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:

\[
\begin{align*}
\forall \phi(f) (m,n) &= \int_{-\infty}^{\infty} \psi(f-n) \cdot M(f-t) \cdot e^{2\pi ift} dt, f \in \mathbb{R} \\
\psi(f) &= e^{-\frac{f^2}{2}} \quad \text{Gaussian window}
\end{align*}
\]

Using the Gaussian window \( \psi = e^{-\frac{f^2}{2}} \),

\[
\psi^2(t) = \left( \frac{i}{2} \right) \gamma e^{-\frac{t^2}{2\gamma^2}}
\]

the phase gradient \( \nabla \Phi_\psi \), can be expressed as [Portnoy, 79]

\[
\begin{align*}
\nabla \Phi_\psi(m,n) &= -\nabla \log(M^t(m,n)) - \frac{n}{\gamma^2} \log(M^t(m,n)) + 2n \gamma \\
\text{Gradient theorem recovers the phase up to}
\end{align*}
\]

\[
\Phi_\psi(m,n) = \Phi_\psi(m_0,n_0) + \int_{0}^{1} \nabla \Phi_\psi(L(t)) \cdot dt, \\
L(t) = [m_0(t), n_0(t)] \text{ is any line} \quad \text{if} \quad (m_0, n_0) \rightarrow (m, n).
\]

Discrete Phase Gradient

The discrete Gabor transform is defined as:

\[
\forall \phi(f) = \sum_{n=0}^{M-1} \sum_{m=0}^{N-1} \phi(m,n) \cdot e^{-2\pi if f} / \mathcal{M} / \mathbb{M}
\]

for \( M = 2^{m}, \ldots, M = 1, \quad n = 0, \ldots, N - 1, \quad M = L/b \quad \text{number of frequency channels}, \quad N = L/\text{num of time shifts}, \quad a \quad \text{a hop factor} \quad \text{and} \quad b \quad \text{a hop factor in frequency}.

Approximation of the STFT phase gradient \( \nabla \Phi_{\psi}(m,n) \approx \nabla \Phi_\psi(m_0,n_0) \)

Scaled gradient

\[
\begin{align*}
\nabla \Phi_{\psi}(m,n) &= \left[ \psi_\psi(m,n), \psi_\psi(m,n) \right] \\
&= \left[ \frac{1}{\mathcal{M}} \frac{\partial D^2 }{D_\psi} \left[ \frac{1}{\mathbb{M}} \frac{\partial D^2 }{D_\psi} \right] (m,n) + 2mn / \mathcal{M}
\end{align*}
\]

where \( D^2 D_\psi \) perform the numerical differentiation along rows (in time) and columns (in frequency) of \( \psi_\psi = \log(\psi) \) respectively.

Phase Gradient Heap Integration

Adaptive-path numerical integration (trapez. rule):

- Start at the largest \( m, n \), 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 \( \Phi_{\psi}(m,n) = \left\{ \Phi^{m,n}_0(m,n), \Phi^{m,n}_1(m,n) \right\} \)

Output: Estimate of the DGT phase \( \Phi(m,n) \).

Create set \( \Gamma = \{ \{ m,n \} : s(m,n) > tol \max( s(m,n) ) \} \);

Assign random values to \( \Phi(m,n) \);

Construct empty heap for \( m,n \);

while \( \Gamma \neq \emptyset \) do

if heap is empty then

Insert \((m,n), \max \{ s(m,n) \} \) into the heap;

else

while heap is not empty do

Remove \((m,n)\) from heap;

end

end

end

end

Exploring Partially Known Phase

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

- Introduce mask \( M \) 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 \( \{ \text{mask} \} \) of indices of coefficients \( m,n \) with known phase \( \Phi(m,n) \).

for \( m,n \in M \) do

if \( \{ m,n \} \notin M \) or \( \{ m,n+1 \} \notin M \) or \( \{ m,n+1 \} \notin M \) then

Add \( \{ m,n \} \) to the heap;

end

end

Implementation

Matlab/GNU Octave implementation available in

- [LTFAT](http://ltfat.github.io)
- constru..phase - for complex signals
- constru..phase.real - for real signals
- [PHASERET](http://ltfat.github.io/phasere..)
- phaseret - Wrapper around constru..phaseret
- phaseret - real-time version of the algorithm
- demo..lockp..phasere.. - real-time audio demo

Acknowledgement

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

References