Practical Considerations for Acoustic Source Localization in the IoT Era: Platforms, Energy Efficiency and Performance

Jose A. Belloch[1], José M. Badía[2], Francisco D. Igual[3], and Maximo Cobos[4]

[1]*Depto. de Tecnología Electrónica, Universidad Carlos III de Madrid, Spain.*
[2]*Depto. de Ingeniería y Ciencia de Computadores, Universitat Jaume I de Castelló, Spain.*
[3]*Depto. de Arquitectura de Computadores y Automática, Universidad Complutense de Madrid, Spain.*
[4]*Computer Science Department, Universitat de València, Spain.*

October 2, 2019

Abstract

The rapid development of the Internet of Things (IoT) has posed important changes in the way emerging acoustic signal processing applications are conceived. While traditional acoustic processing applications have been developed taking into account high-throughput computing platforms equipped with expensive multichannel audio interfaces, the IoT paradigm is demanding the use of more flexible and energy-efficient systems. In this context, algorithms for source localization and ranging in wireless acoustic sensor networks can be considered an enabling technology for many IoT-based environments including security, industrial and health-care applications. This paper is aimed at evaluating important aspects dealing with the practical deployment of IoT systems for acoustic source localization. Recent Systems-On-Chip (SoC) composed of low-power multicore processors, combined with a small graphics accelerator (or GPU), yield a notable increment of the computational capacity needed in intensive signal processing algorithms while partially retaining the appealing low power consumption of embedded systems. Different algorithms and implementations over several state-of-the-art platforms are discussed, analyzing important aspects such as the trade-offs between performance, energy efficiency and exploitation of parallelism by taking into account real-time constraints.

Index terms— wireless acoustic sensor networks, source localization, acoustic signal processing, parallel architectures, parallel processing, heterogeneous (hybrid) systems, energy efficiency.

1 Introduction

Sound in general, and speech in particular, constitutes a natural and intuitive way for the development of human-machine interfaces in emerging IoT scenarios [1]. This is one of the reasons why networks of wireless computing devices incorporating microphones (and sometimes loudspeakers), also known as wireless acoustic sensor networks (WASNs), are increasingly attracting the interest of the IoT community [2–5]. In this context, WASNs may be easily integrated into existing living or industrial environments with well-known advantages [6]. First, audio is a cheap and complimentary sensing modality that does not require visual or physical interaction. Second, acoustic sensing devices can be useful in situations when other sensors fail, such as when a sound-emitting target is in the dark or is occluded. Third, voice is still the dominant human communication modality and can be fused with other sensing modalities for an improved overall performance [7].

The use of location information and its potential for the development of ambient intelligence applications has significantly promoted the design of local positioning systems during the last decade [8]. The localization and ranging capability in networks of wireless devices has traditionally been a desirable property since, besides being easily deployable, the nodes can be substantially cheaper with respect to traditional sensing architectures. While most wireless sensor networks (WSNs) have been typically using the received signal strength (RSS) or the time of arrival (TOA) of radio signals, the use of sound in WASNs brings a set of benefits [9, 10]. For example, since the localization accuracy depends on both the signal propagation speed and the precision of the temporal measurements, acoustic signals may be preferred over radio signals for their lower propagation speed. Additionally, in typical IoT-based applications for ambient intelligence, such as in ambient assisted living (AAL), the system already relies on application-dependent sensors such as cameras and microphones [5]. In this context, the location of the user is a valuable piece of information, since knowing the position of the user enables the implementation of services that may make the living environment easier, safer or more comfortable.

From the hardware perspective, mobile platforms for acoustic sensing are made up of two key elements: low-power processors and acoustic sensors. Audio capturing and processing is considered a challenging matter, which involves a trade-off between the complexity of the audio processing task and the hardware resources. The audio monitoring process introduces some specific requirements on hardware platforms. On the one side, audio signals are normally sampled at relatively high rates, demanding large memories and high computational capabilities. On the other side, signal processing tasks should be programmed carefully to deal with the audio sampling process and to optimize the system resources properly.

A popular approach for sound source localization is the well-known Steered Response Power (SRP) with Phase Transform (PHAT) algorithm [11, 12]. This method is based on a grid-search procedure where the output power of a filter-and-sum beamformer is computed through a grid of candidate source locations.

The power map resulting from the values computed at all these locations (also known as Global Coherence Field) will show a peak at the estimated source position. Since the nodes of a WASN are not usually synchronized, the usual practice is to compute a power map at each node of the network using multiple microphones. Then, the maps are all combined together in a meaningful way before estimating the final source location. As a result, each node of the network must perform intensive signal processing operations, this being an important issue from a resource management perspective. Fortunately, the SRP-PHAT method exhibits a massive fine-grain parallelism, with the same operations performed over many sets of data [13, 14]. Usually, these data sets correspond to the audio samples of the different audio channels involved in the system. Moreover, source localization applications may involve different needs in terms of the number of microphones and spatial resolution. In this context, the computational complexity of the system may be affected by the total number of candidate locations explored by the algorithm [which may depend on the size of the localization space or the desired spatial resolution] and the number of microphones.

From the above considerations, it becomes clear that practical IoT systems incorporating sound source localization features must be scalable and computationally efficient, posing important challenges for the platforms selected for such edge computing tasks. The use of system-on-chips (SoC) within IoT systems is becoming widespread [15, 16]. The emergence of SoC composed of multi-core processors built either from multicore CPUs or even a small graphics accelerator (or GPU) contributes a notable increment of the computational capacity while partially retaining the appealing low-power consumption of embedded systems.

A primary goal of this paper is to provide insight concerning the implementation of SRP-based localization algorithms in IoT-oriented platforms, taking into account the main challenges arising in terms of real-time performance, energy consumption and exploitation of parallelism. Obviously, the complete fulfillment of all design constraints will not be straightforward. For example, high-accuracy localization involves increasing the computational cost, which makes it more difficult to achieve real-time performance. Likewise, using more resources to increase the speed of the algorithms comes at the expense of larger energy consumption. Therefore, implementations designed to leverage the different components and capabilities of the device are a must. In this context, the paper considers different techniques aimed at reducing the cost and energy consumption of the algorithm, while providing sufficient speed to achieve real-time performance. These include the reduction of the frequency of different components of the platform, the pre-computation of some values, the use of the fastest memories of the GPU or arranging the data to get a coalesced access, among others.

We focus on the evaluation and the practical requirements for performing sound source localization over three state-of-the-art multi-core-based SoCs: (1) The ODROID XU3; (2) the Jetson TX1; and (3) the Raspberry Pi 3. We discuss different implementations and analyze the trade-offs between performance and energy efficiency for different distributions of the computational load on

the three proposed SoCs. We use well established programming tools, such as OpenMP, OpenCL and CUDA to obtain portable implementations that can leverage the parallel capabilities of the CPU and GPU cores of a wide range of SoCs. Besides, we analyze the effect of modifying the frequencies of the cores or even disabling some of them on the time and energy consumption of the algorithm. As a result of this analysis, we establish the practical limitations of source localization systems relying on platforms of this kind, taking into account considerations such as the size of the system, the final spatial resolution, the real-time performance capabilities and the resulting energy consumption. In addition, we compare results with another recently proposed algorithm specifically designed to reduce the computational cost of SRP-based localization.

The rest of the paper is structured as follows. Section II reviews IoT application scenarios where sound source localization is useful and describes the fundamentals of the two SRP-based algorithms considered in this paper: the conventional SRP-PHAT method (C-SRP) and the refined volumetric SRP method (RV-SRP). Section III summarizes the main features of the SoCs employed in this work, describing specific implementation issues in Section IV. A thorough performance evaluation is conducted in Section V. Finally, Section VI provides a few concluding remarks.

2 Acoustic Source Localization in IoT

2.1 Sound source Localization in IoT scenarios

Sound source localization has already been applied to different scenarios within the paradigm of IoT, such as automatic surveillance, environmental monitoring, elderly care, smart homes or industrial environments. In most of these environments, energy efficiency is required due to the fact that an electricity connection is not always available [17].

For pervasive IoT acoustic surveillance, it is necessary to detect and localize abnormal acoustic events in a distributed collaborative manner [18]. In fact, we can find in the literature research works aimed at preventing hazardous situations such as might be indicated by the sound of human screams [19,20] or other kinds of sounds such as gunshots, explosions, machine sounds or children voices, among others [21].

In the industry, there also exists a wide range of applications that require to perform acoustic source localization [22]. Currently, smart factories making use of distributed sensors are gaining momentum. In this context, source localization allows the detection of machine break-downs such as in [23]. In [24], the authors propose a system that can detect the acoustic signature of power tools, and the effectiveness of the system being used as an early warning system to detect misuse of machinery is demonstrated.

Smart farms can also benefit from acoustic-based IoT systems. As an example, in [25] the authors implement sound source localization in farms in order to detect sick animals in commercial piggeries, since sick animals emit charac-

teristically unusual sounds.

Another interesting IoT scenario is that related to ambient assisted living [5]. Systems oriented to the monitoring of homes where elderly or disabled people live have been receiving a lot of attention in recent years. The first approaches designed to detect events of this kind were carried out in [26,27] where emergency falling detection systems were considered. The authors of [28] incorporate other high performance computing features to acoustic sensor networks by designing a real-time audio event detection for surveillance remote monitoring.
It is important to highlight that such systems do not only require powerful embedded systems, but they also should be designed to make an efficient usage of energy resources. In applications of this kind, edge computing may play a key role, since it makes it possible to reduce the response time when an emergency occurs: the most important part of the computation is carried out on the device, so there is not need to send the information to a central node and wait to a response as it happens in a cloud-computing environment [29]. Thus, edge computing reduces the response time by limiting communication between nodes and makes it possible to develop applications as the ones considered above. In this paper we analyze the most important aspects that need to be considered in a computational-demanding IoT scenario centered on acoustic source localization, seeking for maximum computational performance, real-time and energy-efficient solutions.

2.2 Source localization networks

A typical WASN for sound source localization assumes a moving acoustic source and a collection of fixed anchor nodes placed at known (or unknown) positions. In common IoT applications, the source usually consists of an unknown speech source or an acoustic event. Since the source and the nodes are not synchronized, the use of time-of-arrival (TOA) information is rarely employed, motivating the use of time-differences of arrival (TDOAs) between synchronized sensors at each node. Therefore, typical WASN nodes incorporate a set of synchronized microphones following a particular geometry from which TDOA estimates can be obtained. Note, however, that the microphone signals from different nodes may not be synchronized.

A popular approach for inferring the location of the sound source relies on the computation of Steered-Response Power (SRP) maps [30]. In a typical setup, each node of the WASN would compute an SRP power map, sending it to a sink node that merges the spatial likelihood information gathered by each node. In order to minimize signal transmissions and allow for an increased battery life in the nodes, sending the signals captured by the microphones to a central node for computing SRP power maps is generally avoided. As a result, the nodes are in charge of performing the required signal processing, sending only the power maps once these have been computed. However, an important aspect of SRP-based localization is that algorithms are relatively expensive from a computational point of view and, thus, powerful edge platforms capable of managing costly signal processing operations over multiple microphone channels

become necessary.

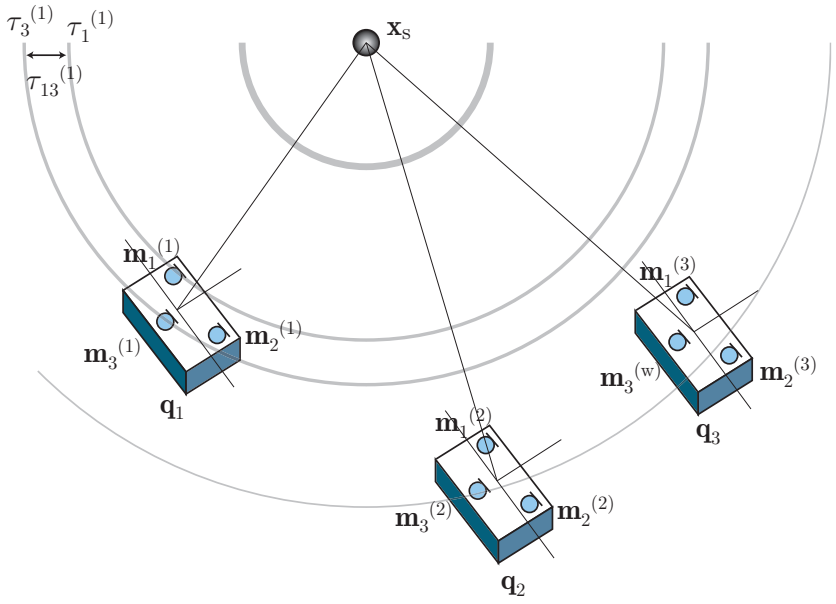

Figure 1: WASN with three nodes and three microphones per node.

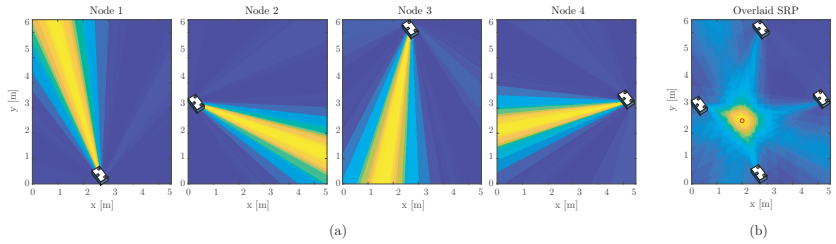

Figure 2: An examle of SRP-based localization using four nodes. (a) SRP power maps computed by each node. (b) Combined SRP matrix.

Figure 1 shows a general WASN with a set of wireless nodes and an emitting sound source. It is assumed that the network consists of M nodes and that each node incorporates S microphones. In the example shown in Figure 1, $M = 3$ and $S = 3$. The nodes are assumed to be located at positions $\mathbf{q}_m = [q_{x,m},\, q_{y,m},\, q_{z,m}]^T$, $m = 1, \ldots, M$, while the microphone locations are denoted as $\mathbf{m}_l^{(m)} = [x_l^{(m)},\, y_l^{(m)},\, z_l^{(m)}]^T$, $l = 1, \ldots, S$, where the superscript $^{(m)}$ identifies the node at which the microphone is located. The source position is denoted as $\mathbf{x}_s = [x_s,\, y_s,\, z_s]^T$, while a general point in space is $\mathbf{x} = [x,\, y,\, z]^T$. Note that all

these location vectors are referenced to the same absolute coordinate system. The time instant at which the source signal arrives to a given microphone, i.e. the TOA, is denoted as $\tau_l^{(m)}$. TDOAs are denoted by $\tau_{kl}^{(m)}$, and correspond to the observed TOA differences between pairs of microphones (kl). For the sake of clarity in the notation, throughout the rest of the paper we will consider pairs of microphones within the same node, so we will omit the superscript $^{(m)}$ of the sensor.

2.3 Conventional SRP-PHAT algorithm (C-SRP)

Consider the output from a microphone l, $m_l(t)$, in a system composed of S microphones. The power of a beamformer steered to a spatial location $\mathbf{x} = [x, y, z]^T$ can be calculated in terms of the generalized cross-correlation (GCC) of the different microphone pairs of the system. Given the pair of microphones k and l, with $k \neq l$, the GCC between $m_l(t)$ and $m_k(t)$ can be written as [31]

$$R_{kl}(\tau) \triangleq \frac{1}{2\pi} \int_{-\pi}^{\pi} M_k(\omega) M_l^*(\omega) \Psi_{ij}(\omega) e^{j\omega\tau} d\omega, \tag{1}$$

where $M_l(\omega)$ is the DTFT of $m_l(t)$, * is the conjugate operator and $\Psi_{ij}(\omega)$ is a suitable weighting function. One of the most common choices is to use the PHAse Transform (PHAT) weighting function, i.e.

$$\Psi_{ij}(\omega) = \frac{1}{|M_k(\omega) M_l^*(\omega)|}.$$

DiBiase [11] demonstrated that the SRP at a spatial location $\mathbf{x} \in \mathbb{R}^3$ calculated over a time interval of T samples can be efficiently computed in terms of GCCs:

$$J_{\mathrm{C}}^{(\mathrm{SRP})}(\mathbf{x}) = \frac{2}{T} \sum_{k=1}^{S} \sum_{l=k+1}^{S} R_{kl}\left(\tau_{kl}(\mathbf{x})\right) + \sum_{k=1}^{S} R_{kk}(0), \tag{2}$$

where $\tau_{kl}(\mathbf{x})$ is the time difference of arrival (TDOA) that would produce a sound source located at $\mathbf{x}$ with a propagation speed c, i.e.

$$\tau_{kl}(\mathbf{x}) = \frac{\|\mathbf{x} - \mathbf{m}_k\| - \|\mathbf{x} - \mathbf{m}_l\|}{c}. \tag{3}$$

The last summation term in Eq. (2) is usually ignored, since it is a power offset independent of the steering location. When GCCs are computed with PHAT, the resulting SRP is known as SRP-PHAT. In practice, the method is implemented by discretizing the location space region $\mathbb{V}$ using a search grid $\mathcal{G}$ consisting of candidate source locations in $\mathbb{V}$ and computing the functional of Eq.(2) at each grid position. The estimated source location is the one providing the maximum functional value:

$$\hat{\mathbf{x}}_s^{(\text{C-SRP})} = \arg\max_{\mathbf{x} \in \mathcal{G}} J_{\mathrm{C}}^{(\mathrm{SRP})}(\mathbf{x}). \tag{4}$$

Figure 2 shows a 2D example of a localization system in a room of dimension 5×6 m, with four nodes located at the mid-points of the walls. Each node incorporates 3 microphones. The power maps computed by each node are represented in (a). It can be clearly observed that the power concentrates in some directions. When all the power maps are combined in (b), the overlaid SRP map shows a peak on the true source location (red circle).

2.4 Refined Volumetric SRP (RV-SRP)

The conventional SRP-PHAT algorithm (C-SRP) previously described requires dense grids for achieving high-accuracy location estimates. This may be prohibitive in real-time applications with many microphones. To deal with this issue, the modified SRP method was first proposed as a robust alternative considering the volume surrounding each point of the search grid, which enables the use of coarser grids and, consequently, reduces considerably the computational cost [32]. Following a similar rationale, the refined volumetric SRP (RV-SRP) method has been recently proposed as an alternative for achieving high-accuracy location estimates with reduced cost, which defines a two-step procedure [33]. First, the entire search space is reduced to a volume $\hat{\mathcal{V}}$, chosen as the volume that maximizes the objective function

$$J_{RV}^{(\mathrm{SRP})}(\mathcal{V}) = \sum_{k=1}^{S} \sum_{l=k+1}^{S} \sum_{\tau=\tau_{kl,\mathcal{V}}^{\min}}^{\tau_{kl,\mathcal{V}}^{\max}} \mathcal{X}_{kl}(\tau, \mathcal{V}) R_{kl}(\tau)), \tag{5}$$

where $\mathcal{X}_{kl} = 1$ when there exists at least one point of the grid $\mathbf{x} \in \mathcal{V}$ such that its associated TDOA $\tau_{kl}(\mathbf{x})$ is equal to τ, and the limits $\tau_{kl,\mathcal{V}}^{\min}$, $\tau_{kl,\mathcal{V}}^{\max}$ are given by the minimum and maximum TDOA within a volume $\mathcal{V}$. Otherwise, $\mathcal{X}_{kl}$ is zero. In the second step, the C-SRP is applied inside the new search space $\hat{\mathcal{V}}$. Note that the RV-SRP method needs to be initialized by considering a fine grid from which lag limits are pre-computed, although a functional is only computed in the second step over those points contained within the winning volume.

2.5 Sequential implementation

The C-SRP and RV-SRP may be implemented following a sequential approach. In fact, the second step of RV-SRP is equivalent to a C-SRP evaluated over a reduced search space.

The SRP-PHAT algorithm takes as input sample buffers of size L corresponding to S microphones. The main steps carried out by the algorithm are the following:

1. For each of the S microphones weight the L samples by a Hamming window vector.

2. For each of the S microphones perform an L-point FFT resulting in S vectors, each containing L frequency bins.

3. Compute the GCC matrix of size $Q \times L$, being Q the number of microphone pairs.

4. For each of the Q rows of GCC compute an inverse $L -$ FFT.

5. Compute a three-dimensional SRP matrix. Each element of the matrix corresponds to a point of a 3D spatial grid and its value depends on the TDOA resulting from the grid point to one pair of microphones.

6. Obtain the position of the maximum SRP value corresponding to the estimated sound source location.

A more detailed version of the algorithm, including an analysis of the cost of each step can be found in [14].

3 SoCs for IoT

We evaluate the SRP-PHAT algorithm on three different SoCs that are representative of the state-of-the-art in heterogeneous low-power architectures. All of them implement the ARM Cortex-A microarchitecture, with support of the ARMv7 (32-bit) and ARMv8 (64-bit) Instruction Set Architectures (ISAs). Although all of them are considered to be low power architectures, they substantially differ in terms of heterogeneity, use of accelerators or amount of memory. Thus, this selection illustrates a range of solutions available in the market and can be applied to different scenarios depending on the specific computation or power consumption requirements of the target application.

3.1 ODROID XU3

The ODROID XU3 is a board that integrates the Samsung Exynos 5422 SoC based on an octa-core ARM Cortex CPU featuring a big.LITTLE paradigm. The SoC integrates four fast ARM Cortex A15 and four energy-efficient ARM Cortex A7 in the same die, together with an ARM Mali-T628 MP6 GPU. The board includes 2 Gbytes LPDDR3.

3.2 Nvidia Jetson TX1

The Nvidia Jetson TX1 is a board that features a SoC based on a quad-core ARM Cortex-A57 CPU with 4 Gbytes LPDDR4 and a high-performance 256-core Nvidia Maxwell GPU. Designed with computer vision and deep learning in mind, it exhibits a plethora of connectivity interfaces, including 802.11ac WiFi, Bluetooth 4.0, Gigabit Ethernet and PCIe Gen2. The board draws up to 15 Watts TDP when the processing module is fully used. It includes 4 Gbytes of LPDDR4.

3.3 Raspberry Pi 3

The Raspberry Pi 3 is the latest release of the Raspberry Pi series. The board integrates a Broadcom BCM2837 SoC with a quad-core ARM Cortex A53 CPU and a small Broadcom VideoCore IV GPU. The system is equipped with 1 Gbyte LPDDR2 memory and wired and wireless interconnection interfaces.

3.4 Power monitoring infrastructure

The energy measurements in this paper require an environment that yields detailed and reproducible values. In order to obtain comparable energy measurements across all boards, we decided to measure the power consumption of the complete boards (that is, measuring at the DC entrance). While the Jetson TX1 and the ODROID XU3 include an isolated energy counter for the SoC, the lack of this mechanism in the Raspberry Pi 3 invalidates their usage if a fair comparison is desired. In our case, we have adapted the `pmlib` framework [34] to interact with the Smart Power device [35] in order to monitor the instantaneous power draw for the complete execution. For the Jetson TX1, we measure the same value leveraging internal power sensors. In all cases, we report *application* energy consumption numbers by subtracting the observed idle power of the corresponding board to that observed during the application execution.

4 Implementation Issues

Each of the six main steps of the the C-SRP algorithm allows us to exploit data parallelism. Specifically, in step 1) every sample can be weighted in parallel, while in steps 3) and 5) every element of matrices GCC and SRP respectively, can be computed in parallel. Regarding steps 2) and 4), there exist several efficient parallel implementations of the FFT included in well-know libraries. Finally, the computation of the maximum in step 5) can be performed by means of a parallel reduction schema.

We have used different programming technologies to leverage the resources of the parallel architectures trying to obtain the best performances both in terms of time and energy consumption. We have chosen well-known tools that allow us to obtain portable algorithms that can be executed in a wide range of parallel architectures, from low-power SoC platforms to high performance parallel computers. Specifically we have implemented a parallel OpenMP version of the algorithm to run on the CPU cores [14]. It is quite straightforward to use parallel `pragmas` to parallelize some of the loops of each of the steps of the algorithm or to use the `reduction` clause to obtain the position of the maximum SRP value. We have compared different loop scheduling strategies and chosen the best in each case. Steps 2) and 4) have been parallelized using the multi-threading capabilities of the FFTW library [36].

Some SoC platforms include a GPU and provide the possibility of using CUDA, OpenCL or both technologies to run the SRP-PHAT algorithm. There-

fore, we have also implemented parallel versions of the algorithm that use both programming tools [13, 14, 37].

Algorithm 1 summarizes the main steps of the parallel implementation using OpenCL. The version using CUDA is very similar. Input parameter B is a vector containing the S input sample buffers of size L, vector H contains the Hamming window and p the position of the microphones. We have implemented five kernels to solve steps 1), 3), 5) and 6) of the sequential version. The two calls to `clFFT` functions in lines 4 and 6 correspond to the routine included in the OpenCL FFT library [38]. In the CUDA version we use the routine included in the library [39]. The algorithm minimizes the transfer of information between the host CPU and the GPU device. It involves an initial transfer of some input vectors from host to GPU and the final transfer of the sound source position $pmax$ from the GPU to the host. All the kernels produce and reuse intermediate results in the global memory of the device and leverage its local and private memories when possible.

Algorithm 1 Parallel OpenCL SRP-PHAT algorithm.

```
1:  function SRP-PHAT(B, H, p, L, S)
2:      Transfer CPU → GPU: B, H, p
3:      RB = kHamming(B, H, L, S)
4:      RFB ← clFFT(RB, "forward")
5:      GCC = kGCC(RFB, L, S)
6:      RGCC = clFFT(GCC, "backward")
7:      SRP = kSRP(RGCC, p, L, S)
8:      max = kRedMax(SRP)
9:      pmax = kPosMax(SRP, max)
10:     Transfer CPU ← GPU: pmax
11:     return pmax
12: end function
```

The implementation of the kernels executed by each work-item to approach steps 1), 3) and 5) of the sequential algorithm are summarized in algorithms 2, 3 and 4 respectively. Kernel `kRedMax` uses a multi-step parallel tree reduction schema, such as the one describe in [40], to obtain the maximum value of matrix SRP. Finally, in kernel `kPosMax` the work-item containing that maximum value returns its position.

We have implemented different versions of the kernels trying to reduce the memory access cost and reuse the data on each work-item. For example, we have analyzed the effect of precomputing the distances from every grid point to every microphone or to every pair of microphones instead of computing them on each iteration of the algorithm. For example, those distances are computed in lines 8 and 11 in the version of the kernel included in algorithm 4. However, our experimental results show that it is faster to precompute all the inter-microphone time delays (see line 12) only once at the beginning of the localization process.

The main drawback of the previous strategy is its spatial cost, because we need to store an integer distance from every point of the grid to every pair of

microphones. If we increase the spatial resolution or the number of microphones those values might not fit into the memory of the CPU or GPU, even more when dealing with low power devices with small memories. A possible solution that could alleviate that problem is to precompute only the distances from every point to every microphone and use them to compute the interdistances during the localization process.

In order to reduce the memory access cost we have arranged the data so that the work-items perform, when possible, a coalesced access to the matrices and vectors. For example in kernel `kHamming` (algorithm 2) work-item i reads the $i-th$ sample of each microphone. As matrix B stores consecutively those samples, we achieve a coalesced access to the global memory of the GPU. Notice also that every work-item uses one element of the Hamming window and in line 3 of the algorithm we copy it from the global memory to much faster private memory, which usually involves registers of the GPU.

We have also implemented versions of some of the kernels where each work-item performs computations with different granularities. For example, in kernel `kHamming` each work-item could compute one of the elements of matrix GCC. However, it is faster to increase the granularity by computing all the elements of one row on one work-item, as it is shown in algorithm 2.

Finally, another aspect that needs to be taken into account in order to optimize the CUDA and OpenCL codes and adapt them to every device is the size of the thread blocks that execute each kernel. The experimental results shown in this paper are always obtained with the thread block sizes that give the minimum time on the corresponding GPU device.

Algorithm 2 Hamming window application kernel.

```
1: function kHamming(B, H, L, S)
2:     i = global index of the work-item
3:     pH = H[i] // copy to private memory
4:     for f ← 1 to S do // for each microphone
5:         RB[i] = ( B[i] * pH, 0.0 )
6:         i += L
7:     end for
8:     return RB
9: end function
```

5 Experimental platforms and testbed

We have performed our experiments in the three well-known low-power platforms described in Section 3.

All the experiments have been conducted using a varying number of microphones (from 6 to 24) over synthetic recordings simulated by means of the image-source method. In the following subsections we will mostly show results using 12 microphones because all the parallel algorithms allow us to perform the

localization in real time on the three experimental platforms using that number of microphones. The algorithms have been tested with sample buffers of size $L = 4096$ for each microphone. For a sample frequency $f_s = 44.1$ KHz, if we want to locate the source in real time, the processing time of the algorithm must be less than $t_p = 92.88$ ms.

Algorithm 3 GCC matrix computation kernel.

```
1: function kGCC(RFB, L, S)
2:     i = global index of the work-item
3:     pa = 0 // pair index
4:     for m1 ← 1 to S do
5:         for m2 ← m1+1 to S do
6:             // Complex conjugate product
7:             c = RFB[m1 * L + i] * RFB[m2 * L + i]
8:             angle = atan2(c)
9:             GCC[pa * L + i] = ( cos(angle), sin(angle) )
10:            pa++
11:        end for
12:    end for
13:    return GCC
14: end function
```

Algorithm 4 SRP matrix computation kernel.

```
1: function kSRP(RGCC, p, L, S)
2:     (i, j, k) = global 3D index of the work-item
3:     x = position of the grid point (i, j, k)
4:     pa = 0 // pair index
5:     r = 0.0
6:     for m1 ← 1 to S do
7:         x1 = p[m1]
8:         d1 = ||x − x1||
9:         for m2 ← m1+1 to S do
10:            x2 = p[m2]
11:            d2 = ||x − x2||
12:            delay = (d1 − d2)/c // integer division
13:            r += RGCC[pa * L + delay]
14:            pa++
15:        end for
16:    end for
17:    SRP[i, j, k] = r
18:    return SRP
19: end function
```

5.1 C-SRP vs. RV-SRP

Firstly we compare two strategies aimed at reducing the temporal cost of SRP-PHAT localization. On the one hand, we use the low-complexity RV-SRP algorithm described in Section 2.4 On the other hand, we implement several high-performance parallel versions of the C-SRP algorithm. The objective of this comparison is to evaluate the gain obtained from a proper resource management scheme versus the one derived from a low-complexity algorithmic approach. Specifically, Figure 3 compares the results obtained using 6 an 12 microphones on a Jetson TX1 platform. The OpenMP results are obtained leveraging the four cores of the CPU, while the CUDA version is executed on the 256 cores of the Maxwell GPU. The RV-SRP version uses initially a coarse grain volumetric grid with a resolution of 0.1 m and refines the results on a smaller volume with a finer resolution of 0.02 m The sequential and parallel versions of the C-SRP algorithm use the same fine grain resolution, but on all the search space. If we are dealing with a sequential platform, the RV-SRP algorithm is an excellent alternative to the C-SRP, as it clearly reduces its execution time. This reduction is larger as we increase the number of microphones and so the cost of computing the SRP value on every grid point. However, the parallel versions of the C-SRP outperform the lower complexity sequential RV-SRP algorithm. For example, using 12 microphones the CUDA implementation is around 16 times faster than the sequential implementation of the same algorithm and around 10 times faster than the RV-SRP algorithm.

Regarding the energy consumption of the different algorithms, the RV-SRP version is also more efficient than the conventional version (see Figure 4). However, the most energy efficient implementation is again the one implemented using CUDA and leveraging the GPU of the platform.

5.2 Results with the Odroid-XU3

In the following sections we will analyze the behaviour of the sequential and parallel versions of the SRP-PHAT algorithm, both in terms of time and energy, in the three experimental platforms.

Figure 5 shows that the sequential version of the algorithm can only perform the localization in real time with low resolution grids ($r >= 0.1$ m) and using the fastest kind of core (Cortex-A15). However the parallel versions of the algorithm allow us to locate the source with medium resolution grids. The best results are obtained with the OpenMP version using the fastest core, but the OpenCL version obtains quite similar results as we increase the resolution.

The following three figures allow us to analyze the effect of modifying the frequencies of some of the components of the architecture on the energy consumption of the parallel algorithms. We are always using a grid resolution of $r = 0.1$ m. Figure 6 shows the energy consumed by the platform in mJ when we execute the OpenMP version of the algorithm on its two types of CPU cores. The power dissipated by both kinds of cores slowly increases with their frequency and the number of cores. However, in terms of energy consumption the

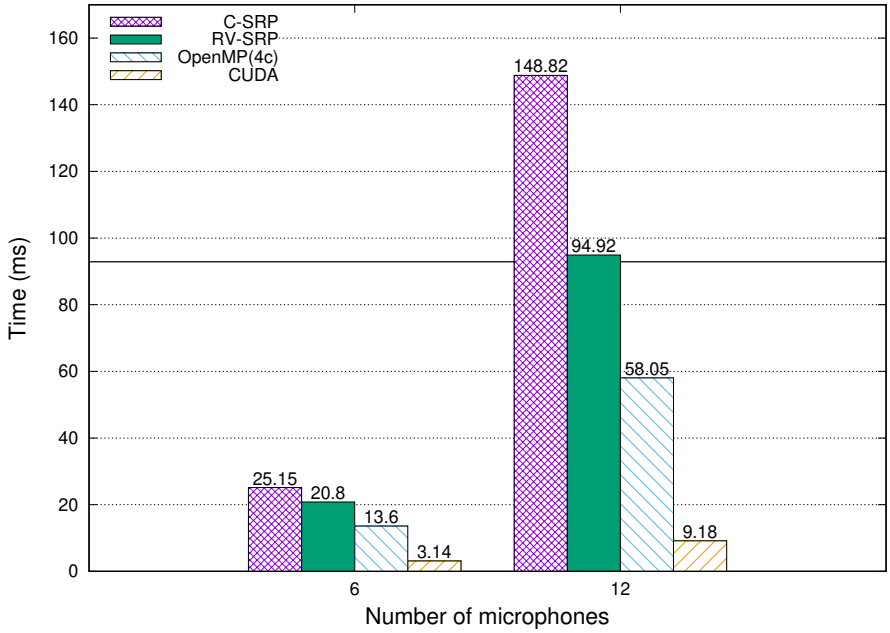

Figure 3: Execution times of the different versions of the SRP method on a Jetson TX1 platform. The horizontal line shows the real-time threshold.

behavior of both types of cores is quite different. In the case of the A7 cores the algorithm consumes more energy at the lowest frequencies and using one core. We obtain the best consumption in the few cases when we can perform the localization in real time. On the contrary, in the case of the A15 cores, the energy consumption increases with the frequency and the best results are obtained at low frequencies (< 1000 MHz) and using 3 or 4 cores.

Finally, Figure 8 allows us to compare the energy consumption of the algorithm in the cases where less energy is consumed by the CPU or the GPU to perform the localization in real time. The OpenMP version is clearly less power consuming than the OpenCL version and in both cases it is better to use the low-power A7 cores. The best option is to execute the OpenMP version of the algorithm on 3 or 4 Cortex-A7 CPU cores at high frequencies.

5.3 Results with the Jetson TX1

The Jetson-TX1 platform allows us to leverage its four CPU cores using OpenMP and also its GPU using CUDA to perform the localization. Both parallel versions clearly improve the sequential execution time and allow us to use finer grids in real time (see figure 9). The CUDA version clearly outperforms the OpenMP version on this platform.

Figures 10 and 11 allow us to analyze and compare the energy consumption

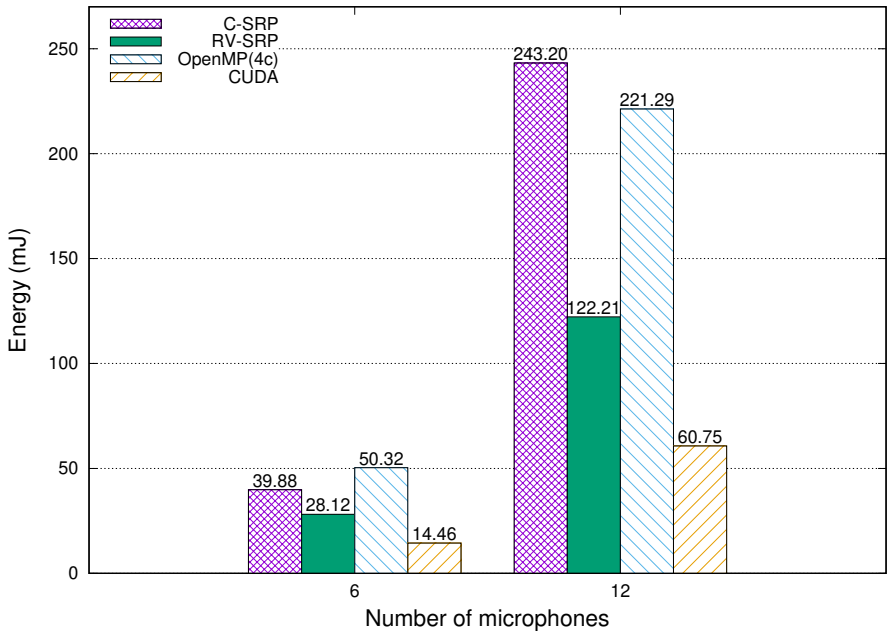

Figure 4: Energy consumption of the different versions of the SRP method on a Jetson TX1 platform.

of the Jetson-TX1 platform using its CPU and GPU cores at their different frequencies. We can see that the CPU cores are much more power-hungry than the GPU even in the less expensive case in terms of energy consumption of the OpenMP algorithm. This behavior is mainly due to the faster execution time of the CUDA algorithm, because the power dissipated by the platform in both cases is quite similar. The energy consumption of the CUDA version is quite stable disregarding the CPU and GPU frequencies and only increases when we combine the lowest GPU frequency with high CPU frequencies and vice versa. In the case of the OpenMP version, increasing the number of cores and reducing the CPU frequencies is beneficial for the energy consumption.

5.4 Results with the Raspberry Pi 3

If we use OpenMP to leverage the four cores of the Raspberry Pi 3, we can perform the localization in real time, which cannot be achieved using the sequential algorithm. In this case we can locate the source using grid resolutions larger than 0.05, as we can see in Figure 12.

On the other hand, the energy consumption of the algorithm decreases with the number of cores and their frequencies. However, by using more than one core it is possible to solve the problem in real time, and the lowest energy consumption is obtained by reducing the core frequencies, as we can see in

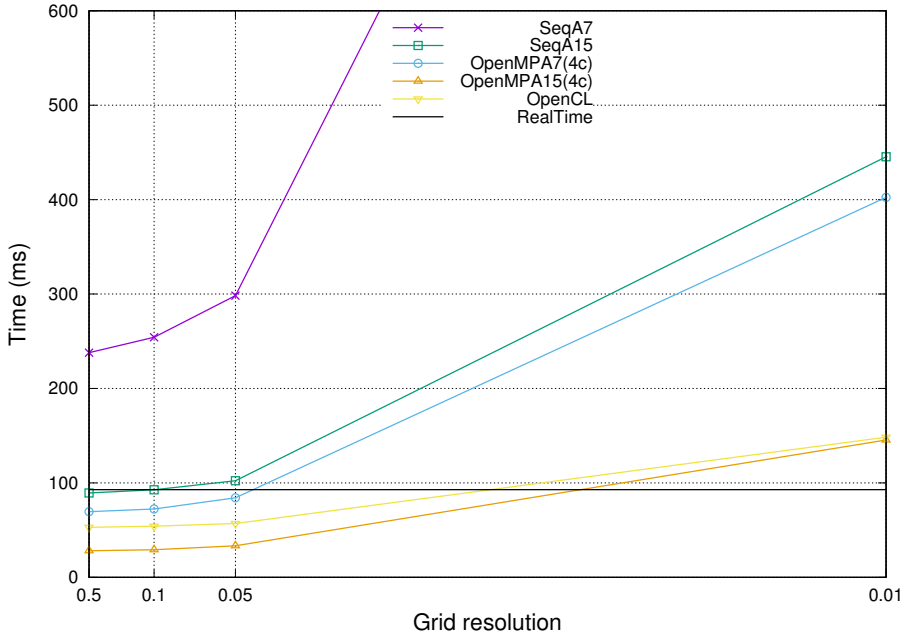

Figure 5: Execution times of the different versions of the algorithm on the ODROID-XU3 platform varying the grid resolution. Sequential versions are executed on one core of each type (A7, A15) and OpenMP versions on the four cores of each type. OpenCL version is executed on the fastest device provided by the Mali GPU and uses one of the A15 cores as host. The horizontal line shows the real-time threshold.

Figure 13.

5.5 Platforms comparison

If we compare the execution times of the fastest algorithms in each of the three experimental platforms, Figure 14 shows that we can use 12 microphones to perform the localization of a sound source using grids with resolutions larger than 0.05 m in all of them. However, the fastest option is clearly to use CUDA to leverage the GPU included in the Jetson-TX1 platform.

If we want to perform the localization while reducing the energy consumption, the best option is also to use the GPU included in the Jetson TX1. The values shown in Figure 15 are always obtained with the best parallel algorithm, number of cores and frequency of the components on each of the platforms that allows us to perform the localization in real time.

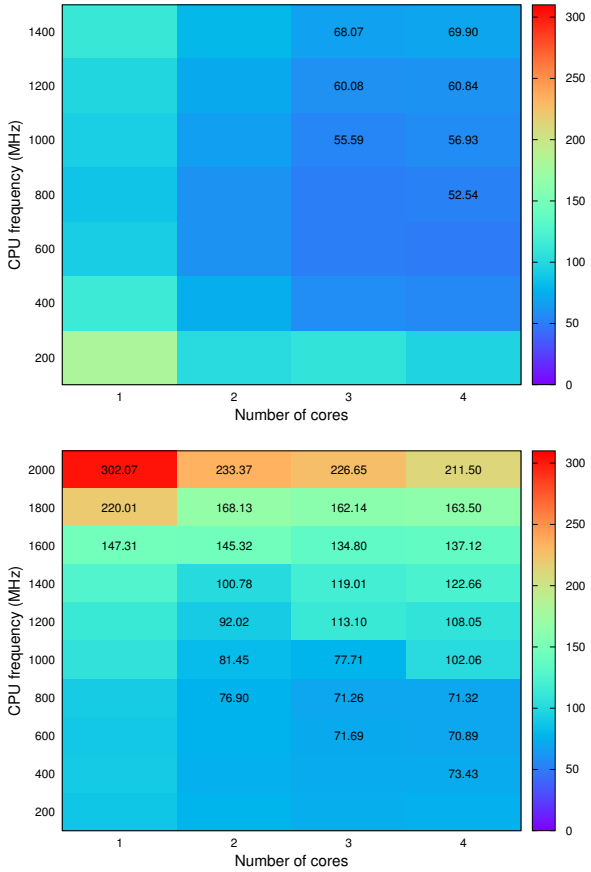

Figure 6: Energy consumed in mJ using the OpenMP algorithm with both kinds of CPU cores of the ODROID platform: A7 (left-hand side) and A15 (right-hand side). We modify both the number of cores and their frequencies. Energy values are only shown in the cases where the localization can be done in real time.

6 conclusions

This paper has shown how to exploit the computational capabilities of recent low-power SoCs to efficiently solve the problem of sound source localization in real time. We have used well-know parallel programming tools, such as OpenMP, CUDA and OpenCL to implement several portable versions of the C-SRP algorithm. This kind of tools allows us to leverage the parallel capabilities of the CPUs and GPUs of the platforms to obtain efficient implementations both in terms of time and energy consumption.

Although low-complexity algorithms such as the RV-SRP can offer great computational advantages over sequential implementations, the use of parallel versions exploiting the computational resources of such platforms has been

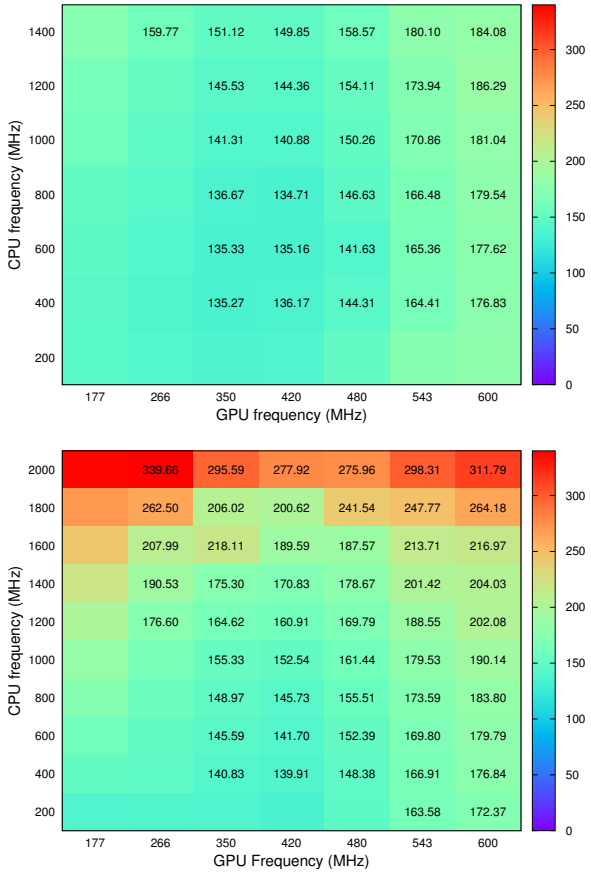

Figure 7: Energy consumed in mJ using the OpenCL algorithm in the Mali GPU and both types of CPU cores of the ODROID platform as host: A7 (left-hand side) and A15 (right-hand side). We modify both the CPU and GPU frequencies. Energy values are only shown in the cases where the localization can be done in real time.

shown to be a key aspect for achieving high-performance real-time systems.

The implementations have been evaluated on three recent SoCs with different features in terms of model of cores, type and amount of memory or accelerator: Raspberry Pi 3, Odroid XU3 and Jetson TX1. Experimental results show that leveraging the computational resources of the three platforms allow us to locate the sound source in real time using up to 12 microphones on a 3D grid with a resolution larger than 0.05. We have also exploited the possibility of controlling the frequencies of the CPU and GPU cores offered by the platforms, and even disabling some of those cores, to reduce the energy consumption of the algorithm. Taking into account the programming tool, number and type of core and frequencies, the best option depends on the platform. For example,

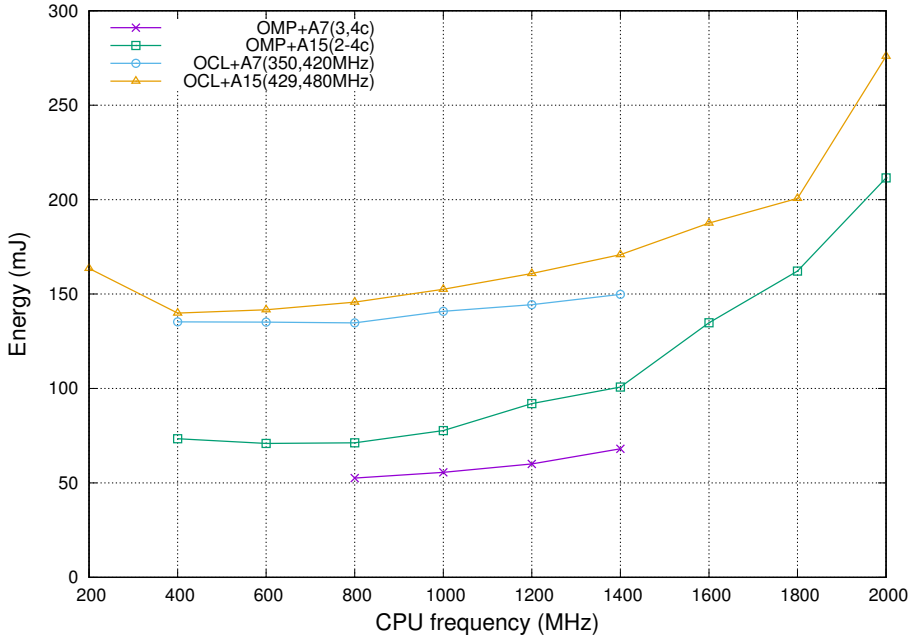

Figure 8: Energy consumed by the OpenMP and OpenCL versions of the algorithm using both kinds of CPU cores of the ODROID platform. Values in brackets show the number of cores or GPU frequencies with a minimum consumption on each scenario. Lines only include the cases where the localization can be done in real time.

the fastest results in the Jetson TX1 are obtained using the CUDA version with the GPU and the CPU at their maximum frequencies. However, the energy consumption is better if we reduce the frequency of both the GPU and the CPU to intermediate values.

All in all, if we want to locate a sound source using one of the three platforms compared, the fastest and lowest energy consuming option is to use CUDA to leverage the GPU included in the Jetson TX1. Moreover, we have shown that appropriately leveraging the resources provided by this kind of low-power platform it is possible to deploy a precise and robust system for acoustic source localization in real time. We hope that our implementations and evaluation can be useful on the design of emerging IoT systems.

As for future work, we intend to parallelize the RV-SRP algorithm using the same techniques that with the C-SRP algorithm and to compare their performance both in terms of time and energy consumption.

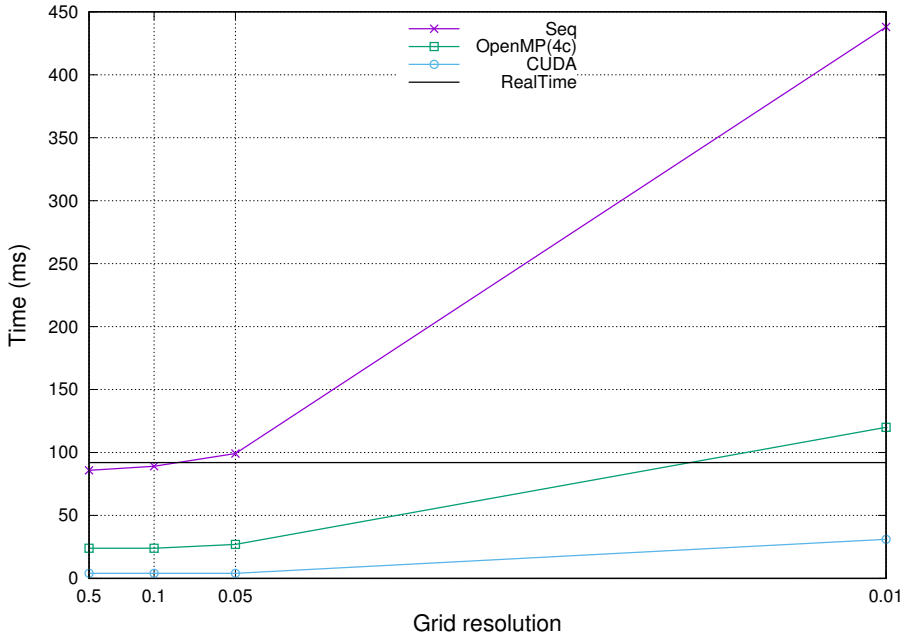

Figure 9: Execution times of the different versions of the algorithm on the Jetson-TX1 platform varying the grid resolution.

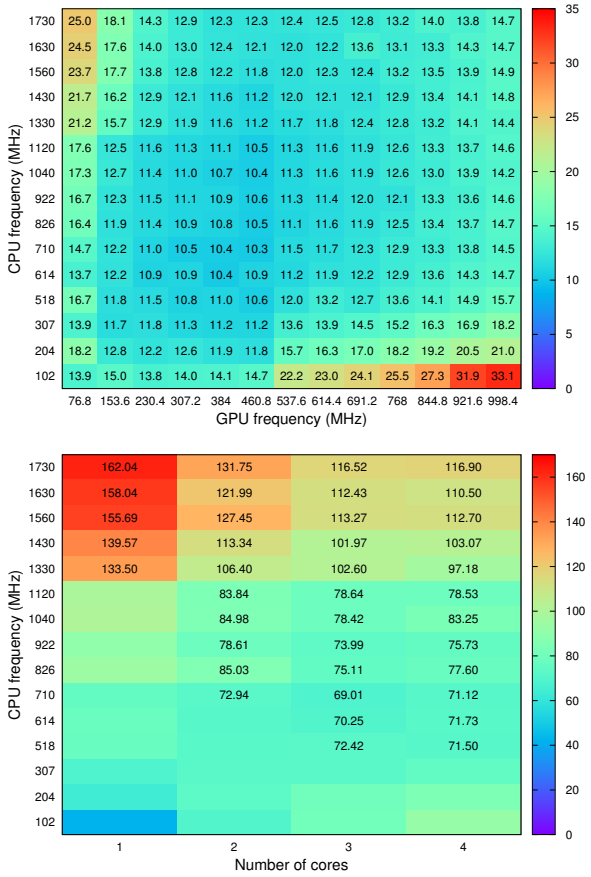

Figure 10: Energy consumed by the Jetson-TX1 platform using CUDA and varying the CPU and GPU frequencies (left-hand side) and using OpenMP on the CPU cores and varying their frequencies (right-hand side). Energy values are only shown in the cases where the localization can be done in real time.

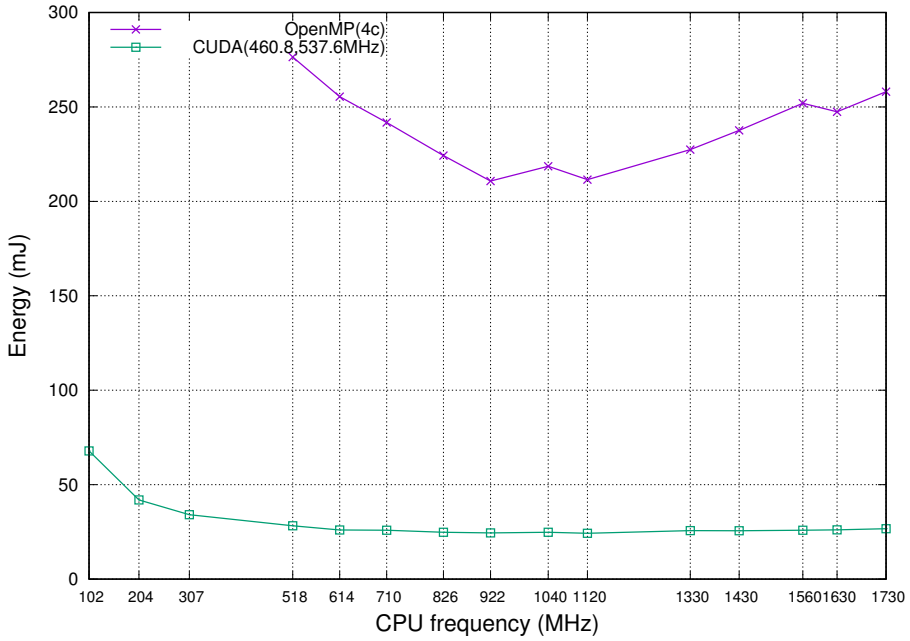

Figure 11: Energy consumed by the Jetson-TX1 platform using OpenMP and CUDA versions of the algorithm. Values in brackets show the number of cores or GPU frequencies with a minimum consumption on each scenario. Lines only include the cases where the localization can be done in real time.

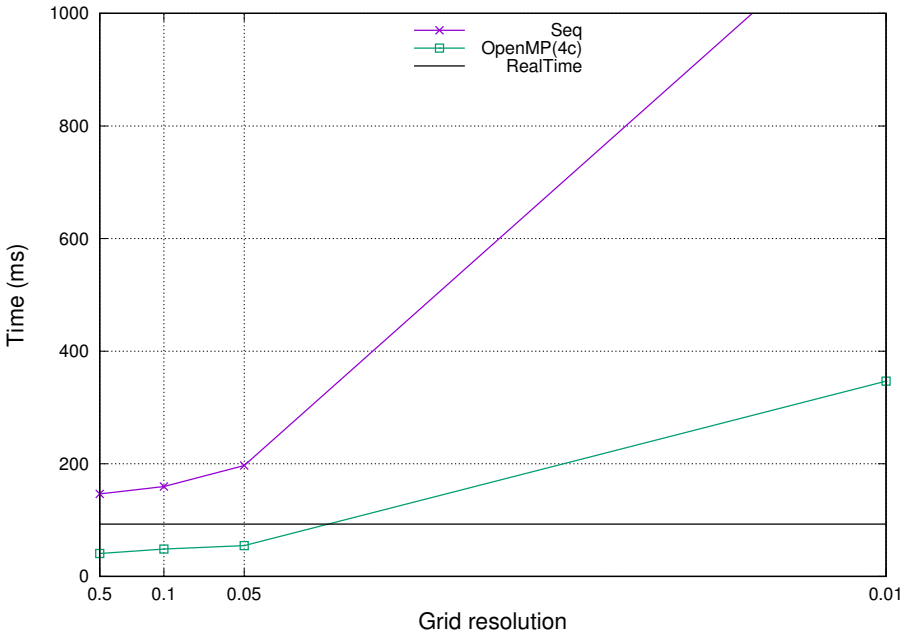

Figure 12: Execution times of the sequential and parallel version of the algorithm on the Raspberry Pi 3 B platform varying the grid resolution.

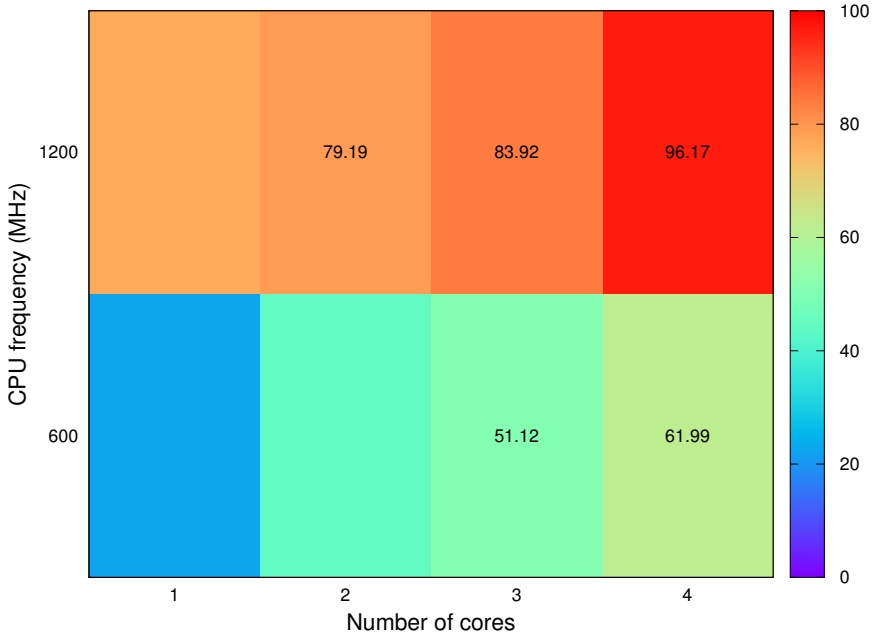

Figure 13: Energy consumed by the Raspberry Pi 3 platform using OpenMP on the CPU cores and varying their frequencies. Energy values are only shown in the cases where the localization can be done in real time.

acknowledgements

his work has been supported by the postdoctoral fellowship from Generalitat Valenciana APOSTD/2016/069, the Spanish Government through TIN2014-53495-R, TIN2015-65277-R and BIA2016-76957-C3-1-R, and the Universidad Jaume I project UJI-B2016-20.

References

[1] Z. Pan, Y. Ge, Y. C. Zhou, J. C. Huang, Y. L. Zheng, N. Zhang, X. X. Liang, P. Gao, G. Q. Zhang, Q. Wang, and S. B. Shi, "Cognitive acoustic analytics service for internet of things," in *2017 IEEE International Conference on Cognitive Computing (ICCC)*, June 2017, pp. 96–103.

[2] H. Chen, F. Li, and Y. Wang, "Soundmark: Accurate indoor localization via peer-assisted dead reckoning," *IEEE Internet of Things Journal*, pp. 1–1, 2018.

[3] M. Cobos, F. Antonacci, A. Alexandridis, A. Mouchtaris, and B. Lee, "A survey of sound source localization methods in wireless acoustic sensor

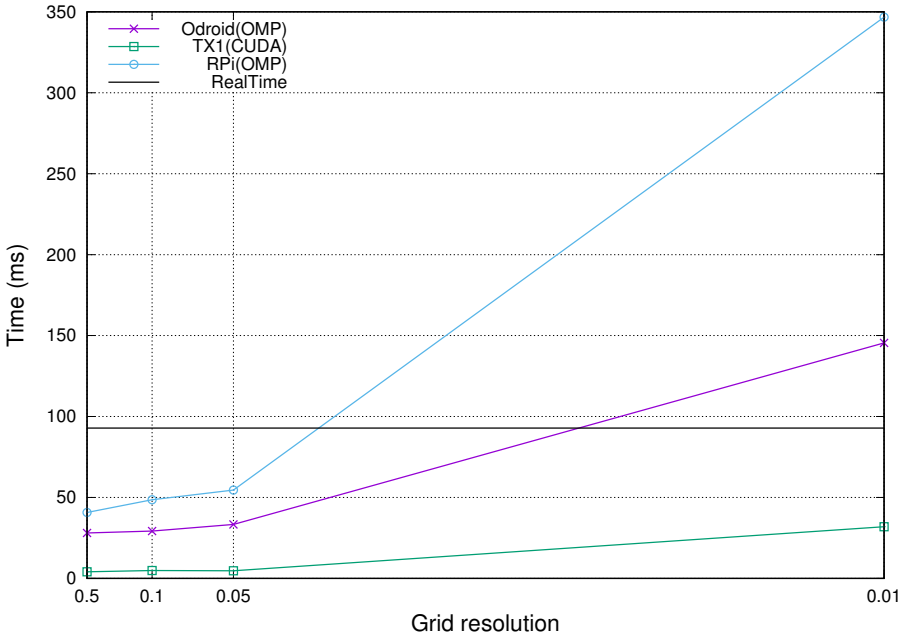

Figure 14: Execution times of the fastest parallel algorithm on each of the three low-power platforms varying the grid resolution.

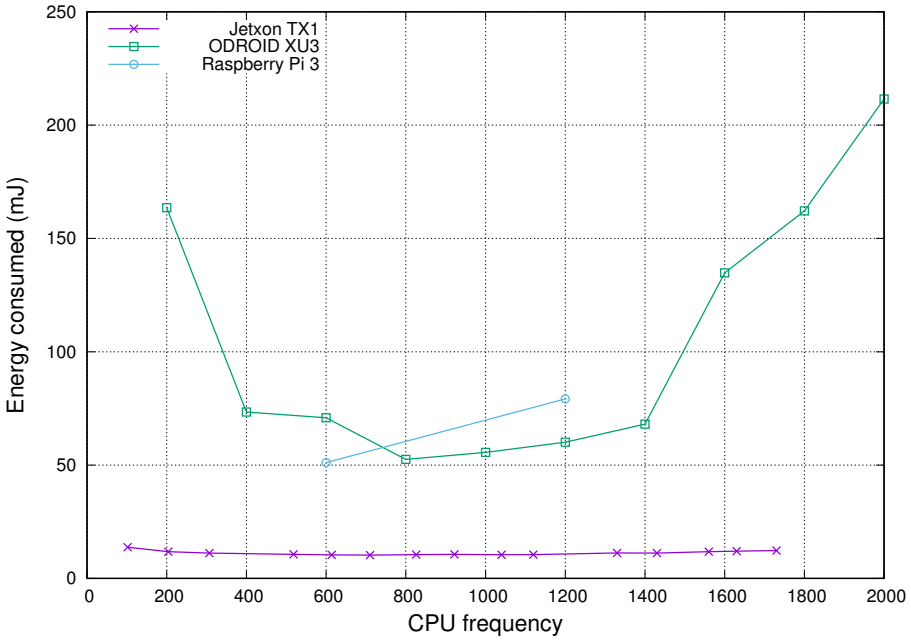

Figure 15: Energy consumed in the best cases on each of the three low-power platforms to perform the localization in real time.

networks," *Wireless Communications and Mobile Computing*, vol. 2017, 2017, article ID 3956282.

[4] P. Cedillo, C. Sanchez, A. Bermeo, and K. Campos, "A systematic literature review on devices and systems for ambient assisted living: Solutions and trends from different user perspectives," in *ICEDG 2018 - 5th International Conference on eDemocracy & eGoverment*. IEEE, April 2018, pp. 59–66.

[5] M. Cobos, J. Perez-Solano, and L. Berger, "Acoustic-based technologies for ambient assisted living," in *Introduction to Smarte eHealth and eCcare Technologies*, S. Merilampi and A. Sirkka, Eds. Boca Raton, FL, USA: Taylor & Francis Group, 2016, ch. 9, pp. 159–180.

[6] J. Segura-Garcia, S. Felici-Castell, J. Perez-Solano, M. Cobos, and J. Navarro, "Low-cost alternatives for urban noise nuisance monitoring using wireless sensor networks," *IEEE Sensors Journal*, vol. 15, no. 2, pp. 836–844, 2015.

[7] L. Ciabattoni, F. Ferracuti, G. Ippoliti, S. Longhi, and G. Turri, "Iot based indoor personal comfort levels monitoring," in *2016 IEEE International Conference on Consumer Electronics (ICCE)*, Jan 2016, pp. 125–126.

[8] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, "A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 829–846, April 2018.

[9] M. Cobos, J. J. Perez-Solano, Ó. Belmonte, G. Ramos, and A. M. Torres, "Simultaneous ranging and self-positioning in unsynchronized wireless acoustic sensor networks," *IEEE Transactions on Signal Processing*, vol. 64, no. 22, pp. 5993–6004, Nov 2016.

[10] M. Cobos, J. Perez-Solano, S. Felici-Castell, J. Segura, and J. Navarro, "Cumulative-sum-based localization of sound events in low-cost wireless acoustic sensor networks," *IEEE/ACM Trans. on Audio, Speech, and Lang. Process.*, vol. 22, no. 12, pp. 1792–1802, 2014.

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

[12] 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*, vol. 18, no. 1, pp. 71–74, 2011.

[13] J. A. Belloch, A. González, A. M. Vidal, and M. Cobos, "On the performance of multi-GPU-based expert systems for acoustic localization involving massive microphone arrays," *Expert Syst. Appl.*, vol. 42, no. 13, pp. 5607–5620, 2015.

[14] J. M. Badía, J. A. Belloch, M. Cobos, F. D. Igual, and E. S. Quintana-Ortí, "Accelerating the SRP-PHAT algorithm on multi- and many-core platforms using OpenCL," *Journal of Supercomputing*, online May, 2018.

[15] F. Conti, R. Schilling, P. D. Schiavone, A. Pullini, D. Rossi, F. K. Grkaynak, M. Muehlberghuber, M. Gautschi, I. Loi, G. Haugou, S. Mangard, and L. Benini, "An iot endpoint system-on-chip for secure and energy-efficient near-sensor analytics," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 9, pp. 2481–2494, Sept 2017.

[16] D. K. Halim, T. C. Ming, N. M. Song, and D. Hartono, "Software-based turbo decoder implementation on low power multi-processor system-on-chip for internet of things," in *2017 4th International Conference on New Media Studies (CONMEDIA)*, Nov 2017, pp. 137–141.

[17] A. R. Hilal and O. Basir, "A collaborative energy-aware sensor management system using team theory," *ACM Trans. Embed. Comput. Syst.*, vol. 15, no. 3, pp. 52:1–52:26, May 2016. [Online]. Available: http://doi.acm.org/10.1145/2910574

[18] A. R. Hilal, A. Sayedelahl, A. Tabibiazar, M. S. Kamel, and O. A. Basir, "A distributed sensor management for large-scale IoT indoor acoustic surveillance," *Future Generation Computer Systems*, vol. 86, pp. 1170 – 1184, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X18300529

[19] S. Ntalampiras, I. Potamitis, and N. Fakotakis, "On acoustic surveillance of hazardous situations," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Taipei, Taiwan, April 2009.

[20] ——, "An adaptive framework for acoustic monitoring of potential hazards," *EURASIP J. Audio Speech Music Process.*, vol. 2009, pp. 13:1–13:15, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1155/2009/594103

[21] A. Rabaoui, M. Davy, S. Rossignol, and N. Ellouze, "Using one-class SVMs and wavelets for audio surveillance," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 4, pp. 763–775, Dec 2008.

[22] S. Kim, J. Cho, and D. Park, "Moving-target position estimation using gpu-based particle filter for iot sensing applications," *Applied Sciences*, vol. 7, no. 11, 2017.

[23] L. Reinfurt, M. Falkenthal, U. Breitenbücher, and F. Leymann, "Applying IoT Patterns to Smart Factory Systems," in *Proceedings of the 11th Advanced Summer School on Service Oriented Computing*. IBM Research Division, 2017, pp. 1–10.

[24] C. J. Grobler, C. P. Kruger, B. J. Silva, and G. P. Hancke, "Sound based localization and identification in industrial environments," in *IECON 2017*

- *43rd Annual Conference of the IEEE Industrial Electronics Society*, Oct 2017, pp. 6119–6124.

[25] V. Exadaktylos, M. Silva, S. Ferrari, M. Guarino, and D. Berckmans, "Sound localisation in practice: An application in localisation of sick animals in commercial piggeries," *IntechOpen*, vol. 1, pp. 575 – 590, 2010.

[26] M. Popescu and A. Mahnot, "Acoustic fall detection using one-class classifiers," in *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Minneapolis, USA, 2009.

[27] C. N. Doukas and I. Maglogiannis, "Emergency fall incidents detection in assisted living environments utilizing motion, sound, and visual perceptual components," *IEEE Transactions on Information Technology in Biomedicine*, vol. 15, no. 2, pp. 277–289, March 2011.

[28] R. M. Alsina-Pags, J. Navarro, F. Alas, and M. Hervs, "homesound: Real-time audio event detection based on high performance computing for behaviour and surveillance remote monitoring," *Sensors*, vol. 17, no. 4, 2017.

[29] A. M. Rahmani, T. N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, and P. Liljeberg, "Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach," *Future Generation Computer Systems*, vol. 78, pp. 641 – 658, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17302121

[30] M. Cobos, M. Garcia-Pineda, and M. Arevalillo-Herraez, "Steered response power localization of acoustic passband signals," *IEEE Signal Processing Letters*, vol. 24, no. 5, pp. 717–721, May 2017.

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

[32] M. Cobos, A. Marti, and J. J. López, "A modified SRP-PHAT functional for robust real-time sound source localization with scalable spatial sampling," *IEEE Signal Process. Lett.*, vol. 18, no. 1, pp. 71–74, 2011.

[33] M. V. Lima, W. A. Martins, L. O. Nunes, L. W. Biscainho, M. V. C. Tadeu N. Ferreira, and B. Lee, "A volumetric srp with refinement step for sound source localization," *IEEE Signal processing letters*, vol. 22, no. 8, pp. 1098–1102, August 2015.

[34] S. Barrachina, M. Barreda, S. Catalán, M. S. Dolz, G. Fabregat, R. Mayo, and E. S. Quintana-Ortí, "An integrated framework for power-performance analysis of parallel scientific workloads," in *Proc. 3rd International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies (ENERGY) (2013)*, Lisbon, Portugal, Mar 2013, pp. 114–119.

[35] "Smart Power device,"
https://goo.gl/esCXLQ, (accessed 2018 June 25).

[36] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005, special issue on "Program Generation, Optimization, and Platform Adaptation".

[37] J. A. Belloch, J. M. Badía, F. D. Igual, M. Cobos, and E. S. Quintana-Ortí, "On the use of a GPU-Accelerated mobile device processor for sound source localization," in *International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland*, 2017, pp. 586–595.

[38] "OpenCL fast fourier transforms," http://clmathlibraries.github.io/clFFT/, (accessed 2018 June 18).

[39] "NVIDIA Library CUFFT," http://developer.download.nvidia.com/compute/DevZone/docs/html/CUDALibraries/doc/CUFFT_Library.pdf.

[40] M. Scarpino, *OpenCL in Action: How to Accelerate Graphics and Computation*. Manning, 2012.