A Vectorized Method for Computationally Efficient SRP-PHAT Sound Source Localization

Bowon Lee and Ton Kalker
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304, USA
{bowon.lee, ton.kalker}@hp.com

***Abstract*—Most microphone array applications require sound source localization for subsequent multichannel signal processing algorithms. Steered response power-based source localization has been the most popular method due to its robustness against reverberation and background noise. Unfortunately, it requires a grid search method whose computational cost is proportional to the number of grid points, which makes it challenging for real-time applications. This paper proposes an efficient method for computing the steered response power by using two different techniques. Firstly we propose to separate pre-runtime computations from runtime computations. Secondly we vectorize the computations and group similar computations with a multi-threaded library (IPP) to efficiently exploit standard processing architectures. Experimental results show that the proposed method significantly reduces the computational load of microphone array processing, making it viable for real-time implementations on constrained devices.**

***Index Terms*—Sound source localization, Steered response power, Microphone array**

I. Introduction

Microphone arrays have been widely used for applications that require robust estimation of speech signals against noise, interfering sources, and reverberation thanks to their spatial selectivity [1]. Such applications generally require sound source localization (SSL) for subsequent multichannel signal processing algorithms such as beamforming, dereverberation, noise suppression, and speaker diarization.

Sound source localization can be done by using a two-stage method which finds time difference of arrival (TDOA) estimations on all possible microphones pairs, followed by hyperbolic localization [2], [3]. The two-stage method is computationally inexpensive, but is subject to fail due to TDOA estimation errors [1]. Unfortunately, background noise and reverberation that are commonly present in most speech processing systems are two major factors that decrease the TDOA estimation accuracies, making this method not suitable for such applications.

Alternatively, we can use search space methods that find a source location having the maximum steered response power (SRP) [4], or the maximum-likelihood (ML). These methods use statistical models based on the Gaussian distribution [5] or the Laplacian distribution [6] to search among all candidate locations. Search space methods are robust, but require a predefined search space resulting in computational complexity proportional to the number of candidate source locations. For example, if searching grid points in a three-dimensional space of $3\,\mathrm{m} \times 3\,\mathrm{m} \times 1\,\mathrm{m}$ (*width* $\times$ *depth* $\times$ *height*) with 2 centimeter spacing, we have a total number of 1162851 candidate locations. With this search space, a system with 32 microphones, 48 kHz sampling frequency, and 2048 point DFT using the SRP with the phase transform (PHAT) frequency weighting requires a computational load of more than 40×10^{12} floating-point operations per second (FLOPS). These numbers are impractical for real-time applications, unless measures are taken to reduce the effective computational complexity or adapt to specific processor architectures.

In order to make the SRP-PHAT algorithm suitable for real-time applications, Johansson and Nordholm [7] proposed the Root-SRP-PHAT algorithm of finding the global extrema for the locations having its first derivative of the SRP curve zero. This method, however, is not robust when the curve has many local maxima which commonly happens at low SNR caused by reverberation and noise. Also, it requires uniform linear array and works for far-field sources because it exploits the symmetry of the array covariance matrix. Recently, Do *et al.* [8] proposed an iterative hierarchical method called *stochastic region contraction* (SRC) to reduce the effective number of candidate sources. Even with these optimizations, a significant computational load is still needed for each candidate source location. At any rate, these methods are not exact implementation of the original SRP-PHAT method because both of them assume smooth SRP curve and may not give the same results as the original method when the SRP curve has many local maxima due to background noise and reverberation.

In this paper, we analyze the computational complexity of the SRP-PHAT algorithm and propose a computationally efficient method by exploiting pre-computation and vectorization. The former refers to separating computations independent of the input data from the SRP-PHAT computations. The latter refers to efficiently using an optimized multi-threaded library that runs efficiently on standard processing architectures. We show that we can considerably reduce the computational cost of SSL. Moreover, the proposed method can be further used in conjunction with hierarchical search space refinement algorithms such as the SRC [8] for even faster implementation.

II. Signal Models

For an array of M microphones, the signal $x_m(t)$ captured at the m^{th} microphone can be expressed as follows

$$x_m(t) = \alpha_m^{\mathbf{q}} s(t - \tau_m^{\mathbf{q}}) + v_m(t), \tag{1}$$

where $s(t)$ is the source signal, $v_m(t)$ is a term due to reverberation, interferences, and background noise, and $\alpha_m^{\mathbf{q}}$ and $\tau_m^{\mathbf{q}}$ denote attenuation and time delay due to propagation of the signal $s(t)$ from a source location $\mathbf{q}$ in the three-dimensional space to the m^{th} microphone. With this signal model, we can express a vector $\mathbf{X}_\omega = [X_1(\omega), X_2(\omega), \cdots, X_M(\omega)]^T$ of the microphone signals in the frequency domain as

$$\mathbf{X}_\omega = S(\omega)\mathbf{D}_\omega^{\mathbf{q}} + \mathbf{V}_\omega, \tag{2}$$

where

$$\mathbf{D}_\omega^{\mathbf{q}} = [D_1^{\mathbf{q}}(\omega), D_2^{\mathbf{q}}(\omega), \cdots, D_M^{\mathbf{q}}(\omega)]^T.$$

The elements $D_m^{\mathbf{q}}(\omega)$ of the delay vector $\mathbf{D}_\omega^{\mathbf{q}}$ express the attenuation $\alpha_m^{\mathbf{q}}$ and phase due to propagation delay $\tau_m^{\mathbf{q}}$, i.e.,

$$D_m^{\mathbf{q}}(\omega) = \alpha_m^{\mathbf{q}} e^{-j\omega\tau_m^{\mathbf{q}}}, \tag{3}$$

which conveys information of the source location $\mathbf{q}$. Thus, SSL is a problem of finding a source location $\mathbf{q}$ corresponding to a set of delay vectors $\mathbf{D}_\omega^{\mathbf{q}}$ given observations $\mathbf{X}_\omega$ for the set of frequencies of interest.

III. Steered Response Power Method

The *steered response power* (SRP) method finds a source location by comparing output power of a filter-and-sum beamformer by steering to all possible source locations [4]. In the frequency domain, this can be expressed as,

$$\hat{\mathbf{q}} = \arg\max_{\mathbf{q}\in\mathcal{Q}} \sum_{m=1}^{M}\sum_{l=1}^{M} \int \psi_{ml}(\omega) X_m(\omega) X_l^*(\omega) e^{j\omega(\tau_m^{\mathbf{q}} - \tau_l^{\mathbf{q}})} d\omega, \tag{4}$$

where $\mathcal{Q}$ denotes the search space of all potential source locations and $\psi_{ml}(\omega)$ denote frequency weightings or *prefilters*. One of the most popular prefilters is the *phase transform* (PHAT), which gives [9]

$$\hat{\mathbf{q}} = \arg\max_{\mathbf{q}\in\mathcal{Q}} \sum_{m=1}^{M}\sum_{l=1}^{M} \int \frac{X_m(\omega) X_l^*(\omega)}{|X_m(\omega) X_l^*(\omega)|} e^{j\omega(\tau_m^{\mathbf{q}} - \tau_l^{\mathbf{q}})} d\omega. \tag{5}$$

Based on the observation that the integrand forms a symmetric matrix with diagonal entries independent of the source location, Do *et al.* [8] suggested that computations can be simplified by reducing the number of summations from M^2 to $M(M-1)/2$. Furthermore, interchanging the order of integration and summation and using its symmetry, Zhang *et al.* [5] showed that Eq. (5) is equivalent to

$$\hat{\mathbf{q}} = \arg\max_{\mathbf{q}\in\mathcal{Q}} \int \left| \sum_{m=1}^{M} \frac{X_m(\omega)}{|X_m(\omega)|} e^{j\omega\tau_m^{\mathbf{q}}} \right|^2 d\omega. \tag{6}$$

which reduces the number of summations by a factor of M.

IV. Computational Complexity Analysis

In this section, we present complexity analysis of the SRP-PHAT method described in the previous section. For this, we separate pre-runtime computations (independent of input data) from run-time computations. Then we analyze the run-time computational complexity of the method as given in Eq. (6) in the DFT domain.

A. SRP-PHAT method in the DFT domain

We can express the SRP-PHAT method in Eq. (6) in the DFT domain as

$$\hat{\mathbf{q}} = \arg\max_{\mathbf{q}\in\mathcal{Q}} \mathcal{P}(\mathbf{q}), \tag{7}$$

where

$$\mathcal{P}(\mathbf{q}) = \sum_{k=0}^{N-1} a^{\mathbf{q}}[k] \tag{8a}$$

$$a^{\mathbf{q}}[k] = |b^{\mathbf{q}}[k]|^2 \tag{8b}$$

$$b^{\mathbf{q}}[k] = \sum_{m=1}^{M} c^{\mathbf{q}}[k] \tag{8c}$$

$$c^{\mathbf{q}}[k] = \frac{X_m[k]}{|X_m[k]|} e^{-j\theta_m^{\mathbf{q}}[k]} \tag{8d}$$

$$\theta_m^{\mathbf{q}}[k] = \frac{2\pi\tau_m^{\mathbf{q}}}{KT} k, \tag{8e}$$

where T denotes the sampling period, K is the DFT size, and N is the number of DFT coefficients selected for SSL, e.g., $N = K/2 + 1$ can be chosen for using the entire spectrum for real input signal or $N \leq K/2$ for the case when a limited bandwidth is considered, e.g., speech sampled at high sampling frequency. Among Eqs. (8a) through (8e), we see that $\theta_m^{\mathbf{q}}[k]$ in Eq. (8e) is independent of the input data $X_m[k]$. So we can pre-compute $e^{j\theta_m^{\mathbf{q}}[k]}$ for $m = 1, 2, \cdots, M$, $k = 0, 1, \cdots, N-1$, and $\forall \mathbf{q} \in \mathcal{Q}$ and store in the memory.

For real-time implementation, we need to consider only the runtime complexity of the SRP-PHAT method. The straightforward execution of the algorithm through Eqs. (7) and (8a)-(8d) is

for $m = 1$ to M **do**
  FFT: $x_m(nT) \rightarrow \{X_m[k]\}$
  for $k = 0$ to $N-1$ **do**
    PHAT: $X_m[k] \rightarrow \frac{X_m[k]}{|X_m[k]|}$
  end for
end for
for all $\mathbf{q} \in \mathcal{Q}$ **do**
  for $k = 0$ to $N-1$ **do**
    for $m = 1$ to M **do**
      $c^{\mathbf{q}}[k] = \frac{X_m[k]}{|X_m[k]|} e^{-j\theta_m^{\mathbf{q}}[k]}$
    end for
    $b^{\mathbf{q}}[k] = \sum_{m=1}^{M} c^{\mathbf{q}}[k]$
    $a^{\mathbf{q}}[k] = |b^{\mathbf{q}}[k]|^2$
  end for
  $\mathcal{P}(\mathbf{q}) = \sum_{k=0}^{N-1} a^{\mathbf{q}}[k]$
end for
Find $\hat{\mathbf{q}} = \arg\max_{\mathbf{q}\in\mathcal{Q}} \mathcal{P}(\mathbf{q})$.

B. Complexity analysis

According to the algorithm described in the previous section, the number of arithmetic operations (real multiplications and additions) at runtime can be described as a function of the number of microphones M, the FFT size K, the number of frequency bins selected for SSL N, and the number of SSL candidates Q as follows.

- Real FFT: $\frac{5}{2}MK\log_2 K$ (multiplications)
- PHAT: $10MN$ ([8]) (multiplications)
- SRP: (Eqs. (8a) through (8d))
  1) $\mathcal{P}(\mathbf{q})$: $2NQ$ (additions)
  2) $a^{\mathbf{q}}[k]$: $2NQ$ (multiplications) + NQ (additions)
  3) $b^{\mathbf{q}}[k]$: $2MNQ$ (additions)
  4) $c^{\mathbf{q}}[k]$: $4MNQ$ (multiplications) + $2MNQ$ (additions)

Typically $Q \gg M, N, K$ as exemplified in Section I, so the SRP calculation stage consumes most of computation time and is the main candidate for further optimizations.

C. Stochastic Region Contraction

According to the complexity analysis, the complexity of the SRP-PHAT method is mainly governed by Q, the number of candidate source locations. Unless we know that we can use a subset with size Q_s out of the entire search space where $Q_s \ll Q$ (e.g., source tracking), we need to compute the SRP of all Q candidate source locations at every frame. In order to reduce the effective number of source locations, Do *et al.* [8] proposed a method referred to as the *stochastic region contraction* (SRC), which is a hierarchical algorithm that iteratively reduces the size of search space, gradually reducing the granularity of the search grid. They showed that the computational complexity can be reduced by more than two orders of magnitude without significantly losing accuracy. Nevertheless, since the SRC is a method to reduce the effective number of source locations, the computational complexity required for each candidate source location still remains the same.

V. Efficient SRP-PHAT Computation

In this section, we describe the use of a multi-threaded library - Intel's Integrated Performance Primitives (Intel IPP) library in our case - for efficiently computing the SRP-PHAT. This method can be used in conjunction with the SRC for even faster implementations.

It is well-known that even for the same number of arithmetic operations, the performance of executing a set of operations is highly dependent upon optimization based on processor architecture, multi-core threading, etc [10]. In particular, the use of a highly optimized library is crucial for real-time applications running algorithms with large computational complexity. For example, the Intel IPP library supports thousands of mathematical operations for vectors and matrices that is highly optimized for various multicore processors [11]. It also uses internal multi-threading to maximize their performance on multi-core processors. Since this library uses optimized implementations of vector/matrix operations, it is necessary to vectorize internal arguments - $a^{\mathbf{q}}[k]$, $b^{\mathbf{q}}[k]$, $c^{\mathbf{q}}[k]$ in Eqs. (8b) through (8d) - to fully exploit its optimization.

A. Vectorization of the SRP-PHAT method

Based on the above discussions, we vectorize computations by defining the following vectors. $\mathbf{a}^{\mathbf{q}} = \{a^{\mathbf{q}}[k]\}$, $\mathbf{b}^{\mathbf{q}} = \{b^{\mathbf{q}}[k]\}$, and $\mathbf{c}^{\mathbf{q}} = \{c^{\mathbf{q}}[k]\}$, Then we execute Eqs. (7) and (8a)-(8d) as follows.

for $m = 1$ to M **do**
  FFT: $x_m(nT) \rightarrow \{X_m[k]\}$
  PHAT: $\{X_m[k]\} \rightarrow \left\{\frac{X_m[k]}{|X_m[k]|}\right\}$
end for
for all $\mathbf{q} \in \mathcal{Q}$ **do**
  for $m = 1$ to M **do**
    $c^{\mathbf{q}} = \left\{\frac{X_m[k]}{|X_m[k]|} e^{-j\theta^{\mathbf{q}}_m[k]}\right\}$
  end for
  $b^{\mathbf{q}} = \left\{\sum_{m=1}^{M} c^{\mathbf{q}}[k]\right\}$
  $a^{\mathbf{q}} = \left\{|b^{\mathbf{q}}[k]|^2\right\}$
  $\mathcal{P}(\mathbf{q}) = \sum_{k=0}^{N-1} a^{\mathbf{q}}[k]$
end for
Find $\hat{\mathbf{q}} = \arg\max_{\mathbf{q}\in\mathcal{Q}} \mathcal{P}(\mathbf{q})$.

By vectorizing with respect to k, we eliminated the nested loops for $k = 0, 1, \cdots, N-1$ present in the algorithm described in Section IV. Even though the number of operations remains the same, we reduce the number of function calls for for each operation by a factor of N, delegating the scheduling of the arithmetic operations to the low-level capabilities of the processor's architecture and the multi-threaded library.

In the following section, we present experimental results to demonstrate performance of the proposed algorithm (named '*Vector*') and the algorithm presented in Section IV (named '*Scalar*').

VI. Experiments

For evaluating computational efficiency of the proposed method, we made multichannel audio recordings with a uniform linear array of eight microphones ($M = 8$) with inter-microphone spacing of 0.15 m in a conference room. For the source speech, we played a pre-recorded clean female speech signal through a loudspeaker, which is located (1.45 m, 3 m) away from the center of the microphone array as illustrated in Fig. VI. Audio is recorded at 48 kHz sampling frequency with 24 bit resolution, which is later downsampled to 32 kHz with 16 bit resolution. The conference room has a dimension of 4.5 m $\times$ 10 m $\times$ 2.5 m with reverberation time $T_{60} = 500$ ms.

The search space used for the experiment is a two-dimensional area with size 3 m $\times$ 2 m having a minimum distance of 3 m from the microphone array (See Fig. VI). The grid points are located in a uniform spacing of 0.2 m. With this search space, the total number of candidate source locations of the search space is $Q = 176$. We set a standard FFT size $K = 2048$, which is the same as the time-domain frame size.

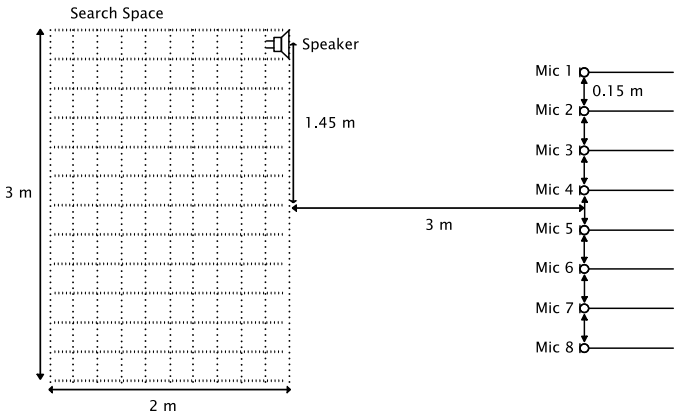

Fig. 1. Experimental configuration

Implementation	*Scalar*	*Vector*
Average CPU Load (%)	39.22	14.46
CPU Load Variance	2.66	2.05

TABLE I
CPU LOAD OF THE *Scalar* AND *Vector* IMPLEMENTATIONS

Since typical speech signals contain most of energy below 8 kHz, we used only a quarter of the FFT bins, i.e., $N = 512$ in the SRP-PHAT algorithm.

For this experimental setup, the number of floating point operations are $\frac{5}{2} \times 8 \times 2048 \log_2 2048 = 450560$ for FFT, $10 \times 8 \times 512 = 40960$ for PHAT, and $8 \times 8 \times 512 \times 176 + 5 \times 512 \times 176 = 6217728$ for SRP, resulting in the total of 6709248 floating point operations per frame. Since the frame rate is $32000/2048 = 15.625$ frames per second, this configuration leads to a total of 104832000 floating point operations per second ($\approx 100\,\mathrm{MFLOPS}$).

We used a HP nw8440 notebook computer with an Intel Core Duo Processor T2500 running at 2 GHz for the experiments. For comparison, we measured the CPU load of the two SRP-PHAT implementations (*Scalar* and *Vector*) running in real-time. The CPU load is measured every two seconds for 100 seconds resulting in 50 measurements for each. The results in terms of mean and variance of the CPU load are summarized in Table I.

The results show that the vectorization reduces the CPU load by a factor of 2.7, even with the same number of arithmetic operations.

VII. CONCLUSION

In this paper, we presented a computationally efficient method for the SRP-PHAT SSL algorithm. By separating pre-runtime and runtime computations, and exploiting the multi-threaded library and vectorizing the internal arguments for runtime computation, we showed that we can significantly reduce the CPU load for SSL computations. In conjunction with iterative search space refinement methods such as the SRC, our proposed method can be realistically applied to real-time microphone array applications for reasonably sized search spaces.

REFERENCES

[1] M. S. Brandstein and D. B. Ward, *Microphone Arrays: Signal Processing Techniques and Applications*. Berlin, Germany: Springer-Verlag, 2001.
[2] M. Brandstein, J. Adcock, and H. Silverman, "A practical time-delay estimator for localizing speech sources with a microphone array," *Computer Speech and Language*, vol. 9, pp. 153–169, 1995.
[3] ——, "A closed-form location estimator for use with room environment microphone arrays," *IEEE Trans. Speech and Audio Process.*, vol. 5, pp. 45–50, 1997.
[4] J. DiBiase, "A high-accuracy, low-latency technique for talker localization in reverberant environments," Ph.D. dissertation, Brown University, Providence, RI, May 2000.
[5] C. Zhang, Z. Zhang, and D. Florêncio, "Maximum likelihood sound source localization for multiple directional microphones," in *Proc. Int. Conf. Acoust., Speech, and Signal Process.*, vol. I, 2007, pp. 125–128.
[6] B. Lee, T. Kalker, and R. W. Schafer, "Maximum-likelihood sound source localization with a multivariate complex Laplacian distribution," in *Proc. Int. Workshop. Acoust. Echo and Noise Control.*, 2008.
[7] A. Johansson and S. Nordholm, "Robust acoustic direction of arrival estimation using Root-SRP-PHAT, a realtime implementation," in *Proc. Int. Conf. Acoust., Speech, and Signal Process.*, vol. IV, 2005, pp. 933–936.
[8] H. Do, H. F. Silverman, and Y. Yu, "A real-time SRP-PHAT source location implementation using stochastic region contraction (SRC) on a large-aperture microphone array," in *Proc. Int. Conf. Acoust., Speech, and Signal Process.*, vol. I, 2007, pp. 121–124.
[9] C. H. Knapp and G. C. Carter, "The generalized correlation method for estimation of time-delay," *IEEE Trans. Acoust., Speech and Audio Process.*, vol. ASSP-24, no. 4, pp. 320–327, 1976.
[10] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 2006.
[11] "Intel integrated performance primitives (intel IPP) 6.1: In-depth," Intel Corporation, http://software.intel.com/sites/products/collateral/hpc/ipp/ipp_indepth%.pdf.