Sliding DFT with Kernel Windowing

Zafar Rafii
03/30/21

Introduction

- French (pardon the accent!)
- Research engineer at Audionamix; worked on audio source separation
- PhD in EE and CS at Northwestern University
  - Worked at the Interactive Audio Lab with Prof. Bryan Pardo
  - Thesis on audio source separation using repetition (REPET)
- Senior research engineer at Gracenote; working on everything audio-related
  - Live/cover song identification
  - Audio fingerprinting
  - **Audio encoding analysis**
  - Audio beamforming
  - Audio classification
- Co-organizer of the San Francisco-BISH Bash

Plan

- Sliding DFT
  - Discrete Fourier transform
  - Definition and derivation of the SDFT
  - Limitation of the SDFT
- Kernel Windowing
  - Parseval’s theorem
  - Derivation of the SDFT-KW
  - Signal-independent and sparse kernels
- Application
  - Framing detection
  - Sliding modified discrete cosine transform (MDCT) with kernel windowing
- Analysis
  - Sparsification errors
  - Computational complexity

Sliding DFT: Discrete Fourier transform

The DFT can turn a discrete time-domain signal of N samples into a discrete complex frequency-domain spectrum of N frequency indices.

DFT at frequency index *k*

$$\underset{0 \le k < N}{X_k} = \sum_{n=0}^{N-1} x_n e^{\frac{-j2\pi nk}{N}}$$

Signal at sample index *n*

Sliding DFT: Definition of the SDFT

The SDFT is an algorithm for computing the N-point DFT of a signal starting at a sample from the N-point DFT of the same signal starting at the previous sample.

$$X_k^{(i)} = \left(X_k^{(i-1)} - x_{i-1} + x_{i+N-1} \right) e^{\frac{j2\pi k}{N}}$$

$$0 \leq k < N$$

DFT of *x* starting at *i*

Signal at *i-1*

Phase

DFT of *x* starting at *i-1*

Signal at *i+N-1*

Sliding DFT: Derivation of the SDFT

The SDFT essentially relies on the shift theorem which shows that the DFT of a shifted signal equals to the DFT of the original signal multiplied by a phase.

$$\underset{0\le k<N}{X_k^{(i)}} = \sum_{n=0}^{N-1} x_{i+n} e^{\frac{-j2\pi nk}{N}}$$ DFT of x starting at i

$$= \sum_{n=0}^{N-1} x_{i+n} e^{\frac{-j2\pi(n+1)k}{N}} e^{\frac{j2\pi k}{N}}$$ Get the phase out

$$= \sum_{n=1}^{N} x_{i-1+n} e^{\frac{-j2\pi nk}{N}} e^{\frac{j2\pi k}{N}}$$ Reorganize the indices

$$= \left(\sum_{n=0}^{N-1} x_{i-1+n} e^{\frac{-j2\pi nk}{N}} - x_{i-1} + x_{i+N-1} \right) e^{\frac{j2\pi k}{N}}$$ Extract the DFT of x starting at i-1

$$= \left(X_k^{(i-1)} - x_{i-1} + x_{i+N-1} \right) e^{\frac{j2\pi k}{N}}$$ SDFT!

Sliding DFT: Limitation of the SDFT

The SDFT does not allow the use of a window function, generally incorporated in the computation of the DFT, as it would break its sliding property.

DFT of the windowed signal

$$\overset{w}{X}_k = \sum_{n=0}^{N-1} w_n x_n e^{\frac{-j2\pi nk}{N}}$$

$$0 \leq k < N$$

Window of *N* samples

DFT of windowed *x* starting at *i*

$$\overset{w(i)}{X}_k \neq \left(\overset{w(i-1)}{X}_k - x_{i-1} + x_{i+N-1}\right) e^{\frac{j2\pi k}{N}}$$

$$0 \leq k < N$$

DFT of windowed *x* starting at *i-1*

Kernel Windowing: Parseval's theorem

Parseval's theorem basically shows that the dot product between two time-domain sequences is equal to the dot product of their frequency-domain transforms.

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

Sequence 1

Complex conjugate of sequence 2

DFT of sequence 1

Complex conjugate of DFT of sequence 2

Kernel Windowing: Derivation of the SDFT-KW

Parseval's theorem can be used to translate the DFT of a windowed signal into the DFT of the signal, multiplied by a kernel derived from the window function.

$$\underset{0 \le k < N}{X_k^{w(i)}} = \sum_{n=0}^{N-1} \underbrace{x_{i+n} w_n e^{\frac{-j2\pi nk}{N}}}_{y_n}$$

DFT of windowed *x* starting at *i*

$$= \sum_{k'=0}^{N-1} X_{k'}^{(i)} \underbrace{K_{k,k'}}_{\frac{1}{N}\overline{Y_{k'}}}$$

Use Parseval's theorem to get the DFT of x and a kernel

$$= \sum_{k'=0}^{N-1} \left[\underbrace{\left(X_{k'}^{(i-1)} - x_{i-1} + x_{i+N-1} \right) e^{\frac{j2\pi k'}{N}}}_{\text{SDFT}} \right] \underbrace{K_{k,k'}}_{\text{Kernel}}$$

SDFT-KW

Kernel Windowing: Signal-independent kernel

The kernel does not depend on the signal but solely on the window function, which means it only needs to be computed once, before the SDFT process.

$$K_{k,\,k'} = \frac{1}{N}\overline{Y_{k'}} = \frac{1}{N}\underbrace{\overline{\sum_{n=0}^{N-1} y_n e^{\frac{-j2\pi nk'}{N}}}}_{}$$

$0 \le k < N$

$0 \le k' < N$

Complex conjugate DFT

$$= \frac{1}{N}\overline{\sum_{n=0}^{N-1} \underbrace{w_n e^{\frac{-j2\pi nk}{N}}}_{} e^{\frac{-j2\pi nk'}{N}}}$$

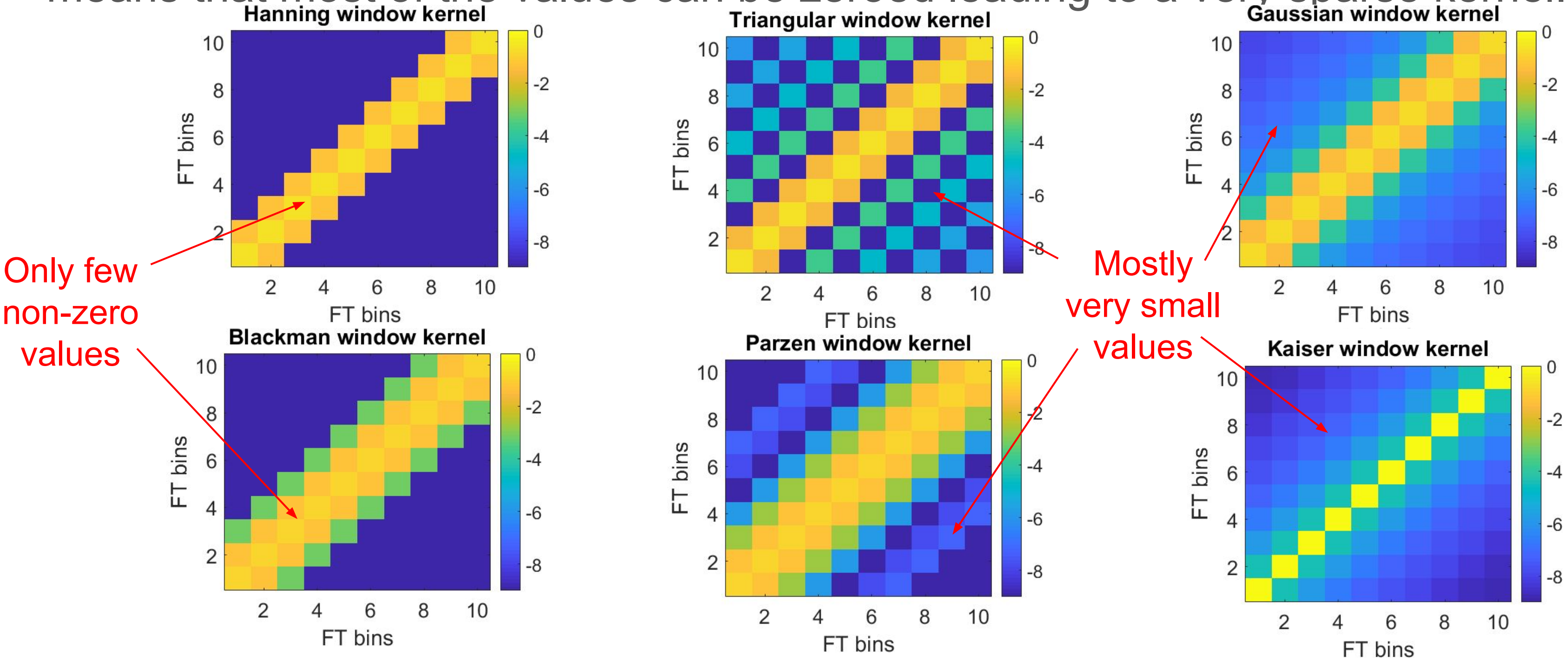
Replace with the window and phase

$$= \underbrace{\frac{1}{N}\sum_{n=0}^{N-1} w_n e^{\frac{j2\pi n(k'-k)}{N}}}_{}$$

Derive the final DFT kernel

Kernel Windowing: Sparse kernels

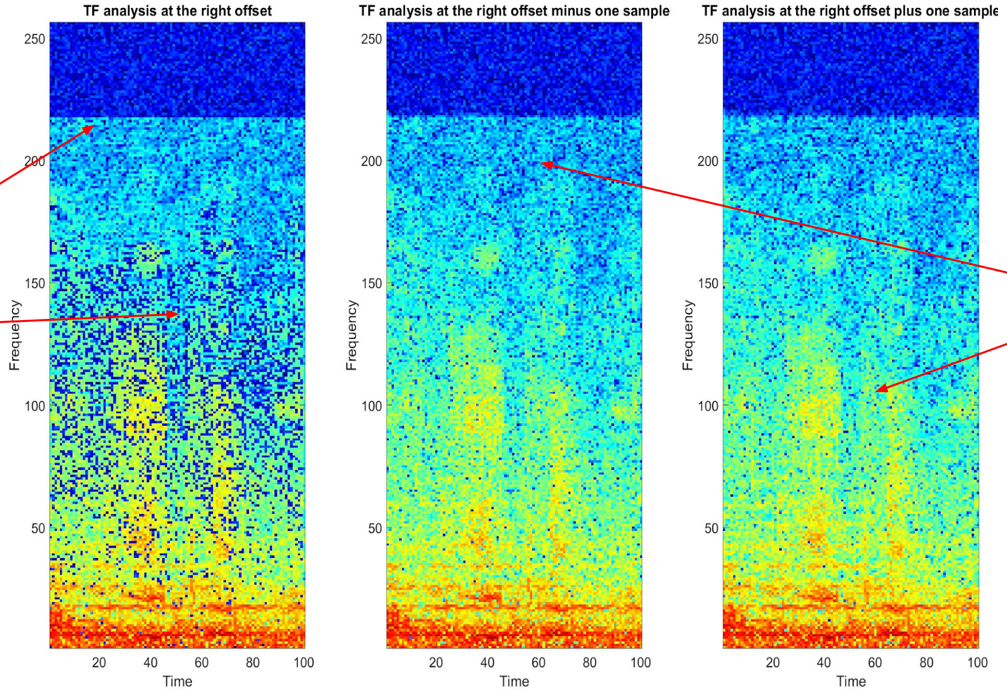
The kernel typically only has a very small number of significant values, which means that most of the values can be zeroed leading to a very sparse kernel.



Application: Framing detection

Lossy coding (MP3, Vorbis, AC-3, etc.) leaves traces of compression which can be detected by using the same parameters and framing used during the encoding.

Encoding analysis with the right parameters and RIGHT framing



Encoding analysis with the right parameters and WRONG framing

Application: Modified discrete cosine transform

Lossy encoding algorithms typically use a transform based on the MDCT, with a variety of window lengths and functions, depending on the coding format.

MDCT at frequency index *k*

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

Signal at sample index *n*

Application: Sliding MDCT-KW

A sliding MDCT with kernel windowing can be derived to help perform framing detection more efficiently, without computing a new MDCT at every sample.

$$\underset{0\le k<\frac{N}{2}}{\overset{w}{Y}_{k}} = \sum_{n=0}^{N-1} x_{i+n} \underbrace{w_n \cos\left(\frac{2\pi}{N}\left(n+\frac{1}{2}+\frac{N}{4}\right)\left(k+\frac{1}{2}\right)\right)}_{y_n}$$

MDCT of windowed *x* starting at *i*

$$= \sum_{k'=0}^{N-1} X_{k'}^{(i)} \underbrace{K_{k,\,k'}}_{\frac{1}{N}\overline{Y_{k'}}}$$

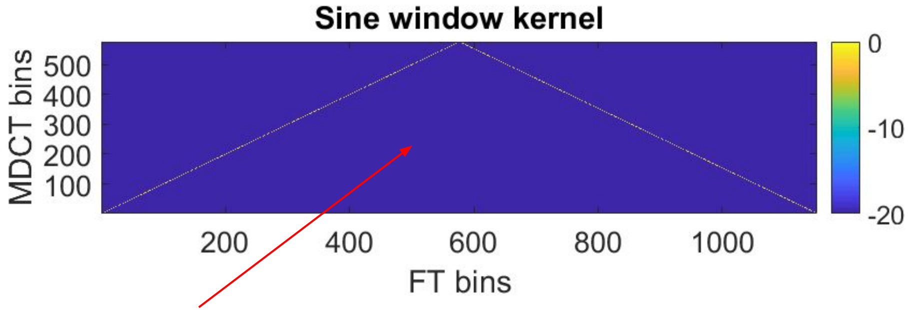
Use Parseval's theorem to get the DFT of x and a kernel

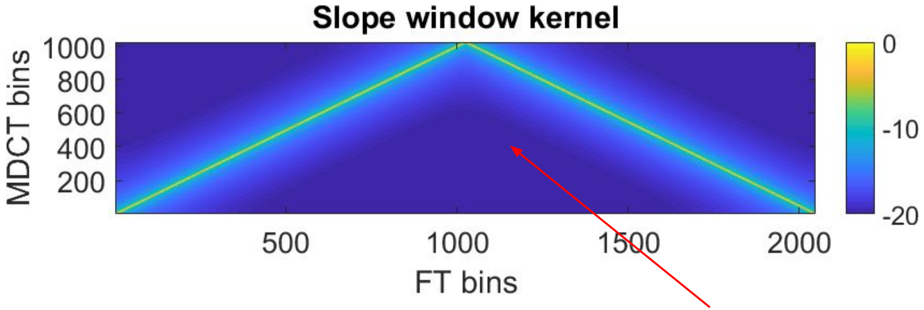
$$= \sum_{k'=0}^{N-1} \Bigg[\underbrace{\left(X_{k'}^{(i-1)} - x_{i-1} + x_{i+N-1}\right) e^{\frac{j\,2\pi k'}{N}}}_{\text{SDFT}} \Bigg] \underbrace{K_{k,\,k'}}_{\text{Kernel}}$$

SMDCT-KW

Application: Independent and sparse SMDCT kernels

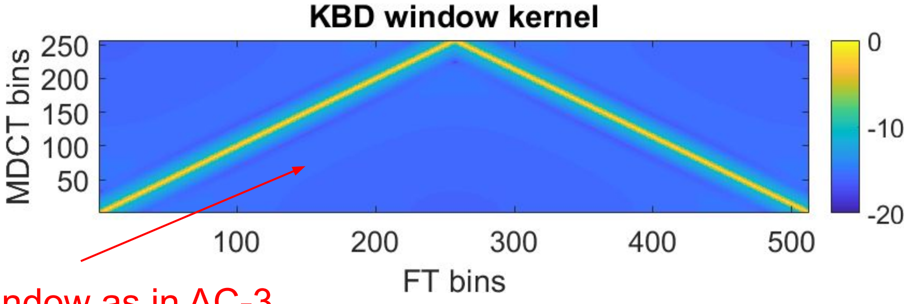
Just like with the kernels derived for the SDFT, the kernels derived for the SMDCT will also be signal-independent and can be made very sparse.





Sine window as in MP3

Slope window as in Vorbis



KBD window as in AC-3

Analysis: Sparsification errors

Window functions	Window length (N)	Errors (T=0.01)	Nonzero values (K)
Triangular	2048	0.049	5
Parzen	2048	0.009	5
Gaussian (α=2.5)	2048	0.020	5
Kaiser (β=0.5)	2048	0.015	3
Sine (MP3)	1152	0.000	2
Slope (Vorbis)	2048	0.022	6
KBD (AC-3)	512	0.013	6

Analysis: Computational complexity

	#Additions	#Multiplications	Complexity
DFT	$N\ (N-1)$	N^2	$O(N^2)$
FFT	$N \log_2(N)$	$(N/2) \log_2(N)$	$O(N \log N)$
SDFT	$2N$	N	$O(N)$
SDFT-KW	$2KN$	KN	$O(N)$
MDCT	$N \log_2(N)$	$N+(N/2)\log_2(N)+N/2$	$O(N \log N)$
SMDCT	$2N$	$N+N+N/2$	$O(N)$
SMDCT-KW	$2KN$	$N+KN+N/2$	$O(N)$

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.

Arigato Gozaimasu!

- Website:http://www.zafarrafii.com/
- GitHub: https://github.com/zafarrafii
- SF-BISH Bash: https://www.meetup.com/bishbash/
- BISH Bash YouTube channel: look for “bish bash meetup”