Time-Frequency Representations for Music Source Separation

Final project presentation

Sevag Hanssian

MUMT 622, Winter 2021

April 22, 2021

2021-04-20

TF representations for music separation

Time-Frequency Representations for Music Source Separation
Final project presentation
Sevag Hanssian
MUMT 622, Winter 2021
April 22, 2021

Music Source Separation

Musical sources are often categorized as either predominantly harmonic, predominantly percussive, or as singing voice.[1] In this project, I consider both cases (HPSS and harmonic/percussive/vocal)

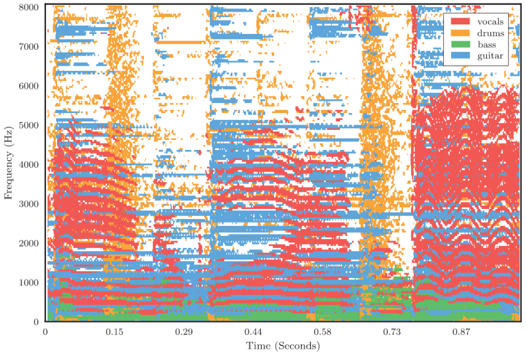
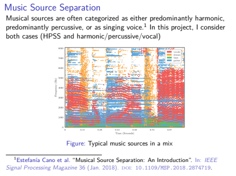


Figure: Typical music sources in a mix

[1] Estefanía Cano et al. "Musical Source Separation: An Introduction". In: *IEEE Signal Processing Magazine* 36 (Jan. 2018). DOI: 10.1109/MSP.2018.2874719.



- steady-state/transient
- tonal/transient in ltfat world

Music Source Separation

A notable property of musical sources is that they are typically *sparse* in the sense that for the majority of points in time and frequency, the sources have very little energy present[2]

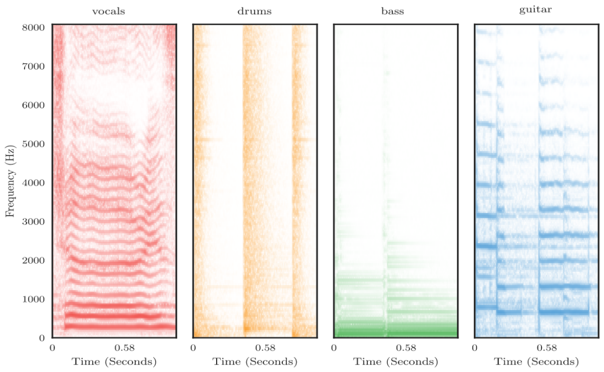
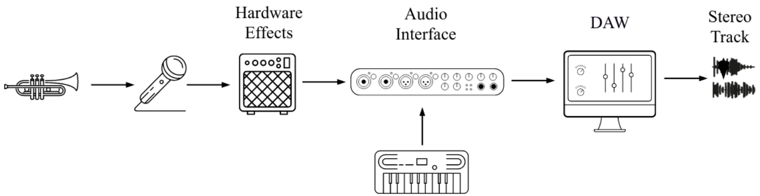


Figure: Typical music sources, separated

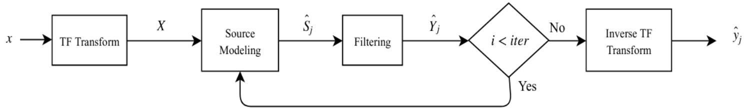
[2]Cano et al., 2018.

Music Source Separation
A notable property of musical sources is that they are typically *sparse* in the sense that for the majority of points in time and frequency, the sources have very little energy present[2]
Figure: Typical music sources, separated
[2]Cano et al., 2018.

Music Source Separation



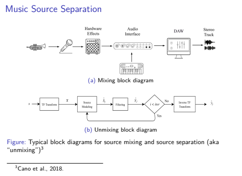
(a) Mixing block diagram



(b) Unmixing block diagram

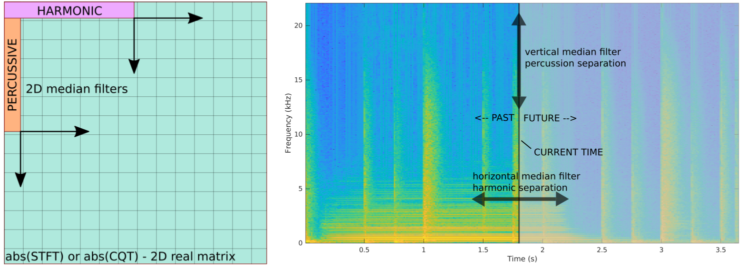
Figure: Typical block diagrams for source mixing and source separation (aka "unmixing")[3]

[3] Cano et al., 2018.



Median filtering HPSS

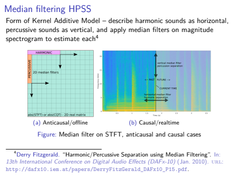
Form of Kernel Additive Model – describe harmonic sounds as horizontal, percussive sounds as vertical, and apply median filters on magnitude spectrogram to estimate each[4]



(a) Anticausal/offline (b) Causal/realtime

Figure: Median filter on STFT, anticausal and causal cases

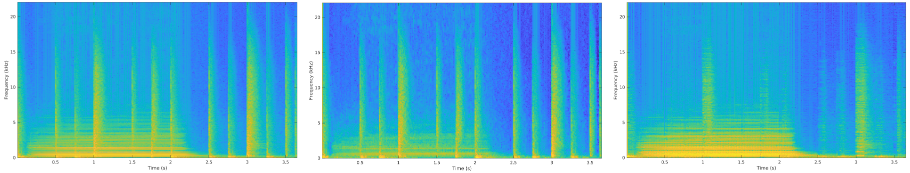
[4] Derry Fitzgerald. "Harmonic/Percussive Separation using Median Filtering". In: *13th International Conference on Digital Audio Effects (DAFx-10)* (Jan. 2010). URL: http://dafx10.iem.at/papers/DerryFitzGerald_DAFx10_P15.pdf.



- sliding STFT, perform causal median filtering, then invert
- keep several columns of STFT in memory, perform windowing + overlap-add – presented this in 501

Median filtering HPSS

Use harmonic and percussive magnitude spectrogram estimates to compute soft masks[5] or hard masks:[6]



(a) Mixed signal (b) Percussive separation (c) Harmonic separation

Figure: Example of median filtering HPSS

Originally based on STFT spectrogram. CQT works fine too.

---

[5] Fitzgerald, 2010.

[6] Jonathan Driedger, Meinard Müller, and Sascha Disch. *Extending Harmonic-Percussive Separation of Audio Signals*. Jan. 2014. URL: https://www.audiolabs-erlangen.de/content/05-fau/assistant/00-driedger/01-publications/2014_DriedgerMuellerDisch_ExtensionsHPSeparation_ISMIR.pdf.

2021-04-20

└Median filtering HPSS

Median filtering HPSS

Use harmonic and percussive magnitude spectrogram estimates to compute soft masks[5] or hard masks:[6]

(a) Mixed signal (b) Percussive separation (c) Harmonic separation

Figure: Example of median filtering HPSS

Originally based on STFT spectrogram. CQT works fine too.

[5] Fitzgerald, 2010.

[6] Jonathan Driedger, Meinard Müller, and Sascha Disch. *Extending Harmonic-Percussive Separation of Audio Signals*. Jan. 2014. URL: https://www.audiolabs-erlangen.de/content/05-fau/assistant/00-driedger/01-publications/2014_DriedgerMuellerDisch_ExtensionsHPSeparation_ISMIR.pdf.

Median filtering HPSS

Soft mask, Wiener filter:[7]

$$M_{\text{harmonic}} = \frac{|\hat{S}_{\text{harmonic}}|^2}{|\hat{S}_{\text{harmonic}}|^2 + |\hat{S}_{\text{percussive}}|^2}$$

Hard mask, binary mask:[8]

$$M_{\text{harmonic}} = \frac{|\hat{S}_{\text{percussive}}|}{|\hat{S}_{\text{harmonic}}| + \epsilon} > \beta$$

Setting $\beta > 1.0$ leads to a third residual component:

$$M_{\text{residual}} = 1 - (M_{\text{harmonic}} + M_{\text{percussive}})$$

Soft mask gives higher audio quality[9]

[7] Fitzgerald, 2010.
[8] Driedger, Müller, and Disch, 2014.
[9] Gerkmann and Vincent, 2018.

Median filtering HPSS
Soft mask, Wiener filter:[7]
$$M_{\text{harmonic}} = \frac{|\hat{S}_{\text{harmonic}}|^2}{|\hat{S}_{\text{harmonic}}|^2 + |\hat{S}_{\text{percussive}}|^2}$$
Hard mask, binary mask:[8]
$$M_{\text{harmonic}} = \frac{|\hat{S}_{\text{percussive}}|}{|\hat{S}_{\text{harmonic}}| + \epsilon} > \beta$$
Setting $\beta > 1.0$ leads to a third residual component:
$$M_{\text{residual}} = 1 - (M_{\text{harmonic}} + M_{\text{percussive}})$$
Soft mask gives higher audio quality[9]
[7] Fitzgerald, 2010.
[8] Driedger, Müller, and Disch, 2014.
[9] Gerkmann and Vincent, 2018.

- define how different harmonic and percussive should be

Iterative median filtering HPSS

2-pass HPSS,[10] , harmonic/percussive/vocal:[11]

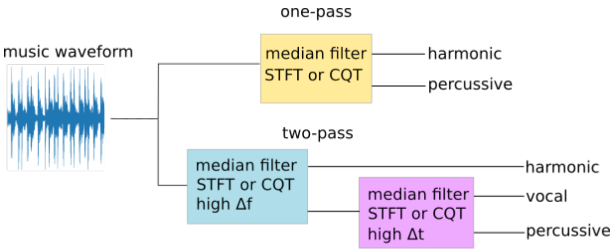
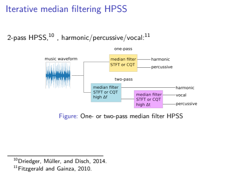


Figure: One- or two-pass median filter HPSS

[10] Driedger, Müller, and Disch, 2014.
[11] Fitzgerald and Gainza, 2010.



HPSS MATLAB pseudocode

$s =$ mixed audio
$\hat{S} = \text{STFT}(s)$ **or CQT**
$S = \text{abs}(\hat{S})$
$H = \text{medianfilter}(S, l_H, \text{axis} = 2)$
$P = \text{medianfilter}(S, l_P, \text{axis} = 1)$
soft $M_H = \frac{H^p}{H^p + P^p}, M_P = \frac{P^p}{H^p + P^p}$
hard $M_H = \frac{H}{P+\epsilon} \geq \beta, M_P = \frac{P}{H+\epsilon} > \beta$
$\hat{H} = \hat{S} \cdot M_H$
$\hat{P} = \hat{S} \cdot M_P$
$h = \text{ISTFT}(\hat{H})$ **or ICQT**
$p = \text{ISTFT}(\hat{P})$ **or ICQT**

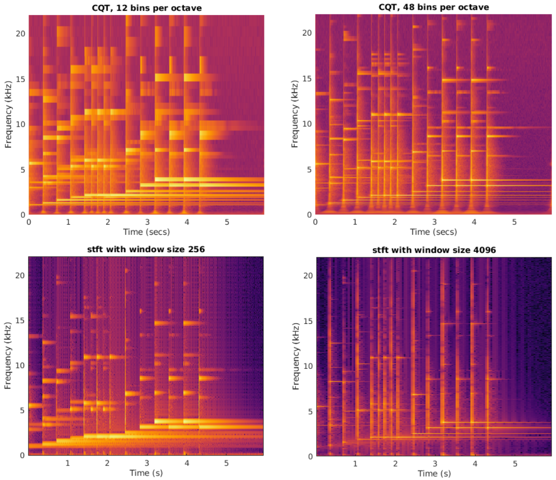
$s =$ mixed audio
$\hat{S}1 = \text{STFT}(s)$ **or CQT**
1-pass algorithm $\rightarrow \hat{H}1, \hat{P}1$
final harmonic $h1 = \text{ISTFT}(\hat{H}1)$ **or ICQT**
$p1 = \text{ISTFT}(\hat{P}1)$ **or ICQT**
$\hat{S}2 = \text{STFT}(p1)$ **or CQT**
1-pass algorithm $\rightarrow \hat{H}2, \hat{P}2$
$h2 = \text{ISTFT}(\hat{H}2)$ **or ICQT**
final percussive $p1 = \text{ISTFT}(\hat{P}2)$ **or ICQT**

Listing 1: 1- and 2-pass median filtering HPSS algorithms

HPSS MATLAB pseudocode
Listing 1: 1- and 2-pass median filtering HPSS algorithms

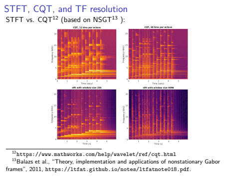
STFT, CQT, and TF resolution

STFT vs. CQT[12] (based on NSGT[13]):



[12]https://www.mathworks.com/help/wavelet/ref/cqt.html

[13]Balazs et al., "Theory, implementation and applications of nonstationary Gabor frames", 2011, https://ltfat.github.io/notes/ltfatnote018.pdf.



- glockenspiel = default file of ltfat, the canonical sound for tonal/transient
- loaded by gspi function
- look how good the cqt is – original motivation for this project, can we take the simple median filter algorithm and make it better with a better tf representation

Sparsity, entropy, and two window sizes

Sparsity and entropy are complementary concepts:[14]

- *Sparsity* is the property of concentrating most of the energy of x in few coefficients of w
- *Entropy* is the property of not concentrating most of the probability mass in few atoms of p – in other words, the entropy of a random variable is a concept of information theory that characterizes the unpredictability inherent in its outcomes

In every algorithm shown, the common element is a 2 dictionary wide + narrow window analysis, to represent tonal and transient parts of the input signal sparsely – or, to represent tonal and transient parts of the input signal with low entropy/high significance

[14] Paul Honeine. *Entropy of Overcomplete Kernel Dictionaries*. 2014. arXiv: 1411.0161 [cs.IT]. URL: https://arxiv.org/pdf/1411.0161.pdf; Giancarlo Pastor et al. *Mathematics of Sparsity and Entropy: Axioms, Core Functions and Sparse Recovery*. 2015. arXiv: 1501.05126 [cs.IT].

2021-04-20

TF representations for music separation

└─Sparsity, entropy, and two window sizes

Sparsity, entropy, and two window sizes

Sparsity and entropy are complementary concepts:[14]

- *Sparsity* is the property of concentrating most of the energy of x in few coefficients of w
- *Entropy* is the property of not concentrating most of the probability mass in few atoms of p – in other words, the entropy of a random variable is a concept of information theory that characterizes the unpredictability inherent in its outcomes

In every algorithm shown, the common element is a 2 dictionary wide + narrow window analysis, to represent tonal and transient parts of the input signal sparsely – or, to represent tonal and transient parts of the input signal with low entropy/high significance

[14] Paul Honeine. *Entropy of Overcomplete Kernel Dictionaries*. 2014. arXiv: 1411.0161 [cs.IT]. URL: https://arxiv.org/pdf/1411.0161.pdf; Giancarlo Pastor et al. *Mathematics of Sparsity and Entropy: Axioms, Core Functions and Sparse Recovery*. 2015. arXiv: 1501.05126 [cs.IT].

Structured sparsity

Group-LASSO:[15] Lasso shrinkage (aka linear least squares regression[16] [17]) to the transform coefficients in the time and frequency dimensions

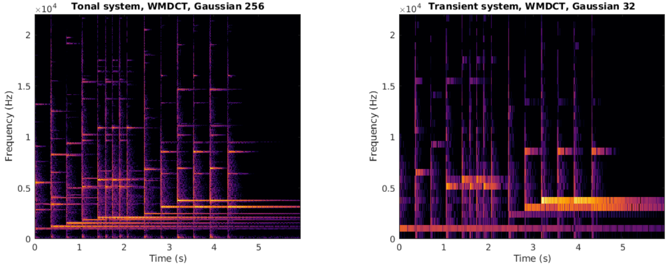
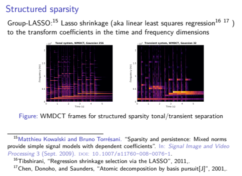


Figure: WMDCT frames for structured sparsity tonal/transient separation

[15] Matthieu Kowalski and Bruno Torrésani. "Sparsity and persistence: Mixed norms provide simple signal models with dependent coefficients". In: *Signal Image and Video Processing* 3 (Sept. 2009). DOI: 10.1007/s11760-008-0076-1.

[16] Tibshirani, "Regression shrinkage selection via the LASSO", 2011,.

[17] Chen, Donoho, and Saunders, "Atomic decomposition by basis pursuit[J]", 2001,.



Structured sparsity

WMDCT[18] + Group-LASSO – "audioshrink"[19]

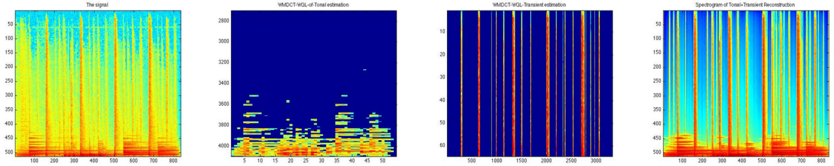


Figure: Audioshrink for tonal/transient separation in jazz music

Use 2 WMDCT transforms (wide + narrow window) + Group-LASSO to shrink input signal into significant coefficients in "time" and "frequency" groups

[18] Kai Siedenburg and Monika Doerfler. "Structured sparsity for audio signals". In: *Proc. of the 14th International Conference on Digital Audio Effects (DAFx-11), Paris, France*. Jan. 2011.

[19] `https://homepage.univie.ac.at/monika.doerfler/StrucAudio.html`, `https://ltfat.github.io/doc/demos/demo_audioshrink.html`

Structured sparsity
WMDCT[18] + Group-LASSO – "audioshrink"[19]
Figure: Audioshrink for tonal/transient separation in jazz music
Use 2 WMDCT transforms (wide + narrow window) + Group-LASSO to shrink input signal into significant coefficients in "time" and "frequency" groups
[18] Kai Siedenburg and Monika Doerfler. "Structured sparsity for audio signals". In: Proc. of the 14th International Conference on Digital Audio Effects (DAFx-11), Paris, France. Jan. 2011.
[19] https://homepage.univie.ac.at/monika.doerfler/StrucAudio.html, https://ltfat.github.io/doc/demos/demo_audioshrink.html

Audioshrink MATLAB pseudocode

f = mixed audio
$F1 = \text{frametight}(\text{frame}(\text{wmdct}, \text{gauss}, \text{winsize}_h))$ **WMDCT**
$F2 = \text{frametight}(\text{frame}(\text{wmdct}, \text{gauss}, \text{winsize}_p))$
$c1 = \text{franagrouplasso}(F1, f, \lambda_h, \text{soft}, \text{freq})$
$c2 = \text{franagrouplasso}(F2, f, \lambda_p, \text{soft}, \text{time})$
$xh = \text{frsyn}(F1, c1)$ **Inverse WMDCT**
$xp = \text{frsyn}(F2, c2)$

Listing 2: WMDCTLasso tonal/transient separation algorithm

franagrouplasso in LTFAT solves the Group-LASSO regression problem in the time-frequency domain
Soft thresholding vs. hard thresholding – similar to soft/hard masking

Audioshrink MATLAB pseudocode
f = mixed audio
$F1 = \text{frametight}(\text{frame}(\text{wmdct}, \text{gauss}, \text{winsize}_h))$ **WMDCT**
$F2 = \text{frametight}(\text{frame}(\text{wmdct}, \text{gauss}, \text{winsize}_p))$
$c1 = \text{franagrouplasso}(F1, f, \lambda_h, \text{soft}, \text{freq})$
$c2 = \text{franagrouplasso}(F2, f, \lambda_p, \text{soft}, \text{time})$
$xh = \text{frsyn}(F1, c1)$ **Inverse WMDCT**
$xp = \text{frsyn}(F2, c2)$
Listing 2: WMDCTLasso tonal/transient separation algorithm
franagrouplasso in LTFAT solves the Group-LASSO regression problem in the time-frequency domain
Soft thresholding vs. hard thresholding – similar to soft/hard masking

Rényi entropy vs. Lasso

- Using Lasso-like methods ($l1, 2$ norm in linear least squares regression) assumes that the underlying systems are linear Gaussian, or approximately so[20]
- If the system is not linear-Gaussian, linear least squares regression can lead to suboptimal results
- Rényi entropy, which is an adaptation of Shannon entropy[21] is a measure of information in signal processing, and can be used as an optimization target (i.e., loss function) without imposing restrictions on the system being optimized

[20] Ed Beadle et al. "An Overview of Renyi Entropy and Some Potential Applications". In: Nov. 2008, pp. 1698–1704. DOI: 10.1109/ACSSC.2008.5074715.

[21] A. Rényi. "On Measures of Entropy and Information". In: *Proceedings IV Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, 20-30 June 1961* 1 (1961), pp. 547–561; C. E. Shannon. "A mathematical theory of communication". In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

Rényi entropy vs. Lasso

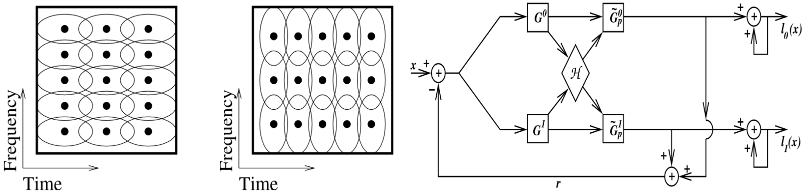
- Using Lasso-like methods ($l1, 2$ norm in linear least squares regression) assumes that the underlying systems are linear Gaussian, or approximately so[20]
- If the system is not linear-Gaussian, linear least squares regression can lead to suboptimal results
- Rényi entropy, which is an adaptation of Shannon entropy[21] is a measure of information in signal processing, and can be used as an optimization target (i.e., loss function) without imposing restrictions on the system being optimized

[20] Ed Beadle et al. "An Overview of Renyi Entropy and Some Potential Applications". In: Nov. 2008, pp. 1698–1704. DOI: 10.1109/ACSSC.2008.5074715.

[21] A. Rényi. "On Measures of Entropy and Information". In: *Proceedings IV Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, 20-30 June 1961* 1 (1961), pp. 547–561; C. E. Shannon. "A mathematical theory of communication". In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

Time-Frequency Jigsaw Puzzle

1. Create time-frequency "super-tiles" by superimposing a large window + small window Gabor analysis
2. Use Rényi entropy to set coefficients to zero where sound has more entropy than random white noise



(a) TF supertiles (b) Entropy criterion

Figure: TF Jigsaw Puzzle tonal/transient separation[22]

[22] Florent Jaillet and Bruno Torrésani. "Time-Frequency Jigsaw Puzzle: adaptive multiwindow and multilayered Gabor expansions". In: *International Journal of Wavelets, Multiresolution and Information Processing* 05 (Mar. 2007). DOI: 10.1142/S0219691307001768.

Time-Frequency Jigsaw Puzzle
1. Create time-frequency "super-tiles" by superimposing a large window + small window Gabor analysis
2. Use Rényi entropy to set coefficients to zero where sound has more entropy than random white noise
(a) TF supertiles (b) Entropy criterion
Figure: TF Jigsaw Puzzle tonal/transient separation[22]
[22] Florent Jaillet and Bruno Torrésani. "Time-Frequency Jigsaw Puzzle: adaptive multiwindow and multilayered Gabor expansions". In: International Journal of Wavelets, Multiresolution and Information Processing 05 (Mar. 2007). DOI: 10.1142/S0219691307001768.

- i.e. good tonal/good transient
- high entropy = indicating sound is poorly represented

TFJigsaw

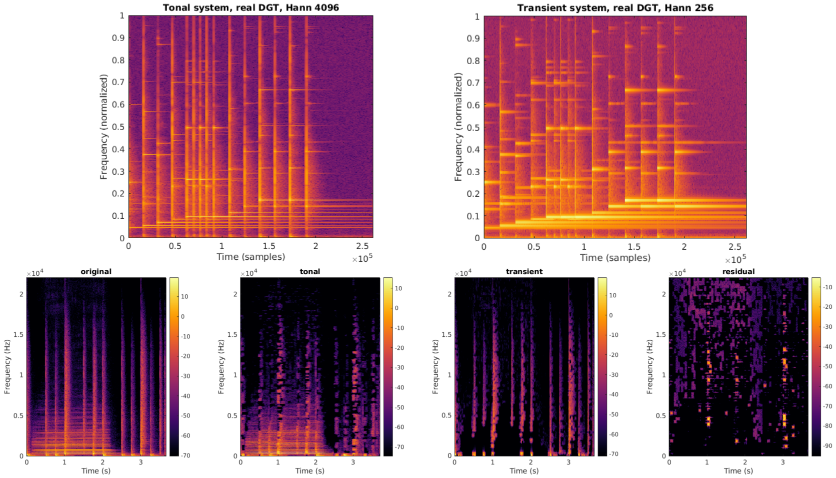
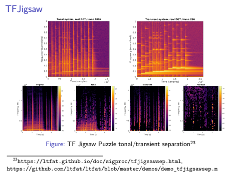


Figure: TF Jigsaw Puzzle tonal/transient separation[23]

[23] https://ltfat.github.io/doc/sigproc/tfjigsawsep.html, https://github.com/ltfat/ltfat/blob/master/demos/demo_tfjigsawsep.m



- Hann DGT = stft, basically

TFJigsaw MATLAB pseudocode

f = mixed audio
a, M, winsize, $b\{1,2\}$ = Gabor systems 1 and 2 configuration
$r\{1,2\}$ = significance level of tonal and transient layer re: white noise ref
$[\text{ref}1, \text{ref}2]$ =
↪ generate estimate of random white noise entropy within supertile
$[\text{tau}1, \text{tau}2] = [\text{ref}1 \cdot r1, \text{ref}2 \cdot r2]$
$c1 = \text{DGTReal}(f, \text{winsize}1, a1, M1)$ **Discrete Gabor Transform**
$c2 = \text{DGTReal}(f, \text{winsize}2, a2, M2)$
for all time and frequency supertiles
    $f\{1,2\}$ = frequency supertile location, Gabor system 1,2
    $t\{1,2\}$ = time supertile location, Gabor system 1,2
    $[c1, c2] = \text{decision}(c1, c2, f1, f2, t1, t2, \text{tau}1, \text{tau}2)$
endfor
$f_{\text{tonal}} = \text{IDGTReal}(c1)$ **Inverse discrete Gabor Transform**
$f_{\text{transient}} = \text{IDGTReal}(c2)$

Listing 3: TFJigsaw tonal/transient separation algorithm

TFJigsaw MATLAB pseudocode
f = mixed audio
a, M, winsize, $b\{1,2\}$ = Gabor systems 1 and 2 configuration
$r\{1,2\}$ = significance level of tonal and transient layer re: white noise ref
$[\text{ref}1, \text{ref}2]$ =
↪ generate estimate of random white noise entropy within supertile
$[\text{tau}1, \text{tau}2] = [\text{ref}1 \cdot r1, \text{ref}2 \cdot r2]$
$c1 = \text{DGTReal}(f, \text{winsize}1, a1, M1)$ Discrete Gabor Transform
$c2 = \text{DGTReal}(f, \text{winsize}2, a2, M2)$
for all time and frequency supertiles
$f\{1,2\}$ = frequency supertile location, Gabor system 1,2
$t\{1,2\}$ = time supertile location, Gabor system 1,2
$[c1, c2] = \text{decision}(c1, c2, f1, f2, t1, t2, \text{tau}1, \text{tau}2)$
endfor
$f_{\text{tonal}} = \text{IDGTReal}(c1)$ Inverse discrete Gabor Transform
$f_{\text{transient}} = \text{IDGTReal}(c2)$
Listing 3: TFJigsaw tonal/transient separation algorithm

Evaluation testbench

Inspired by SigSep[24], SISEC (Signal Separation Evaluation Campaign):

- BSS[25] (BSSv4 variant[26]) and PEASS[27] (MATLAB toolkit[28]). BSS vs. PEASS?[29] BSSv4 is used widely in modern literature,[30] but perceptual measures are important! Use PEASS
- Testing files: MUSDB18-HQ [31]
- MATLAB/Python testbench using file system + JSON interchanges
- Open-Unmix[32] as a reference, open, near-SOTA neural solution
- Compare different configurations of each algorithm in "group stages," winners move to next stage and may be combined in hybrid algorithms

[24] `https://sigsep.github.io/`
[25] Vincent, Gribonval, and Févotte, 2006.
[26] `https://github.com/sigsep/bsseval`
[27] Emiya et al., 2011.
[28] `http://bass-db.gforge.inria.fr/peass/`
[29] Ward et al., 2018.
[30] Stöter, Liutkus, and Ito, 2018.
[31] Rafii et al., 2019.
[32] Stöter et al., 2019.

Evaluation testbench
Inspired by SigSep[24], SISEC (Signal Separation Evaluation Campaign):
- BSS[25] (BSSv4 variant[26]) and PEASS[27] (MATLAB toolkit[28]). BSS vs. PEASS?[29] BSSv4 is used widely in modern literature,[30] but perceptual measures are important! Use PEASS
- Testing files: MUSDB18-HQ [31]
- MATLAB/Python testbench using file system + JSON interchanges
- Open-Unmix[32] as a reference, open, near-SOTA neural solution
- Compare different configurations of each algorithm in "group stages," winners move to next stage and may be combined in hybrid algorithms
[24] https://sigsep.github.io/
[25] Vincent, Gribonval, and Févotte, 2006.
[26] https://github.com/sigsep/bsseval
[27] Emiya et al., 2011.
[28] http://bass-db.gforge.inria.fr/peass/
[29] Ward et al., 2018.
[30] Stöter, Liutkus, and Ito, 2018.
[31] Rafii et al., 2019.
[32] Stöter et al., 2019.

- global score omitted. target, interference, artifact

HPSS – PEASS results

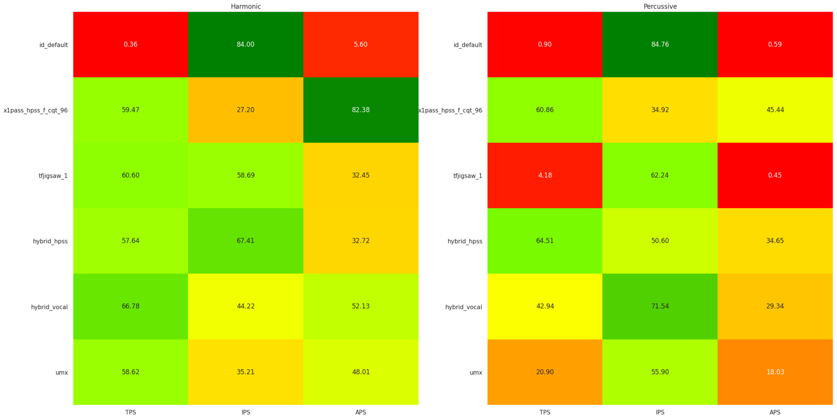
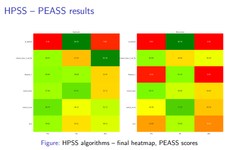


Figure: HPSS algorithms – final heatmap, PEASS scores



HPSS – BSSv4 results

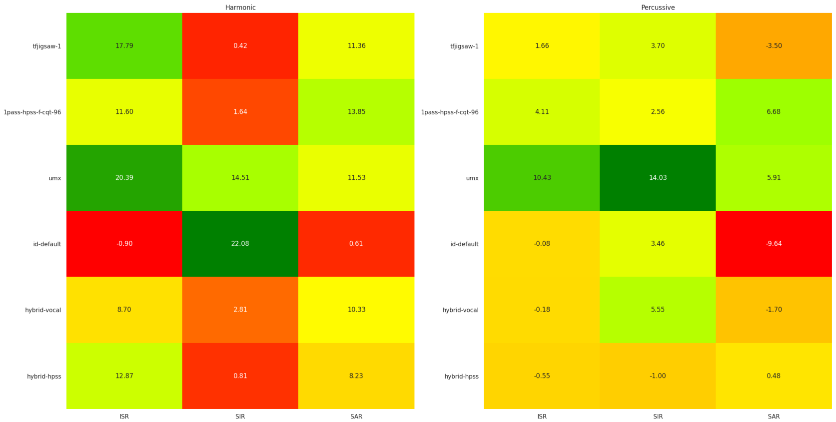
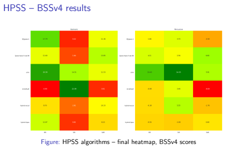


Figure: HPSS algorithms – final heatmap, BSSv4 scores

2021-04-20

TF representations for music separation

└HPSS – BSSv4 results



HPSS + vocal – PEASS results

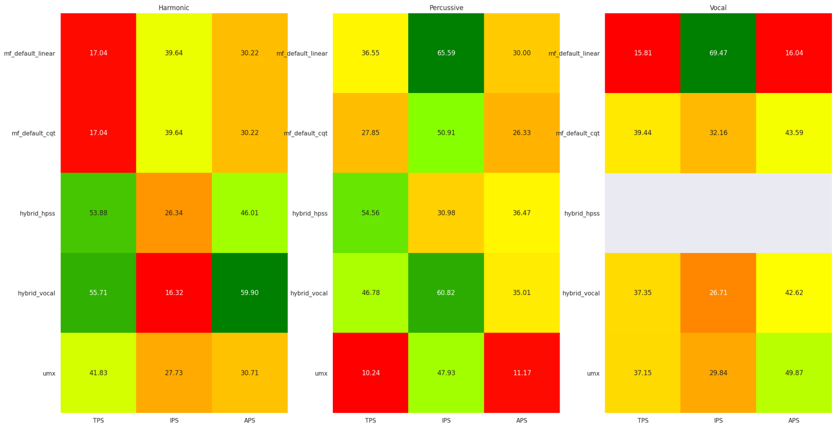
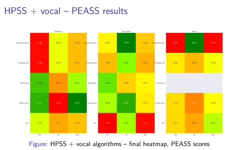


Figure: HPSS + vocal algorithms – final heatmap, PEASS scores

2021-04-20

TF representations for music separation

└HPSS + vocal – PEASS results



HPSS + vocal – BSSv4 results

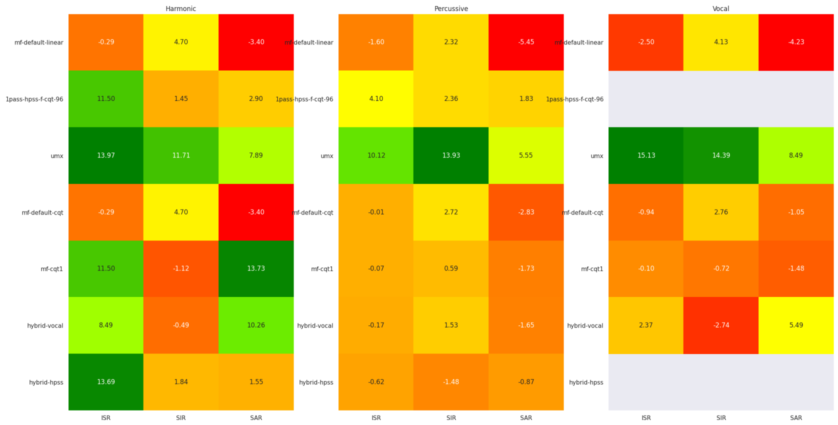


Figure: HPSS + vocal algorithms – final heatmap, BSSv4 scores

2021-04-20

TF representations for music separation

└HPSS + vocal – BSSv4 results

HPSS + vocal – BSSv4 results

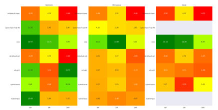
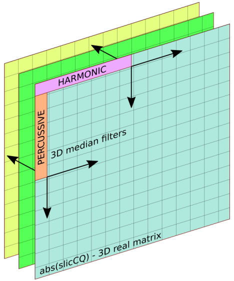


Figure: HPSS + vocal algorithms – final heatmap, BSSv4 scores

Best 1-pass HPSS – CQT-96, sliCQ median filter

Best performing 1-pass offline/anticausal algorithm: CQT with 96 bins-per-octave + median filter + **soft** mask. Realtime/causal: sliCQ[33] with 12 bins-per-octave + median filter + **hard** mask



Realtime STFT:[34] input stream = hop size, $2 \times$ hop ringbuffer (window).
sliCQ transform: input stream = trlen, $4 \times$ trlen ringbuffer (sllen)

[33] Nicki Holighaus et al. "A Framework for Invertible, Real-Time Constant-Q Transforms". In: *IEEE Transactions on Audio, Speech and Language Processing* 21 (Sept. 2012). DOI: 10.1109/TASL.2012.2234114. URL: https://arxiv.org/pdf/1210.0084.pdf.

[34] https://github.com/sevagh/Real-Time-HPSS

Best 1-pass HPSS – CQT-96, sliCQ median filter
Best performing 1-pass offline/anticausal algorithm: CQT with 96 bins-per-octave + median filter + soft mask. Realtime/causal: sliCQ[33] with 12 bins-per-octave + median filter + hard mask
Realtime STFT:[34] input stream = hop size, 2 × hop ringbuffer (window).
sliCQ transform: input stream = trlen, 4 × trlen ringbuffer (sllen)
[33] Nicki Holighaus et al. "A Framework for Invertible, Real-Time Constant-Q Transforms". In: IEEE Transactions on Audio, Speech and Language Processing 21 (Sept. 2012). DOI: 10.1109/TASL.2012.2234114. URL: https://arxiv.org/pdf/1210.0084.pdf.
[34] https://github.com/sevagh/Real-Time-HPSS

- 3D sliCQ coefficients can be median filtered, analogous to RGB/RGBA images

Best 2-pass HPSS – hybrid

Hybrid HPSS: combine TFJigsaw and STFT median filtering

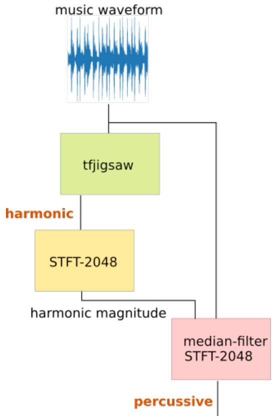


Figure: Block diagram of hybrid HPSS algorithm

2021-04-20

TF representations for music separation

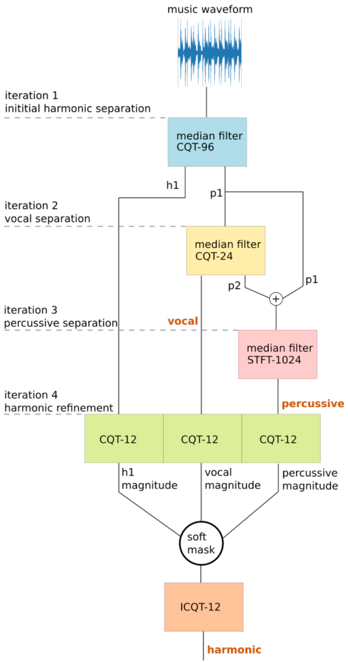
└Best 2-pass HPSS – hybrid

Best 2-pass HPSS – hybrid

Hybrid HPSS: combine TFJigsaw and STFT median filtering

Figure: Block diagram of hybrid HPSS algorithm

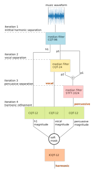
Multipass hybrid vocal



2021-04-20

TF representations for music separation

Multipass hybrid vocal



HPSS – audio clips

Algorithm	Harmonic	Percussive
Reference mix	h	p
1-pass CQT 96 bins + soft mask [35]	h	p
Realtime sliCQ 12 bins + hard mask	h	p
Iterative Driedger[36]	h	p
Hybrid-HPSS	h	p
UMX	h	p

[35] Fitzgerald, 2010.
[36] Driedger, Müller, and Disch, 2014.

HPSS – audio clips

Algorithm	Harmonic	Percussive
Reference mix	h	p
1-pass CQT 96 bins + soft mask [35]	h	p
Realtime sliCQ 12 bins + hard mask	h	p
Iterative Driedger[36]	h	p
Hybrid-HPSS	h	p
UMX	h	p

[35] Fitzgerald, 2010.
[36] Driedger, Müller, and Disch, 2014.

HPSS + vocal – audio clips

Algorithm	Harmonic	Percussive	Vocal
Reference 🔊 mix	🔊 h	🔊 p	🔊 v
Iterative Fitzgerald, vocal [37]	🔊 h	🔊 p	🔊 v
Iterative Fitzgerald, percussive	🔊 h	🔊 p	🔊 v
Hybrid-Vocal	🔊 h	🔊 p	🔊 v
UMX	🔊 h	🔊 p	🔊 v

[37] Fitzgerald and Gainza, 2010.

2021-04-20

TF representations for music separation

└HPSS + vocal – audio clips

HPSS + vocal – audio clips

Algorithm	Harmonic	Percussive	Vocal
Reference 🔊 mix	🔊 h	🔊 p	🔊 v
Iterative Fitzgerald, vocal [37]	🔊 h	🔊 p	🔊 v
Iterative Fitzgerald, percussive	🔊 h	🔊 p	🔊 v
Hybrid-Vocal	🔊 h	🔊 p	🔊 v
UMX	🔊 h	🔊 p	🔊 v

[37] Fitzgerald and Gainza, 2010.

NSGT in STFT-based neural network

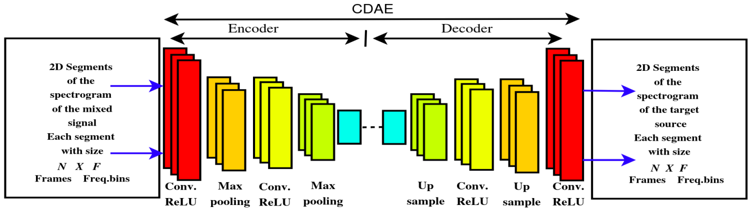


Figure: Convolutional denoising autoencoders[38]

Idea: replace STFT spectrogram with NSGT spectrogram. Both MATLAB Wavelet Toolbox CQT (based on NSGT) and reference Python implementation of NSGT (https://github.com/grrrr/nsgt) can produce a rectangular matrix of TF coefficients, solving a common difficulty of using the CQT in spectrogram-based algorithms.

[38] Emad M. Grais and Mark D. Plumbley. "Single Channel Audio Source Separation using Convolutional Denoising Autoencoders". In: *CoRR* abs/1703.08019 (2017). arXiv: 1703.08019. URL: http://arxiv.org/abs/1703.08019.

NSGT in STFT-based neural network

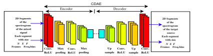


Figure: Convolutional denoising autoencoders[38]

Idea: replace STFT spectrogram with NSGT spectrogram. Both MATLAB Wavelet Toolbox CQT (based on NSGT) and reference Python implementation of NSGT (https://github.com/grrrr/nsgt) can produce a rectangular matrix of TF coefficients, solving a common difficulty of using the CQT in spectrogram-based algorithms.

[38] Emad M. Grais and Mark D. Plumbley. "Single Channel Audio Source Separation using Convolutional Denoising Autoencoders". In: *CoRR* abs/1703.08019 (2017). arXiv: 1703.08019. URL: http://arxiv.org/abs/1703.08019.

NSGT-spectrogram neural network

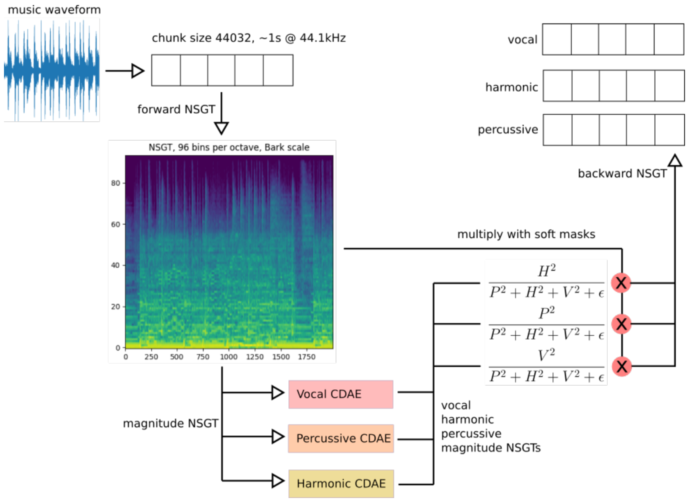
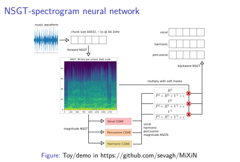


Figure: Toy/demo in https://github.com/sevagh/MiXiN

2021-04-20

└NSGT-spectrogram neural network



Conclusions

- Real-world application of 622 concepts – sparsity, entropy, overcomplete dictionaries, pursuit
- Swap STFT for "better" TF representations in simple algorithms to improve source separation results
- Competitive PEASS separation results in hybrid algorithms based on advanced DSP/time-frequency analysis (not so good in BSSv4)
- Swap STFT for NSGT in a both traditional DSP algorithms, and machine/deep learning networks – lots of future potential

501 to 622:

Algorithm	Harmonic	Percussive
HPSS (MUMT 501) mix	h	p
MiXiN (MUMT 622)	h	p

Conclusions

- Real-world application of 622 concepts – sparsity, entropy, overcomplete dictionaries, pursuit
- Swap STFT for "better" TF representations in simple algorithms to improve source separation results
- Competitive PEASS separation results in hybrid algorithms based on advanced DSP/time-frequency analysis (not so good in BSSv4)
- Swap STFT for NSGT in a both traditional DSP algorithms, and machine/deep learning networks – lots of future potential

501 to 622:

Algorithm	Harmonic	Percussive
HPSS (MUMT 501) mix	h	p
MiXiN (MUMT 622)	h	p