Zafar Rafii

Sliding Discrete Fourier Transform with Kernel Windowing

The sliding discrete Fourier transform (SDFT) is an efficient method for computing the N-point DFT of a given signal starting at a given sample from the N-point DFT of the same signal starting at the previous sample [1]. However, the SDFT does not allow the use of a window function, generally incorporated in the computation of the DFT to reduce spectral leakage, as it would break its sliding property. This article will show how windowing can be included in the SDFT by using a kernel derived from the window function, while keeping the process computationally efficient. In addition, this approach allows for turning other transforms, such as the modified discrete cosine transform (MDCT), into efficient sliding versions of themselves.

Relevance

The SDFT can be used to perform spectral analysis on successive samples in a signal without having to compute a new DFT from scratch every time, provided that windowing can be incorporated into the computation of the DFTs without harming the efficiency of the method. A notable application of the SDFT with windowing can then be framing detection in audio signals that have undergone lossy compression in the context of audio compression identification [2]. A lossy compression algorithm will typically introduce traces of compression in the signal being encoded, which can become visible in the time-frequency representation when using the same parameters and framing that were used for the encoding. Therefore, the parameters and framing can be recovered by computing time-frequency representations at successive samples in the signal and identifying when traces of compression become visible. This demanding process can be translated into an efficient one by using the SDFT with kernel windowing.

Prerequisites

Basic knowledge of digital signal processing is required to understand this article, particularly concepts such as the DFT, windowing, and general spectral analysis. More details about the SDFT and lossy audio compression identification can also be found in [1] and [2], respectively.

Problem statement and solution

Problem statement

The SDFT allows for the computation of the N-point DFT of a signal from the N-point DFT of the same signal starting one sample earlier, in a sense by sliding a rectangular window of length N one sample forward. The SDFT essentially relies on the shift theorem, which states that multiplying a signal by a linear phase is equivalent to a circular shift in the corresponding DFT.

Equation (1), shown at the bottom of the page, shows the derivation of $X^{(i)}$, the N-point DFT of signal x starting at sample i, from $X^{(i-1)}$, the N-point DFT of x starting at $i-1$, a process hence known as SDFT.

The SDFT thus only requires two N additions and N multiplications, leading to a linear time complexity of $O(N)$, while the full and direct computation of the DFT and the fast Fourier transform (FFT) are $O(N^2)$ and $O(N \log N)$, respectively.

Transforms such as the DFT typically use a window function in their computation to reduce spectral leakage and enhance spectral analysis. However, the SDFT does not allow the incorporation of a window function as it will break the process shown in (1). One solution would be to perform the windowing in the frequency domain, i.e., on the derived DFT through convolution. A practical window function for that matter could be the Hanning window, as the corresponding windowing in the frequency domain equals a simple three-point convolution [1]. Other window functions, however, may not be as practical, as the corresponding convolutions may involve many more operations, which will ultimately hurt the computational efficiency of the SDFT. Therefore, the problem is to incorporate any window function into the computation of the DFTs in an efficient manner without breaking the SDFT process.

Solution: Kernel windowing

The idea of performing the windowing in the frequency domain can still be exploited by reformulating the convolution

$$\begin{aligned}
\underset{0 \le k < N}{X_k^{(i)}} &= \sum_{n=0}^{N-1} x_{i+n} e^{\frac{-j2\pi nk}{N}} \\
&= \sum_{n=0}^{N-1} x_{i+n} e^{\frac{-j2\pi(n+1)k}{N}} e^{\frac{j2\pi k}{N}} \\
&= \sum_{n=1}^{N} x_{i-1+n} e^{\frac{-j2\pi nk}{N}} e^{\frac{j2\pi k}{N}} \\
&= \left(\sum_{n=0}^{N-1} x_{i-1+n} e^{\frac{-j2\pi nk}{N}} - x_{i-1} + x_{i+N-1} \right) e^{\frac{j2\pi k}{N}} \\
&= \left(X_k^{(i-1)} - x_{i-1} + x_{i+N-1} \right) e^{\frac{j2\pi k}{N}}. \qquad (1)
\end{aligned}$$

Digital Object Identifier 10.1109/MSP.2018.2855727
Date of publication: 13 November 2018

as a multiplication by a kernel that can be derived from any window function. Such a kernel will be independent from the signal to be processed and only need to be computed once. It will typically have a very small number of values that would be significant, which means that most of the values can then be ignored. This would lead to a very sparse kernel, which can then be applied to the DFT of the signal, producing results virtually equivalent to the DFT of the same signal modified by the corresponding window function while preserving the computational efficiency of the SDFT.

The constant-Q transform (CQT) is a transform with a logarithmic frequency resolution that was proposed as a more adapted alternative to the FT for analyzing music signals [3]. A fast algorithm was proposed soon after, which translated the slow computation of the CQT into the multiplication of a DFT, which can be efficiently computed using the FFT, and a kernel, which is computed once beforehand and typically very sparse [4]. The idea was to use Parseval's theorem to turn the direct computation in the time domain into a multiplication between a DFT and a kernel in the frequency domain, essentially demonstrating the property of energy conversation between the time and the frequency domains [5].

Parseval's theorem is recalled in (2). The value X is the N-point DFT of x and $\bar{x}$ represents the complex conjugate of x

$$\sum_{n=0}^{N-1} x_n \overline{y_n} = \frac{1}{N}\sum_{k=0}^{N-1} X_k \overline{Y_k}. \tag{2}$$

Following a similar idea, we propose the use of Parseval's theorem to translate the DFT of a windowed signal into the DFT of the signal, multiplied by a kernel that is derived from the corresponding window function: a multiplication that will happen after the SDFT process. Unlike in the fast CQT case, the purpose here is not to speed up the computation of the transform by taking advantage of the efficiency of the FFT algorithm in conjunction with the use of sparse kernel but to extract the windowing operation from the DFT computation so that the SDFT process shown in (1) still holds.

Equation (3), shown at the bottom of the page, illustrates the computation of $\mathcal{X}^{(i)}$, the N-point DFT of the signal x starting at sample i and modified by the window function w, from $X^{(i-1)}$, the N-point DFT of x starting at $i-1$ without windowing, multiplied by the kernel K, which is derived from w.

As we can see in (3), the kernel is completely independent from the signal; therefore, it only needs to be computed once, before the SDFT process. Furthermore, given the nature of such kernel, typically only a very small number of its values will be significant, which means that most of the values can then be zeroed, given some threshold, leading to a very sparse kernel. The multiplication of the derived DFT by such kernel will thus only involve few more operations, keeping the whole process computationally efficient.

A notable application of the SDFT with windowing can then be framing detection in audio signals that have undergone lossy compression in the context of audio compression identification.

Figure 1 shows the kernels derived from some common window functions, i.e., Hanning, Blackman, triangular, Gaussian, Parzen, and Kaiser windows. As we can see, the Hanning window kernel shows only three nonzero values per row, confirming that the corresponding windowing in the frequency domain equals a simple three-point convolution, while the Blackman window kernel shows five nonzero values per row. Both those windows are actually special cases of the generalized cosine window whose corresponding windowing in the frequency domain equals convolutions with typically only few points. Unlike the Hanning and Blackman window kernels, the triangular, Parzen, Gaussian, and Kaiser window kernels show additional nonzero values around their main diagonal, suggesting that the corresponding windowings in the frequency domain equal convolutions with many more points. However, most of those nonzero values have very small magnitudes ($\ll 0.01$) and could then be ignored without significantly affecting the actual windowing process. By using an appropriate threshold, those kernels can therefore be made very sparse with only a few meaningful values per row in the same manner as in the fast CQT case [4].

As proposed in [4], we computed for each of those kernels the error in keeping the values greater than a chosen threshold by dividing the sum of the magnitudes of the values after thresholding by the sum of the magnitudes of all the values before thresholding. A threshold of 0.01 will thus give very small errors of 0.049, 0.009, 0.020, and 0.015, for the triangular, Parzen, Gaussian, and Kaiser window kernels, respectively, when derived for an N-point DFT with $N = 2{,}048$. With such a threshold, the first three kernels will then only have

$$\begin{aligned}
\underset{0\le k<N}{\mathcal{X}_k^{(i)}} &= \sum_{n=0}^{N-1} x_{i+n} \underbrace{w_n e^{\frac{-j2\pi nk}{N}}}_{\overline{y_n}} \\
&= \sum_{k'=0}^{N-1} X_{k'}^{(i)} \underbrace{K_{k,k'}}_{\frac{1}{N}\overline{Y_{k'}}} \\
&= \sum_{k'=0}^{N-1} \left[\left(X_{k'}^{(i-1)} - x_{i-1} + x_{i+N-1} \right) e^{\frac{j2\pi k'}{N}} \right] K_{k,k'} \\
\underset{\substack{0\le k<N \\ 0\le k'<N}}{K_{k,k'}} &= \frac{1}{N}\overline{Y_{k'}} = \frac{1}{N}\overline{\sum_{n=0}^{N-1} y_n e^{\frac{-j2\pi nk'}{N}}} \\
&= \frac{1}{N}\overline{\sum_{n=0}^{N-1} \overline{w_n e^{\frac{-j2\pi nk}{N}}}\, e^{\frac{-j2\pi nk'}{N}}} \\
&= \frac{1}{N}\sum_{n=0}^{N-1} w_n e^{\frac{j2\pi n(k'-k)}{N}}.
\end{aligned} \tag{3}$$

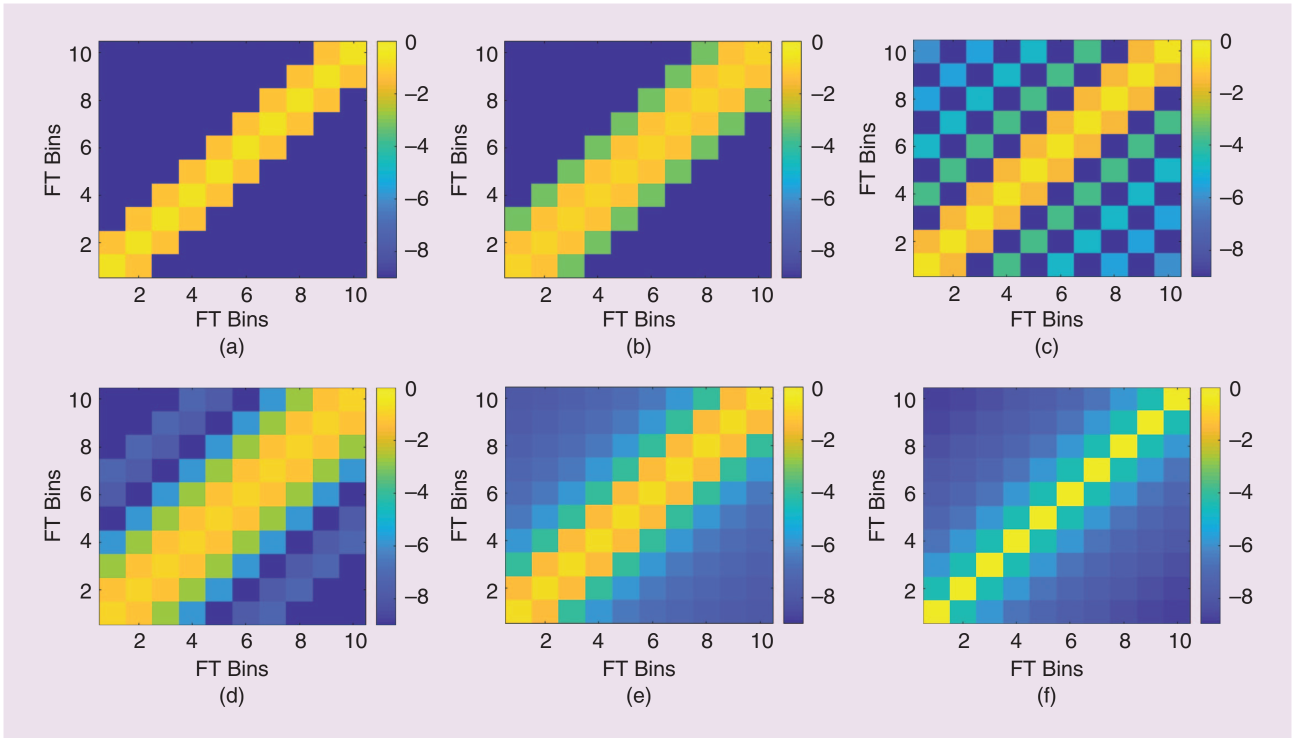

FIGURE 1. The kernels derived from the (a) Hanning, (b) Blackman, (c) triangular, (d) Parzen, (e) Gaussian (with $\alpha = 2.5$), and (f) Kaiser (with $\beta = 0.5$) windows. The kernels were derived for an N-point DFT where $N = 2{,}048$ samples. Only the first 100 coefficients at the bottom-left corner of the N-by-N kernels are shown. The values are displayed in log of amplitude.

five nonzero values per row, while the latter one will have three nonzero values per row. This shows that only a very small number of values is actually significant in such kernels. The multiplication of the DFT by those very sparse kernels will then only involve KN multiplications and KN additions, with $K = 3$ or 5, barely affecting the computational efficiency of the SDFT, still maintaining a linear complexity of $O(N)$, and producing results virtually equivalent to taking the DFT of the signal modified by the corresponding window functions.

Computational examples

Framing detection and lossy audio coding

The SDFT with kernel windowing can be particularly useful for fast framing detection in the context of audio compression identification. Audio compression identification is the recovery of information regarding the data compression that an audio signal has undergone. In particular, the recovery of the parameters and framing used at the time-frequency decomposition stage of the encoding could allow for identifying the coding format or detecting alterations in audio signals that have undergone lossy compression [2], [6]–[9]. Lossy compression algorithms typically introduce traces of compression in the audio signal being encoded in the form of time-frequency coefficients quantized to zeros, which can become visible when using the same parameters and framing that were used for the encoding. One approach to identify if and when lossy compression was used would then be to compute the time-frequency representation at successive samples in the audio signal and search for traces of compression every time, given a set of parameters associated with a known coding format, such as time-frequency transform, window length, and window function, a process also known as *framing detection*.

Lossy audio coding formats, perhaps the most popular ones being MP3, Advanced Audio Coding (AAC), AC-3, Vorbis, and Windows Media Audio (WMA), are widely used for storage (e.g., in music and video files) or transmission (e.g., in radio and television broadcasting). Compression algorithms that can encode to such formats first transform the audio signal into a time-frequency representation, derive a psychoacoustic model to locate regions of perceptually less significance, then quantize the data given the psychoacoustic model, and, finally, convert it into a bitstream. The transform used at the time-frequency decomposition stage is typically based on the MDCT, and a variety of window lengths and window functions can be used depending on the coding format. In particular, specialized window functions such as the sine, slope, and Kaiser–Bessel-derived (KBD) windows, are generally required for the MDCT to be invertible. For more information about lossy audio coding, see [10]. The computation of the MDCT without windowing is

$$\underset{0 \le k < \frac{N}{2}}{Y_k} = \sum_{n=0}^{N-1} x_n \cos\left[\frac{2\pi}{N}\left(n + \frac{1}{2} + \frac{N}{4}\right)\left(k + \frac{1}{2}\right)\right]. \quad (4)$$

Sliding MDCT with kernel windowing

In this context, performing framing detection for audio compression identification would involve computing an MDCT for every set of window length and window function associated with a known lossy coding format at successive samples in the audio signal and searching for time-frequency coefficients quantized to zero until one of the sets shows visible traces of compression for a specific framing of the signal. We can see that such a process will be computationally demanding, as a full transform would have to be computed every time. The direct computation of the MDCT, including the windowing using one of the specialized window functions presented earlier, can actually be translated into an SDFT with a kernel windowing by incorporating the computation of the window function and a part of the MDCT into a kernel, which will still happen to be very sparse, thus making the process computationally efficient.

Equation (5), shown at the bottom of the page, shows the computation of $\mathcal{Y}^{(i)}$, the N-point MDCT of signal x starting at sample i and modified by the window function w, from $X^{(i-1)}$, the N-point DFT of x starting at $i-1$ without windowing, multiplied by the kernel K, which is derived from w.

Figure 2 shows the kernels derived for an N-point MDCT, from the sine window where $N = 1,152$ samples, as in MP3, from the slope window where $N = 2,048$ samples, as in Vorbis, and from the KBD window where $N = 512$ samples, as in AC-3. As we can see, most of the values in those kernels appear to have negligible magnitudes, while the very few values with significant magnitudes appear to be concentrated around two diagonals, one going from the bottom-left to the top-center and one going from the top-center to the bottom-right. As in [4], we computed for each of those kernels the error in keeping the values greater than 0.01 and obtained very small errors of 0.000, 0.022, and 0.013, for the sine, slope, and KBD window kernel, respectively. With such a threshold, the sine kernel will only have around two nonzero values per row and the slope and KBD kernels around six nonzero values per row. Therefore, these very sparse kernels will barely affect the computational efficiency of the SDFT while still producing results equivalent to taking the MDCT of the signal modified by the corresponding window functions.

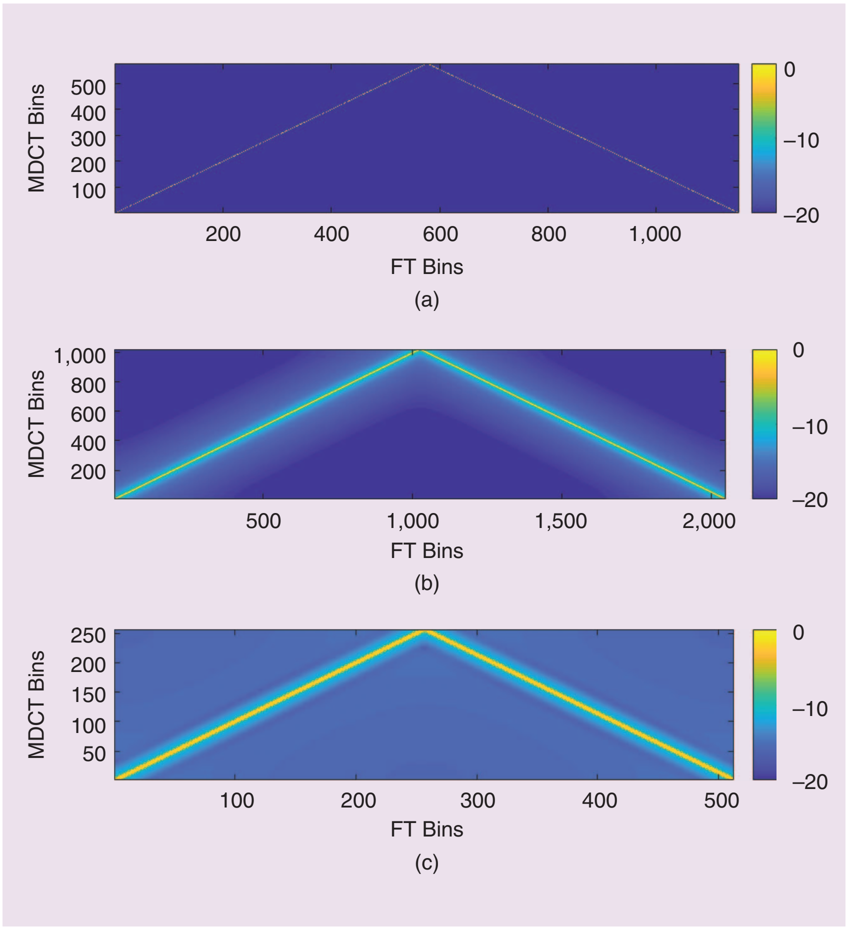

FIGURE 2. The kernels derived for an N-point MDCT, from (a) the sine window where $N = 1,152$ samples, (b) the slope window where $N = 2,048$ samples, and (c) the KBD window where (c) $N = 512$ samples. The values are displayed in log of amplitude.

$$
\begin{aligned}
\underset{0\le k<\frac{N}{2}}{\mathcal{Y}_k^{(i)}} &= \sum_{n=0}^{N-1} x_{i+n}\underbrace{w_n\cos\Big[\frac{2\pi}{N}\Big(n+\frac{1}{2}+\frac{N}{4}\Big)\Big(k+\frac{1}{2}\Big)\Big]}_{y_n}\\
&= \sum_{k'=0}^{N-1} X_{k'}^{(i)}\underbrace{K_{k,k'}}_{\frac{1}{N}\overline{Y_{k'}}}\\
&= \sum_{k'=0}^{N-1}\Big[(X_{k'}^{(i-1)} - x_{i-1} + x_{i+N-1})e^{\frac{j2\pi k'}{N}}\Big]K_{k,k'}\\
\underset{\substack{0\le k<\frac{N}{2}\\ 0\le k'<N}}{K_{k,k'}} &= \frac{1}{N}\overline{Y_{k'}} = \frac{1}{N}\overline{\sum_{n=0}^{N-1} y_n e^{\frac{-j2\pi nk'}{N}}}\\
&= \frac{1}{N}\overline{\sum_{n=0}^{N-1}\overline{w_n\cos\Big[\frac{2\pi}{N}\Big(n+\frac{1}{2}+\frac{N}{4}\Big)\Big(k+\frac{1}{2}\Big)\Big]}e^{\frac{-j2\pi nk'}{N}}}\\
&= \frac{1}{N}\sum_{n=0}^{N-1} w_n\cos\Big[\frac{2\pi}{N}\Big(n+\frac{1}{2}+\frac{N}{4}\Big)\Big(k+\frac{1}{2}\Big)\Big]e^{\frac{j2\pi nk'}{N}}. \qquad (5)
\end{aligned}
$$

What we have learned

We have shown that the SDFT can incorporate windowing in its computation by using a kernel that can be derived from any window function and can be made very sparse. This SDFT with kernel windowing will produce results equivalent to the DFT of the signal modified by the corresponding window function, while keeping the process computationally efficient. This approach may be applied in audio compression identification, in particular by making the process of framing detection much more efficient, allowing for the translation of a transform, such as the MDCT, into an efficient sliding version of itself.

Author information

Zafar Rafii (zafar.rafii@nielsen.com) received his M.S. degree in electrical engineering from the Ecole Nationale Superieure de l'Electronique et de ses Applications, Cergy, France, and another M.S. degree in the same field from the Illinois Institute of Technology, Chicago, in 2006. He received his Ph.D. degree in electrical engineering and computer science from Northwestern University, Evanston, Illinois, in 2014. He is currently a senior research engineer at Gracenote. He also previously worked as a research engineer at Audionamix in France. His research interests focus on audio analysis, somewhere between signal processing, machine learning, and cognitive science, with a predilection for source separation and audio identification in music.

References

[1] E. Jacobsen and R. Lyons, "The sliding DFT," *IEEE Signal Process. Mag.*, vol. 20, no. 2, pp. 74–80, Mar. 2003.

[2] B. Kim and Z. Rafii, "Lossy audio compression identification," in *Proc. 26th European Signal Processing Conf.*, Rome, Italy, Sept. 2018.

[3] J. C. Brown, "Calculation of a constant Q spectral transform," *J. Acoust. Soc. Amer.*, vol. 89, no. 1, pp. 425–434, Jan. 1991.

[4] J. C. Brown and M. S. Puckette, "An efficient algorithm for the calculation of a constant Q transform," *J. Acoust. Soc. Amer.*, vol. 92, no. 5, pp. 2698–2701, Nov. 1992.

[5] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing.* Englewood Cliffs, NJ: Prentice Hall, 1975.

[6] J. Herre and M. Schug, "Analysis of decompressed audio—The 'inverse decoder,'" in *Proc. 109th Audio Engineering Society Conv.*, Los Angeles, CA, Sept. 2000, p. 5256.

[7] S. Moehrs, J. Herre, and R. Geiger, "Analysing decompressed audio with the 'inverse decoder'—Towards an operative algorithm," in *Proc. 112th Audio Engineering Society Conv.*, Munich, Germany, Apr. 2002, p. 5576.

[8] R. Yang, Z. Qu, and J. Huang, "Detecting digital audio forgeries by checking frame offsets," in *Proc. 10th ACM Workshop on Multimedia and Security*, Oxford, U.K., Sept. 2008, pp. 21–26.

[9] D. Gärtner, C. Dittmar, P. Aichroth, L. Cuccovillo, S. Mann, and G. Schuller, "Efficient cross-codec framing grid analysis for audio tampering detection," in *Proc. 136th Audio Engineering Society Conv.*, Berlin, Apr. 2014, pp. 306–316.

[10] M. Bosi and R. E. Goldberg, *Introduction to Digital Audio Coding and Standards*. New York: Springer, 2003.

SP

IEEE Signal Processing Society

JOIN the

IEEE Signal Processing Cup 2019:

Search & Rescue with Drone-Embedded Sound Source Localization

The IEEE Signal Processing Society is proud to announce the sixth edition of the Signal Processing Cup: an exciting audio-based drone-embedded search and rescue challenge.

- **Goal:** Building a system capable of localizing a sound source based on audio recordings made with an 8-channel microphone array embedded in an unmanned aerial vehicle (UAV)
- **Eligibility:** Any team composed of one faculty member, at most one graduate student and 3-10 undergraduate students is welcomed to join the open competition
- **Dataset:** A novel dataset of UAV-embedded microphone-array recordings is provided for the challenge
- **Website:** The detailed guidelines, dataset and inscription portal are available on the official website: https://signalprocessingsociety.org/get-involved/signal-processing-cup
- **Prize:** The three teams with highest performance in the open competition will be selected as finalists and will be invited to participate in the final competition at ICASSP 2019. The champion team will receive a grand prize of $5,000. The first and the second runner-up will receive a prize of $2,500 and $1,500, respectively, in addition to travel grants and complimentary conference registrations.

A joint initiative of

The IEEE Technical Committee for Audio and Acoustic Signal Processing

The IEEE Autonomous System Initiative

Important dates:

Data release: November 14, 2018
Submission deadline: February 28, 2019
Finalists announcement: March 20, 2019
Final @ICASSP: May 12-17, 2019

Sponsored by MathWorks®

Digital Object Identifier 10.1109/MSP.2018.2876581