Real-Time Spectrogram Inversion Using Phase Gradient Heap Integration

Zdeněk Průša[1] and Peter L. Søndergaard[2]
[1,2]Acoustics Research Institute, Austrian Academy of Sciences, Vienna, Austria
[2] Oticon A/S, Smørum, Denmark

Overview

1. Motivation
2. Algorithm description
3. Results
4. Live demonstration

Nontrivial modification of spectrogram:

- Time stretching/pitch shifting
- Source separation/denoising
- Missing data inpainting
- Comb filter-free channel mixing

Introduction – STFT

Short-time Fourier transform:

$$(\mathcal{V}_g f)(\omega, t) = \int_{\mathbb{R}} f(\tau + t) g(\tau) e^{-i2\pi\omega\tau} \, d\tau, \quad \omega, t \in \mathbb{R} \tag{1}$$

$$= M_g^f(\omega, t) e^{i\Phi_g^f(\omega, t)}, \tag{2}$$

Clearly:

$$\log(\mathcal{V}_g f)(\omega, t) = \log M_g^f(\omega, t) + i\Phi_g^f(\omega, t). \tag{3}$$

STFT Phase-Magnitude relationship

Gaussian window: $\varphi_\gamma(t) = \left(\frac{\gamma}{2}\right)^{-\frac{1}{4}} \mathrm{e}^{-\pi \frac{t^2}{\gamma}}$

Log-magnitude gradient: $\nabla \log M^f_{\varphi_\gamma}(\omega, t) = \left(\frac{\partial \log M^f_{\varphi_\gamma}}{\partial \omega}(\omega, t), \frac{\partial \log M^f_{\varphi_\gamma}}{\partial t}(\omega, t) \right)$

Phase gradient: $\nabla \Phi^f_{\varphi_\gamma}(\omega, t) = \left(\frac{\partial \Phi^f_{\varphi_\gamma}}{\partial \omega}(\omega, t), \frac{\partial \Phi^f_{\varphi_\gamma}}{\partial t}(\omega, t) \right)$

Relationship between the gradients[1,2,3]

$$\frac{\partial \Phi^f_{\varphi_\gamma}}{\partial \omega}(\omega, t) = -\gamma \frac{\partial}{\partial t} \log M^f_{\varphi_\gamma}(\omega, t) \tag{4}$$

$$\frac{\partial \Phi^f_{\varphi_\gamma}}{\partial t}(\omega, t) = \frac{1}{\gamma} \frac{\partial}{\partial \omega} \log M^f_{\varphi_\gamma}(\omega, t) + 2\pi\omega. \tag{5}$$

1. Michael R. Portnoff, “Magnitude-phase relationships for short-time Fourier transforms based on Gaussian analysis windows,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP ’79*, Apr. 1979, vol. 4, pp. 186–189.
2. F. Auger, E. Chassande-Mottin, and P. Flandrin, “On phase-magnitude relationships in the short-time Fourier transform,” IEEE Signal Process ing Letters, vol. 19, no. 5, pp. 267–270, May 2012
3. Zdeněk Průša, Peter Balazs, and Peter L. Søndergaard, “A Non-iterative Method for (Re)Construction of Phase from STFT magnitude,” 2016, Preprint available from http://ltfat.github.io/notes/ltfatnote040.pdf.

Line integration and phase shift

Gradient theorem (aka *Fundamental theorem of calculus for line integrals*):

$$\Phi^{f}_{\varphi_\gamma}(\omega, t) - \Phi^{f}_{\varphi_\gamma}(\omega_0, t_0) = \int_0^1 \nabla \Phi^{f}_{\varphi_\gamma}\left(r(\tau)\right) \cdot \frac{\mathrm{d}r}{\mathrm{d}\tau}(\tau)\, \mathrm{d}\tau, \tag{6}$$

where $r(\tau) = [r_\omega(\tau), r_t(\tau)]$ is a parametric representation of any curve starting at (ω_0, t_0) and ending at (ω, t).

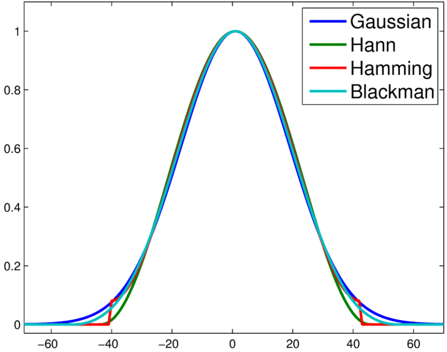
Discrete STFT

$$\left(V_g f\right)(m,n) = \sum_{l \in \mathcal{J}} f(l+na) g(l) \mathrm{e}^{-\mathrm{i}2\pi ml/M} \qquad (7)$$

$$= s(m,n)\mathrm{e}^{\mathrm{i}\phi(m,n)} \qquad (8)$$

γ for non-Gaussian windows

$$\gamma = C_g \cdot \mathrm{len}(g)^2 \qquad (9)$$



The (scaled) discrete STFT phase gradient $\nabla\phi = (\phi_\omega, \phi_t)$ is estimated from $s_{\log} = \log(s)$ using centered finite differences.

Algorithm 1: Phase gradient heap integration – PGHI

Input: Phase gradient $\nabla\phi(m,n) = \big(\phi_\omega(m,n), \phi_t(m,n)\big)$, magnitude of coefficients $s(m,n)$, relative tolerance *tol*.
Output: Estimate of the DGT phase $\widehat{\phi}(m,n)$.
Set $\mathcal{I} = \Big\{(m,n) : s(m,n) > tol \cdot \max\big(s(m,n)\big)\Big\}$;
Assign random values to $\widehat{\phi}(m,n)_{(m,n)\notin\mathcal{I}}$;
Construct a self-sorting *heap* for (m,n) pairs;

```
while I is not ∅ do
    if heap is empty then
        Insert (m,n)_max = arg max ( s(m,n)_{(m,n)∈I} ) into the heap;
        φ̂(m,n)_max ← 0 and remove (m,n)_max from I;
    end
    while heap is not empty do
        (m,n) ← remove the top of the heap;
        if (m+1,n) ∈ I then
            φ̂(m+1,n) ← φ̂(m,n) + 1/2 (φ_ω(m,n) + φ_ω(m+1,n));
            Remove (m+1,n) from I and insert it into heap;
        end
        if (m−1,n) ∈ I then
            φ̂(m−1,n) ← φ̂(m,n) − 1/2 (φ_ω(m,n) + φ_ω(m−1,n));
            Remove (m−1,n) from I and insert it into heap;
        end
        if (m,n+1) ∈ I then
            φ̂(m,n+1) ← φ̂(m,n) + 1/2 (φ_t(m,n) + φ_t(m,n+1));
            Remove (m,n+1) from I and insert it into heap;
        end
        if (m,n−1) ∈ I then
            φ̂(m,n−1) ← φ̂(m,n) − 1/2 (φ_t(m,n) + φ_t(m,n−1));
            Remove (m,n−1) from I and insert it into heap;
        end
    end
end
```

Algorithm 2: Real-Time Phase Gradient Heap Integration for n-th frame

Input: Phase time derivative $\phi_t(m,n)$ and magnitude $s(m,n)$ of frames n and $n-1$, phase frequency derivative $\phi_\omega(m,n)$ for frame n, estimated phase $\widehat{\phi}(m,n)$ for frame $n-1$ and relative tolerance *tol*.
Output: Phase estimate $\widehat{\phi}(m,n)$ for frame n.
$abstol \leftarrow tol \cdot \max\left(s(m,n) \cup s(m,n-1)\right)$;
Create set $\mathcal{I} = \left\{(m,n) : s(m,n) > abstol\right\}$;
Assign random values to $\widehat{\phi}(m,n)_{(m,n)\notin\mathcal{I}}$;
Construct a self-sorting *heap* for (m,n) tuples;
Insert $(m,n-1)$ for $m = \left(m : s(m,n-1) > abstol\right)$ into the *heap*;
while $\mathcal{I}$ *is not* $\emptyset$ **do**
  while *heap is not empty* **do**
    $(m_{heap}, n_{heap}) \leftarrow$ remove the top of the *heap*;
    if $n_{heap} == n-1$ **then**
      if $(m_{heap}, n) \in \mathcal{I}$ **then**
        $\widehat{\phi}(m_{heap}, n) \leftarrow$
        $\widehat{\phi}(m_{heap}, n-1) + \frac{1}{2}\left(\phi_t(m_{heap}, n-1) + \phi_t(m_{heap}, n)\right)$;
        Insert (m_{heap}, n) into the *heap* and remove it from $\mathcal{I}$;
      end
    end
    if $n_{heap} == n$ **then**
      if $(m_{heap}+1, n) \in \mathcal{I}$ **then**
        $\widehat{\phi}(m_{heap}+1, n) \leftarrow$
        $\widehat{\phi}(m_{heap}, n) + \frac{1}{2}\left(\phi_\omega(m_{heap}, n) + \phi_\omega(m_{heap}+1, n)\right)$;
        Insert $(m_{heap}+1, n)$ into the *heap* and remove it from $\mathcal{I}$;
      end
      if $(m_{heap}-1, n) \in \mathcal{I}$ **then**
        $\widehat{\phi}(m_{heap}-1, n) \leftarrow$
        $\widehat{\phi}(m_{heap}, n) - \frac{1}{2}\left(\phi_\omega(m_{heap}, n) + \phi_\omega(m_{heap}-1, n)\right)$;
        Insert $(m_{heap}-1, n)$ into the *heap* and remove it from $\mathcal{I}$;
      end
    end
  end
end

Comparison with SPSI and RTISI-LA EBU SQAM database, $M = 2048$

Overlap 3/4	Gauss	Hann	Hamming	Blackman
SPSI	-17.82	-16.72	-16.53	-17.62
Proposed (0)	-17.76	-17.50	-17.58	-17.82
RTISI (0)	**-22.80**	**-21.75**	**-21.08**	**-22.82**
Proposed (1)	-20.91	-20.25	-20.07	-20.76
RTISI-LA (1)	**-26.08**	**-25.14**	**-24.01**	**-26.63**

Overlap 15/16	Gauss	Hann	Hamming	Blackman
SPSI	-17.99	-17.16	-16.82	-17.92
Proposed (0)	**-26.13**	**-23.48**	**-22.33**	**-25.93**
RTISI (0)	-20.18	-19.70	-19.01	-20.35
Proposed (1)	**-26.83**	**-23.50**	**-22.47**	**-26.21**
RTISI-LA (1)	-17.85	-16.66	-16.12	-17.71

Overlap 7/8	Gauss	Hann	Hamming	Blackman
SPSI	-17.88	-17.09	-16.79	-17.79
Proposed (0)	**-21.79**	**-21.10**	**-21.01**	**-21.78**
RTISI (0)	-21.15	-20.20	-19.53	-21.07
Proposed (1)	**-24.87**	**-22.74**	**-21.98**	**-24.59**
RTISI-LA (1)	-22.11	-19.90	-19.14	-21.90

$$E = 20\log_{10} \frac{\left\| s - \left| V_g \widehat{f} \right| \right\|_{\text{fro}}}{\|s\|_{\text{fro}}}$$

SPSI: Gerald T. Beauregard, Mithila Harish, and Lonce Wyse, “Single pass spectrogram inversion,” in *IEEE International Conference on Digital Signal Processing (DSP), 2015*, July 2015, pp. 427–431.
RTISI(-LA): Xinglei Zhu, Gerald T. Beauregard, and Lonce Wyse, “Real-time signal estimation from modified short-time Fourier transform magnitude spectra,” in *IEEE Transactions on Audio, Speech, and Language Processing,* vol. 15, no. 5, pp. 1645–1653, July 2007.

DEMO

Reproducible research

Complemetary material (Matlab code, extended comparison, sound examples):

`http://ltfat.github.io/notes/043`

Matlab/GNU Octave toolboxes:

- LTFAT – Large Time-Frequency Analysis Toolbox
  `http://ltfat.github.io`
- PHASERET – Phase Retrieval Toolbox
  `http://ltfat.github.io/phaseret`

Thank you for your attention.

Questions?