PADERBORN UNIVERSITY

Computer Science, Electrical Engineering and Mathematics

NT Communications Engineering
Prof. Dr.-Ing. Reinhold Häb-Umbach

NARA-WPE: A Python package for weighted prediction error dereverberation in Numpy and Tensorflow for online and offline processing

Lukas Drude, Jahn Heymann, Christoph Boeddeker, Reinhold Haeb-Umbach
Department of Communications Engineering, Paderborn University, Germany
{drude, heymann, boeddeker, haeb}@nt.upb.de

Introduction

- Open-source implementation of Weighted prediction error (WPE)
  - Single & multi channel
  - Offline, block-online & frame-online processing
  - Numpy & TensorFlow implementation
- Modular code design to facilitate further research

Weighted prediction error

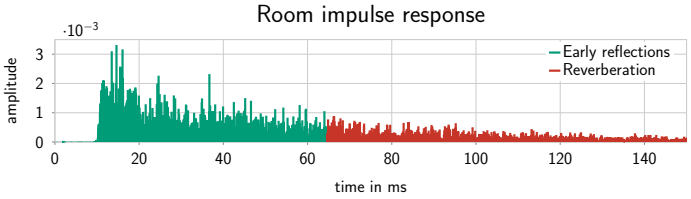
WPE is an effective dereverberation method and one of the few front-ends which still improves WERs even in combination with a strong back-end

STFT model assumption: $\mathbf{y}_{t,f} = \mathbf{x}^{(\text{early})}_{t,f} + \mathbf{x}^{(\text{tail})}_{t,f}$

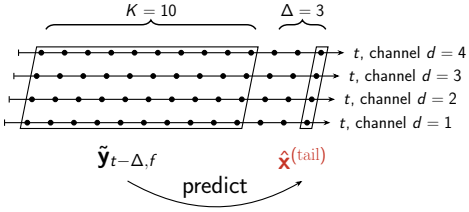
$$\mathbf{x}^{(\text{early})} = \mathbf{h}^{(\text{early})} * s$$

$$\mathbf{x}^{(\text{tail})} = \mathbf{h}^{(\text{tail})} * s$$

Room impulse response



Use linear prediction to subtract reverberation

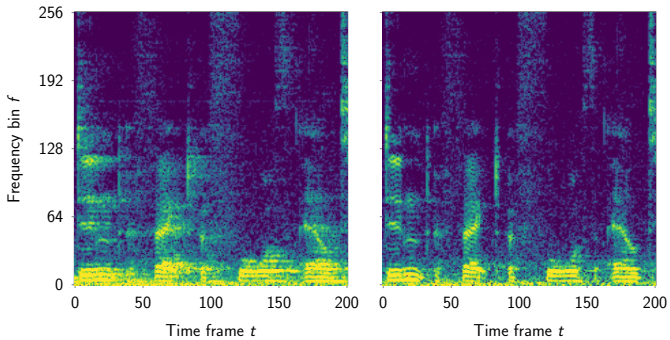


Algorithm

- Initialize: $\hat{\mathbf{x}}^{(\text{early})}_{t,f} = \mathbf{y}_{t,f}$

1. Power estimation $\lambda_{t,f}$, e.g. smooth $\hat{\mathbf{x}}^{(\text{early})}_{t,f}$
2. Correlation terms:
   - Offline: $\mathbf{R}_f = \frac{1}{T}\sum_t \frac{\tilde{\mathbf{y}}_{t-\Delta,f}\tilde{\mathbf{y}}^{\mathsf{H}}_{t-\Delta,f}}{\lambda_{t,f}}$, $\mathbf{P}_f = \frac{1}{T}\sum_t \frac{\tilde{\mathbf{y}}_{t-\Delta,f}\mathbf{y}^{\mathsf{H}}_{t,f}}{\lambda_{t,f}}$
   - Block-Online: $\mathbf{R}_{k,f} = \alpha\mathbf{R}_{k-1,f} + \sum_{\tau\in B}\frac{\tilde{\mathbf{y}}_{\tau-\Delta,f}\tilde{\mathbf{y}}^{\mathsf{H}}_{\tau-\Delta,f}}{\lambda_{\tau,f}}$, $\mathbf{P}_{k,f} = \alpha\mathbf{P}_{k-1,f} + \sum_{\tau\in B}\frac{\tilde{\mathbf{y}}_{\tau-\Delta,f}\mathbf{y}^{\mathsf{H}}_{\tau,f}}{\lambda_{\tau,f}}$
   - Frame-Online: $\mathbf{R}_{t,f} = \alpha\mathbf{R}_{t-1,f} + \frac{\tilde{\mathbf{y}}_{t-\Delta,f}\tilde{\mathbf{y}}^{\mathsf{H}}_{t-\Delta,f}}{\lambda_{t,f}}$, $\mathbf{P}_{t,f} = \alpha\mathbf{P}_{t-1,f} + \frac{\tilde{\mathbf{y}}_{t-\Delta,f}\mathbf{y}^{\mathsf{H}}_{t,f}}{\lambda_{t,f}}$
3. Filter matrix estimation: $\mathbf{G}_f = \mathbf{R}_f^{-1}\mathbf{P}_f$
4. Early-arriving speech prediction: $\hat{\mathbf{x}}^{(\text{early})}_{t,f} = \mathbf{y}_{t,f} - \overbrace{\mathbf{G}_f^{\mathsf{H}}\tilde{\mathbf{y}}_{t-\Delta,f}}^{\hat{\mathbf{x}}^{(\text{tail})}}$
5. Goto 1. (iterative) or stop

Example utterance



Installation

Start your Python 3 environment and run:

```
pip install nara_wpe
```

Alternatively, install it directly from Github:

```
pip install git+https://github.com/fgnt/nara_wpe
```

Code examples

Offline example: Numpy

```
import numpy as np
from nara_wpe import np_wpe as wpe
from nara_wpe.utils import stft, istft

y = aquire_audio_data()
Y = stft(y).transpose(2, 0, 1)

Z = wpe(Y)

z = istft(Z.transpose(1, 2, 0))
```

Offline example: TensorFlow

```
import tensorflow as tf
from nara_wpe import tf_wpe as wpe
from nara_wpe.utils import stft, istft

y = aquire_audio_data()
Y = stft(y).transpose(2, 0, 1)
Y_tf = tf.placeholder(...)
Z_tf = wpe(Y_tf)
with tf.Session() as session:
    Z = session.run(Z_tf, {Y_tf: Y})
z = istft(Z.transpose(1, 2, 0))
```

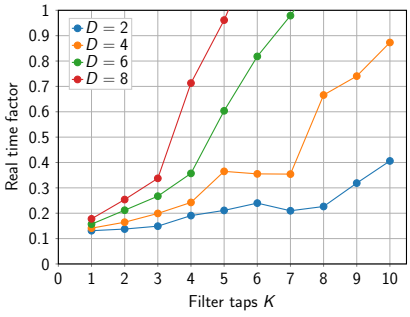
Frame-online example: TensorFlow

```
import numpy as np
import tensorflow as tf
from nara_wpe.tf_wpe import online_wpe_step
from nara_wpe.utils import stft, istft

Q = np.identity(...)
G = np.zeros(...)
with tf.Session() as session:
    Y_tf, Q_tf, G_tf = tf.placeholder(...), tf.placeholder(...), tf.placeholder(...)
    results = online_wpe_step(Y_tf, Q_tf, G_tf)
    for Y in aquire_framebuffer():
        feed_dict = {Y_tf: Y, Q_tf: Q, G_tf: G}
        Z, Q, G = session.run(results, feed_dict)
```

Real-time performance on Raspberry Pi 3

Real time factor for different parameter settings when running the frame-online WPE implementation on a Raspberry Pi Model 3b+ (1 GB LPDDR2 RAM, 1.4 GHz). Each dot represents an average of at least 100 runs.



ASR Results

Word error rates/% on the *real* REVERB evaluation sets:

	Offline		Block-Online		Frame-Online	
	2 ch	8 ch	2 ch	8 ch	2 ch	8 ch
Unprocessed	17.6	17.6	17.6	17.6	17.6	17.6
Iterative	**14.4**	**10.9**				
Non-iterative	16.1	13.0	15.7	14.0	17.4	16.2

13. ITG Fachtagung Sprachkommunikation Oldenburg

This work was in part supported by Deutsche Forschungsgemeinschaft (DFG) under contract no. Ha3455/11-1 and a Google Faculty Research Award. We thank Dr. Kinoshita, Dr. Nakatani and his colleagues for valuable discussions. Computational resources were provided by the Paderborn Center for Parallel Computing.