WWW.KFS.OEAW.AC.AT

ÖAW ÖSTERREICHISCHE AKADEMIE DER WISSENSCHAFTEN

Signal Reconstruction from the Discrete Gabor Transform Magnitude

ISF

Zdeněk Průša, Peter Balazs and Peter L. Søndergaard

The magnitude of the Short-Time Fourier transform (STFT) with respect to the Gaussian window uniquely determines the phase (up to a global constant phase shift) via the relationship of the partial derivatives of the phase and of the log-magnitude. Based on this relationship, we develop an efficient algorithm for the reconstruction of the phase from the discrete Gabor transform (DGT, sampled and discretized STFT) magnitude, and, by extension, propose a method for reconstruction of the original signal. The algorithm is particularly suitable for audio signals for which the phase reconstruction imperfections are partially masked by the human hearing system. We show that the performance of the algorithm depends on the STFT sampling density.

Theory

The Short-Time Fourier transform is defined as:

$$(\mathcal{V}_g f)(\omega,t) = \int_{\mathbb{R}} f(\tau)\overline{g(\tau-t)}\mathrm{e}^{-\mathrm{i}2\pi\omega(\tau-t)}\,\mathrm{d}\tau,\quad \omega,t\in\mathbb{R}$$
$$=: M_g^f(\omega,t)\cdot e^{\mathrm{i}\Phi_g^f(\omega,t)}.$$

Using the Gaussian window $g=\varphi_\gamma$

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

the *phase gradient* $\nabla\Phi^f_{\varphi_\gamma}$ can be expressed as [Portnoff, 79]

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

The *gradient theorem* recovers the phase up to $\Phi^f_{\varphi_\gamma}(\omega_0,t_0)$:

$$\Phi^f_{\varphi_\gamma}(\omega,t)-\Phi^f_{\varphi_\gamma}(\omega_0,t_0) = \int_0^1 \nabla\Phi^f_{\varphi_\gamma}(L(\tau))\cdot\frac{\mathrm{d}L}{\mathrm{d}\tau}(\tau)\,\mathrm{d}\tau,$$

$L(\tau)=[L_\omega(\tau),L_t(\tau)]$ is any line $(\omega_0,t_0)\to(\omega,t)$.

Discrete Phase Gradient

The discrete Gabor transform is defined as:

$$c(m,n) = \sum_{l=0}^{L-1} f(l)\overline{g(l-na)}\mathrm{e}^{-\mathrm{i}2\pi m(l-na)/M}$$
$$=: s(m,n)\cdot \mathrm{e}^{\mathrm{i}\phi(m,n)}$$

for $m=0,\dots,M-1$, $n=0,\dots,N-1$, $M=L/b$ number of frequency channels, $N=L/a$ number of time shifts, a is a hop factor and b is a hop factor in frequency.

Approximation of the STFT phase gradient

$$\nabla\phi(m,n) = (\phi_\omega(m,n),\phi_t(m,n)) \approx \nabla\Phi^f_{\varphi_\lambda}(bm,an)$$

Scaled gradient

$$\nabla\phi^{\mathrm{SC}}(m,n) = (b\phi_\omega(m,n), a\phi_t(m,n)) := \left(-\frac{\gamma}{aM}(s_{\log}D_t^T)(m,n), \frac{aM}{\gamma}(D_\omega s_{\log})(m,n)+2\pi am/M\right)$$

where D_t^T, D_ω perform the numerical differentiation along rows (in time) and columns (in frequency) of $s_{\log}=\log(s)$ respectively.

Phase Gradient Heap Integration

Adaptive-path numerical *integration* (trapez. rule):

- Start at the largest $s(m,n)$, spread the phase to the neighbors and repeat with the next largest coefficient with already computed phase.

***Heap* data structure for tuples (m,n):**

- Keeps (m,n) with the max. $s(m,n)$ at the top.
- Dynamic, efficient heap insertion and deletion.

The Algorithm

Input: Phase gradient $\nabla\phi^{\mathrm{SC}}(m,n) = \left(\phi^{\mathrm{SC}}_\omega(m,n),\phi^{\mathrm{SC}}_t(m,n)\right)$, magnitude of DGT coefficients $s(m,n)$, tolerance *tol*.
Output: Estimate of the DGT phase $\widehat{\phi}(m,n)$.
Create set $\mathcal{I} = \left\{(m,n): s(m,n) > tol\cdot\max\left(s(m,n)\right)\right\}$;
Assign random values to $\widehat{\phi}(m,n)_{(m,n)\notin\mathcal{I}}$;
Construct empty *heap* for (m,n);
while $\mathcal{I}$ *is not* $\emptyset$ **do**
  if *heap is empty* **then**
    Insert $(m,n)_{\max} = \arg\max\left(s(m,n)_{(m,n)\in\mathcal{I}}\right)$ into the *heap*;
    $\widehat{\phi}(m,n)_{\max} \leftarrow 0$;
    Remove $(m,n)_{\max}$ from $\mathcal{I}$;
  end
  while *heap is not empty* **do**
    $(m,n) \leftarrow$ remove the top of the *heap*;
    if $(m+1,n)\in\mathcal{I}$ **then**
      $\widehat{\phi}(m+1,n) \leftarrow \widehat{\phi}(m,n) + \frac{1}{2}\left(\phi^{\mathrm{SC}}_\omega(m,n)+\phi^{\mathrm{SC}}_\omega(m+1,n)\right)$;
      Remove $(m+1,n)$ from $\mathcal{I}$ and insert into the *heap*;
    end
    if $(m-1,n)\in\mathcal{I}$ **then**
      $\widehat{\phi}(m-1,n) \leftarrow \widehat{\phi}(m,n) - \frac{1}{2}\left(\phi^{\mathrm{SC}}_\omega(m,n)+\phi^{\mathrm{SC}}_\omega(m-1,n)\right)$;
      Remove $(m-1,n)$ from $\mathcal{I}$ and insert into the *heap*;
    end
    if $(m,n+1)\in\mathcal{I}$ **then**
      $\widehat{\phi}(m,n+1) \leftarrow \widehat{\phi}(m,n) + \frac{1}{2}\left(\phi^{\mathrm{SC}}_t(m,n)+\phi^{\mathrm{SC}}_t(m,n+1)\right)$;
      Remove $(m,n+1)$ from $\mathcal{I}$ and insert into the *heap*;
    end
    if $(m,n-1)\in\mathcal{I}$ **then**
      $\widehat{\phi}(m,n-1) \leftarrow \widehat{\phi}(m,n) - \frac{1}{2}\left(\phi^{\mathrm{SC}}_t(m,n)+\phi^{\mathrm{SC}}_t(m,n-1)\right)$;
      Remove $(m,n-1)$ from $\mathcal{I}$ and insert into the *heap*;
    end
  end
end

Exploiting Partially Known Phase

In case the phase of some of the coefficients of regions of coefficients is known.

- Introduce mask $\mathcal{M}$ to select the reliable coefficients.
- Identify the border coefficients i.e. coefficients with at least one neighbor with unknown phase.
- Pre-load the heap with the border coefficients.

Formally, execute the following before entering the main loop of the algorithm:

Additional input: Set (mask) of indices of coefficients $\mathcal{M}$ with known phase $\phi(m,n)_{(m,n)\in\mathcal{M}}$.
$\widehat{\phi}(m,n) \leftarrow \phi(m,n)$ for $(m,n)\in\mathcal{M}$;
for $(m,n)\in\mathcal{M}\cap\mathcal{I}$ **do**
  if $(m+1,n)\notin\mathcal{M}$ *or* $(m-1,n)\notin\mathcal{M}$ *or* $(m,n+1)\notin\mathcal{M}$ *or* $(m,n-1)\notin\mathcal{M}$ **then**
    Add (m,n) to the *heap*;
  end
end

Implementation

Matlab/GNU Octave implementation available in

- LTFAT `http://ltfat.github.io`
  - `constructphase` – for complex signals
  - `constructphasereal` – for real signals
- PHASERET `http://ltfat.github.io/phaseret`
  - `pghi` – wrapper around `constructphasereal`
  - `rtpghi` – real-time version of the algorithm
  - `demo_blockproc_phaseret` – real-time audio demo

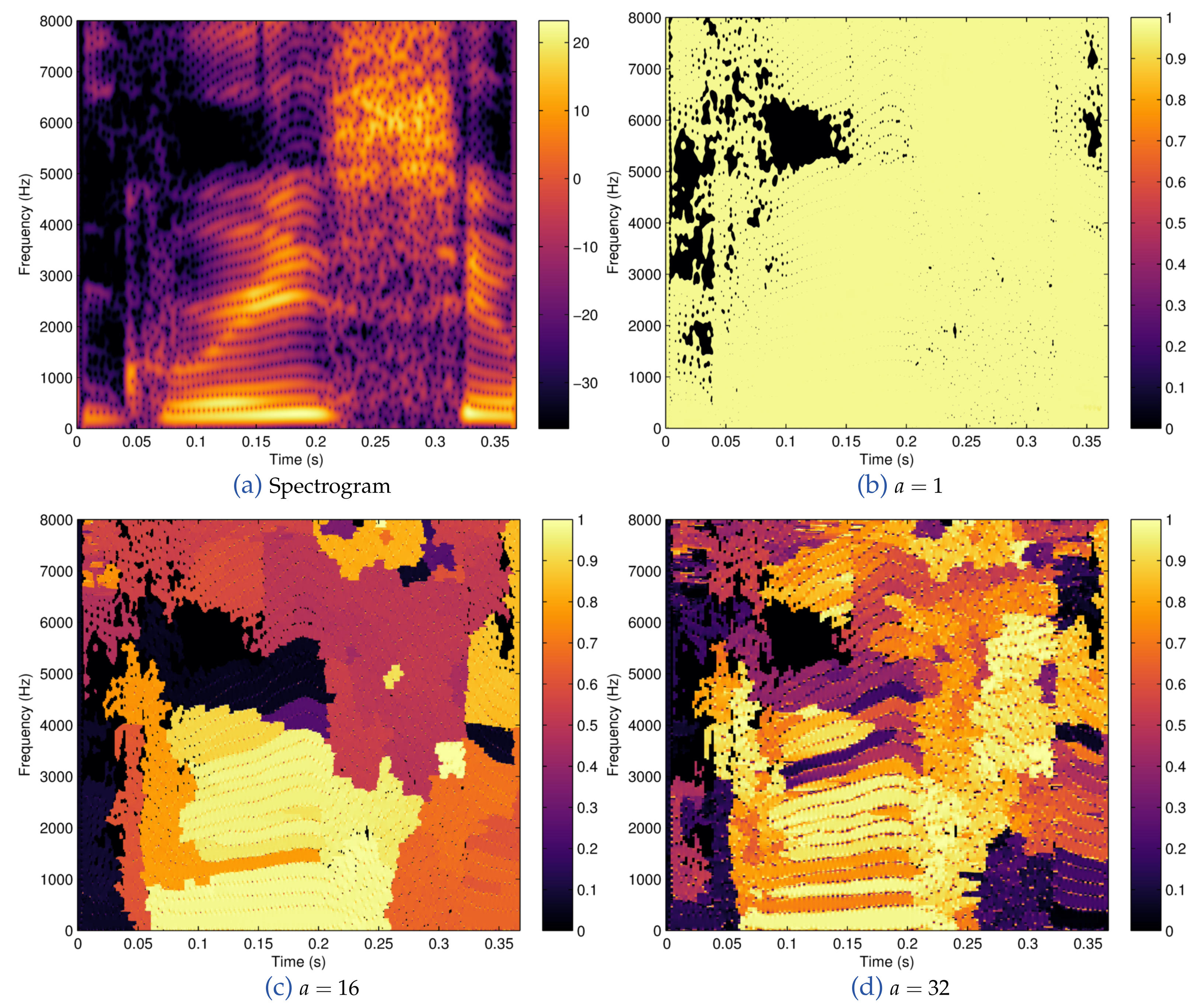
Acknowledgement

Work presented in this poster was supported by the Austrian Science Fund (FWF) START-project FLAME ("Frames and Linear Operators for Acoustical Modeling and Parameter Estimation"; Y 551-N13).

References

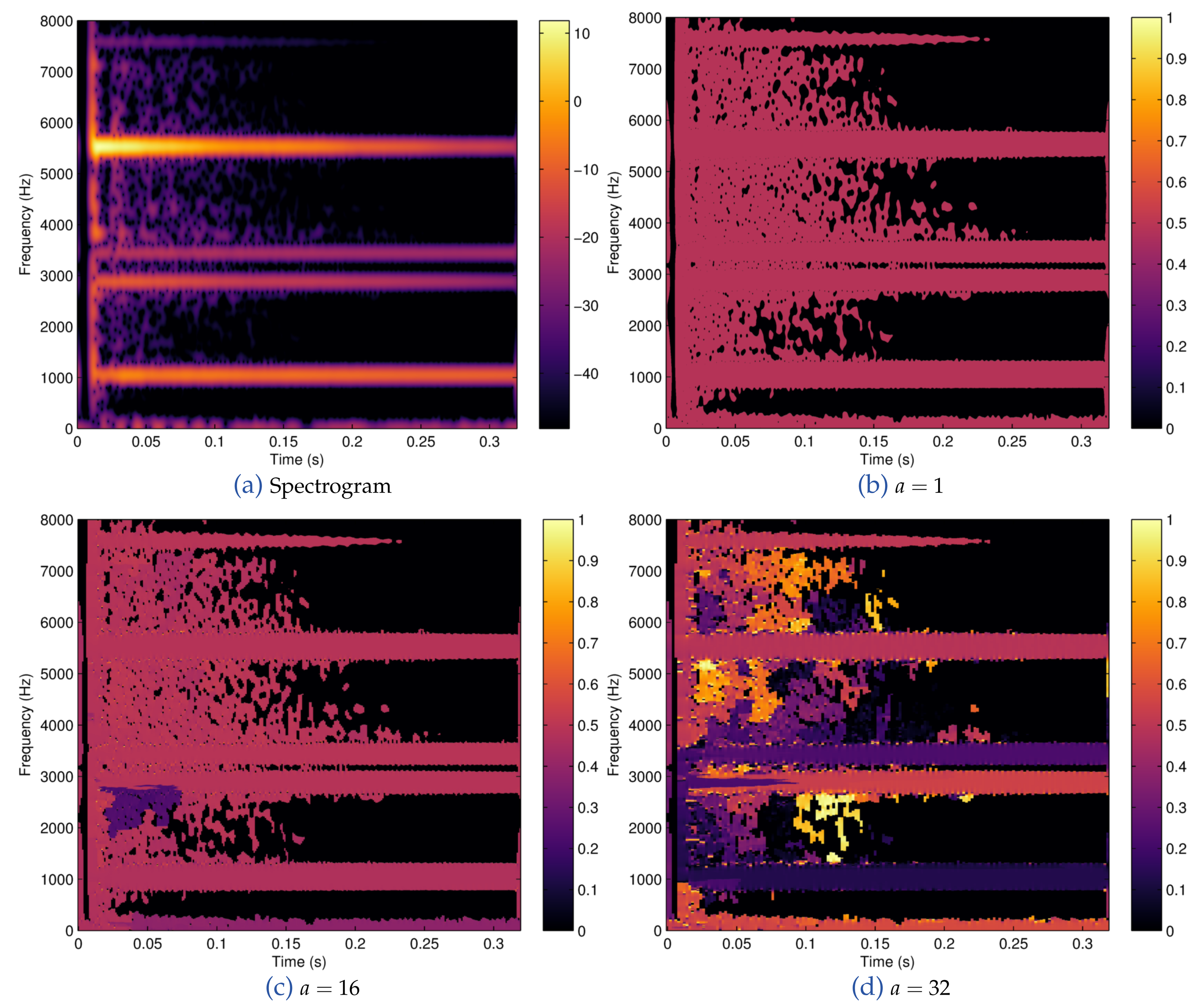
- M. R. Portnoff, *Magnitude-phase relationships for short-time Fourier transforms based on Gaussian analysis windows*, in Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP 79., vol. 4, Apr 1979, pp. 186189.
- Z. Průša, P. Balazs, P. L. Søndergaard: *A Non-iterative Method for STFT Phase (Re)Construction*, 2016. Preprint available from `http://ltfat.github.io/notes/ltfatnote040.pdf`
- Z. Průša, P. L. Søndergaard: *Real-Time Spectrogram Inversion Using Phase Gradient Heap Integration*, Proc. Int. Conf. Digital Audio Effects DAFx (DAFx-16), 2016. To appear. Preprint available from `http://ltfat.github.io/notes/ltfatnote043.pdf`
- Z. Průša, P. L. Søndergaard: *PhaseRet: STFT Phase Retrieval Toolbox For Audio Signals*, In preparation. Preprint available from `http://ltfat.github.io/notes/ltfatnote045.pdf`

Experiments



(a) Spectrogram (b) $a=1$ (c) $a=16$ (d) $a=32$

Figure : Spectrogram of a spoken word *greasy* (a). Phase differences of the STFT of the original and reconstructed signals for varying time hop size a (b) (c) (d).



(a) Spectrogram (b) $a=1$ (c) $a=16$ (d) $a=32$

Figure : Spectrogram of an excerpt from *glockenspiel* (a). Phase differences of the STFT of the original and reconstructed signals for varying time hop size a (b) (c) (d).

ISF-INSTITUT FÜR SCHALLFORSCHUNG