugate Complex Multiplication of the 2L elements
        cv.r = M_buffk,e.r * M_buffl,e.r + M_buffk,e.i * M_buffl,e.i;
        cv.i = M_buffk,e.r * M_buffl,e.i - M_buffk,e.i * M_buffl,e.r;
        GCC_q,e.r = cos(atan(cv.i, cv.r)).
        GCC_q,e.i = sin(atan(cv.i, cv.r)).
    end for
end for
/*------NVIDIA CUFFT library------*/
for q = 1,...,Q do                                   ▷IFFT of each row of the GCC matrix
    gcc_q = IFFT(GCC_q).                             ▷Q IFFTs of size 2L.
end for
/*------Kernel 3------*/
for t = 1,...,N_θ do
    for f = 1,...,N_φ do
        for q = 1,...,Q do                           ▷For each microphone pair
            τ = TDM_t,f,q.
            SRP_t,f = SRP_t,f + gcc_q,τ.             ▷SRP accumulation as in Equation (10)
        end for
    end for
end for
/*------Kernel 4------*/
Max_srp = MAX(SRP);
pos_θ,φ = FIND(Max_srp, SRP)
```

elements of the matrix **TDM** are required in order to indicate which elements of matrix **GCC** are used to compute the matrix **SRP**. Since the value of the $\mathbf{TDM}_{t,f,q}$ can refer to indicate any position of the column of the **GCC** matrix, coalesced access to the GPU memory is not guaranteed.

Kernel 4 is launched to find the grid position corresponding to the maximum of the computed matrix **SRP** (Max_{srp}). This kernel exactly follows the reduction example in Harris' implementation [5] that comes with the NVIDIA GPU Computing SDK (Software development kit), but it changes the sum operation for a maximum operation.

4 Experiments and Simulations

To analyze both the computing and localization performance of the embedded GPU implementation, we generated a set of acoustic simulations using the image-source method [6]. A shoe-box-shaped room with dimensions 10×10×5 meters was considered, taking different wall

reflection factors [13] $\rho \in \{0.1, 0.5, 0.9\}$. Independent white Gaussian noise was added to each microphone signal to simulate different signal to noise ratio (SNR) conditions, considering the set $SNR \in \{20, 15, 10, 5, 0\}$. For each ρ and SNR condition, a total of 10 simulations with the source positioned at random azimuth and elevation angles were tested. Thus, a total of 150 localization cases where analyzed. The microphone locations of the spherical microphone array are positioned centrally on the facets of a truncated icosahedron of diameter 8.4 cm.

4.1 Computational Performance

The system (Jetson-MicArray) provides L samples per microphone every $\frac{L}{f_s}$ s (with f_s being the sample frequency in Hz). We denote this time t_{buff}. The time employed for the computation is denoted by t_{proc}. The localization system works in real time as long as $t_{\text{proc}} < t_{\text{buff}}$. In case this does not happen, microphone samples are lost and the localization is not accurately performed.

Figure 4 shows the computational performances of the real-time localization system. It presents the frame computation time t_{proc} achieved by different combinations of spatial grid resolutions N_θxN_ϕ with two different L values. Specifically, the tackled angular resolutions are: 20x40, 40x80, 80x160, 160x320 and 320x640. The selected sample frequency is 44.1 kHz and the values of L are 2048 and 4096, which set the time t_{buff} to 46.43 ms and 92.88 ms, respectively. It is important to observe that, as the resolution increases, the matrix **SRP** grows and this boosts the time t_{proc}. In both cases, we are able to locate a sound source in real time by assessing 160 points in the θ direction and 320 points in the ϕ direction.

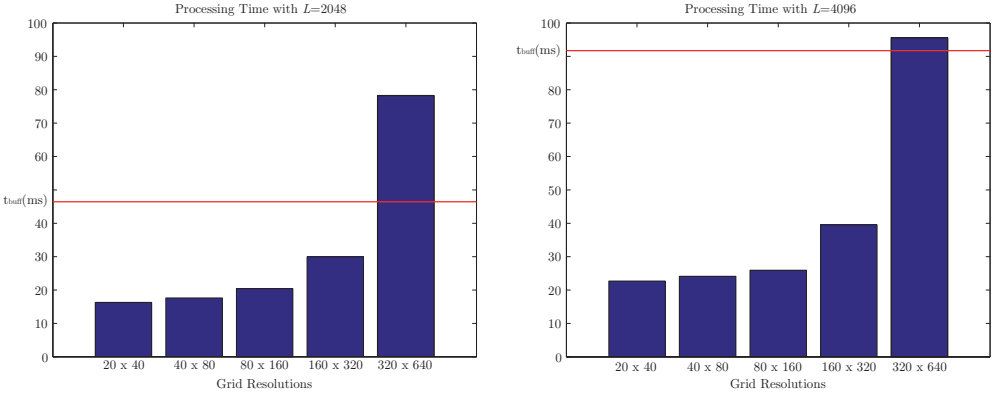

Figure 4: Time t_{proc} for different resolutions.

4.2 Localization Performance

The localization performance in diverse acoustic environments achieved by the spherical array is shown in Figure 5. Simulated acoustic signals of length 10 s corresponding to a male speaker were used to study the directional error by considering the parameters described in Section 4. To enhance the localization capability of the array, a very coarse angular resolution was selected (20x40 points).

The localization accuracy performance of the system is given in terms of the average solid angle error in steradians Ω_e. To understand better this error, the corresponding solid angle is represented in 3D in Figure 6.

As expected, the localization accuracy improves when the reverberation level is low (small ρ) and the SNR increases. Note that, although the performance decreases when the noise is high (low SNR), the main factor affecting the localization performance is the reverberation. As a result, for a given ρ value, the average error remains quite similar. These results confirm that the 32-capsule spherical array is capable of localizing an active sound source with good accuracy even in adverse acoustic conditions and a relatively coarse angular resolution.

Better localization accuracy could be theoretically achieved by using finer spherical grids together with higher sampling frequencies. Further work will be aimed at evaluating the performance of such a computationally demanding system. As already evaluated in Section 4.1, the embedded system is still capable of using finer spherical grids with up to 160x320 points.

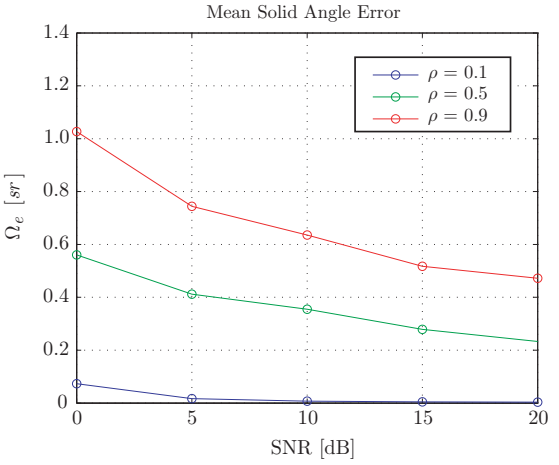

Figure 5: Mean solid angle error for different reflection factors ρ and SNR conditions.

5 Conclusion

Emerging embedded systems with GPU processors allow to develop computationally demanding signal processing applications in real time. The reduced size and low power consumption of embedded platforms are very convenient for building compact processing systems, such as the one presented in this paper. Concretely, we have presented an acoustic source localization system with 32 microphones on a sphere, we proposed a GPU implementation of the well-known SRP-PHAT algorithm, and we assessed its real-time performance. To this end, diverse acoustic environments and the GPU of the embedded system Jetson development kit (DevKit) were considered. This article demonstrates that sound source localization using a compact and portable system can be efficiently performed in real time. The application was tested on the GPU of the TK1 processor, which is found in contemporary mobile devices.

References

[1] Google's nexus 9. `http://blogs.nvidia.com/blog/2014/10/17/nvidia-tegra-k1-google-nexus-9/`. (accessed 2015 January 11).

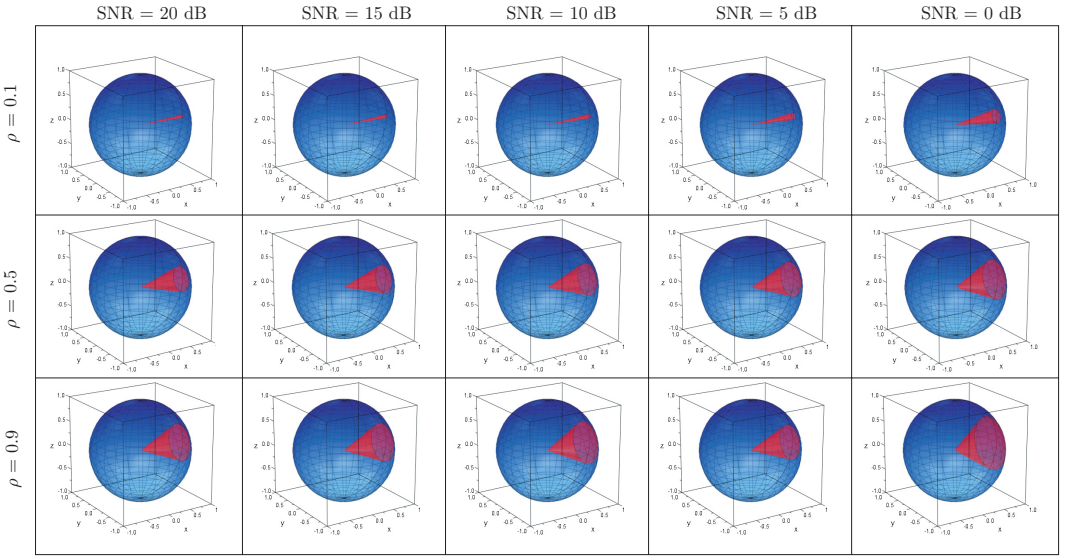

Figure 6: Representation of the mean solid angle error in 3D.

[2] Mobile GPU: Jetson. `http://developer.download.nvidia.com/embedded/jetson/TK1/docs/Jetson_platform_brief_May2014.pdf`. (accessed 2014 November 22).

[3] NVIDIA CUDA Developer Zone. *online at: http://developer.download.nvidia.com/.*

[4] NVIDIA Library CUFFT. *online at: http://developer.download.nvidia.com/compute/DevZone/docs/html/CUDALibraries/doc/CUFFT_Library.pdf.*

[5] Optimizing parallel reduction in CUDA NVIDIA. *online at: http://developer.download.nvidia.com/cuda-downloads.*

[6] J. B. Allen and D. A. Berkley. Image method for efficiently simulating small-room acoustics. *J. Acoust. Soc. Am.*, 65(4):943–950, 1979.

[7] Jose A. Belloch, Alberto Gonzalez, Antonio M. Vidal, and Maximo Cobos. Real-time sound source localization on graphics processing units. *Procedia Computer Science*, 18(0):2549 – 2552, 2013. 2013 International Conference on Computational Science.

[8] M. Brandstein and D. Ward. *Microphone arrays.* Springer, 2001.

[9] J. Chen, J. Benesty, and Y. Huang. Time delay estimation in room acoustic environments: an overview. *EURASIP Journal on Applied Signal Processing*, 2006:1–19, 2006.

[10] M. Cobos, A. Marti, and J. J. Lopez. A modified SRP-PHAT functional for robust real-time sound source localization with scalable spatial sampling. *IEEE Signal Processing Letters*, 18(1):71–74, January 2011.

[11] J. H. DiBiase. *A high accuracy, low-latency technique for talker localization in reverberant environments using microphone arrays.* PhD thesis, Brown University, Providence, RI, May 2000.

[12] C. H. Knapp and G. C. Carter. The generalized correlation method for estimation of time delay. *Transactions on Acoustics, Speech and Signal Processing*, ASSP-24:320–327, 1976.

[13] Heinrich Kuttruff. *Room acoustics.* Taylor & Francis, Abingdon, Oxford, UK, October 2000. 368 pages.