Efficient Implementation of the Room Simulator for Training Deep Neural Network Acoustic Models

Chanwoo Kim[1†], *Ehsan Variani*[2], *Arun Narayanan*[2], *and Michiel Bacchiani*[2]

[1]Samsung Research, [2]Google Speech

[1]chanw.com@samsung.com [2]{variani, arunnt, michiel}@google.com

Abstract

In this paper, we describe how to efficiently implement an acoustic room simulator to generate large-scale simulated data for training deep neural networks. Even though *Google Room Simulator* in [1] was shown to be quite effective in reducing the Word Error Rates (WERs) for far-field applications by generating simulated far-field training sets, it requires a very large number of FFTs. *Room Simulator* used approximately 80 % of CPU usage in our *CPU/GPU* training architecture [2]. In this work, we implement an efficient OverLap Addition (OLA) based filtering using the open-source `FFTW3` library. Further, we investigate the effects of the Room Impulse Response (RIR) lengths. Experimentally, we conclude that we can cut the tail portions of RIRs whose power is less than 20 *dB* below the maximum power without sacrificing the speech recognition accuracy. However, we observe that cutting RIR tail more than this threshold harms the speech recognition accuracy for rerecorded test sets. Using these approaches, we were able to reduce CPU usage for the room simulator portion down to 9.69 % in CPU/GPU training architecture. Profiling result shows that we obtain 22.4 times speed-up on a single machine and 37.3 times speed up on Google's distributed training infrastructure.

Index Terms: Simulated data, room acoustics, robust speech recognition, deep learning

1. Introduction

With advancements in deep learning [3, 4, 5, 6, 7, 8], speech recognition accuracy has improved dramatically. Now, speech recognition systems are used not only on portable devices but also on standalone devices for far-field speech recognition. Examples include voice assistant systems such as Amazon Alexa and Google Home [1, 9]. In far-field speech recognition, the impact of noise and reverberation is much larger than near-field cases. Traditional approaches to far-field speech recognition include noise robust feature extraction algorithms [10, 11, 12], on-set enhancement algorithms [13, 14], and multi-microphone approaches [15, 16, 17, 18, 19, 20, 21]. Recently, we observed that training with large-scale noisy data generated by a *Room Simulator* [1] improves speech recognition accuracy dramatically. This system has been successfully employed for training acoustic models for Google Home or Google voice search [1].

Room Simulator creates millions of virtual rooms with different dimensions and different number of sound sources at different locations and Signal-to-Noise Ratios (SNRs). For every new utterance in the training set, we use a randomly sampled room configuration, so that the same utterance is simulated under different acoustic environments in every epoch during training. As will be seen in Sec. 3, if we generate the simulated utterance only once for each input example, the performance is worse. Since different RIRs are applied for the same utterance at every epoch, the intermediate results cannot be cached. Therefore, the noisification process requires very large number of convolution operations. As will be seen in Sec. 3, during training, *Room Simulator* used up to 80 % of the whole CPU usage if we use the CPU/GPU architecture shown in Fig. 1. In this paper, we describe our approach to reduce the computational portion of the *Room Simulator* below 10 % of the CPU usage in our CPU/GPU training scheme.

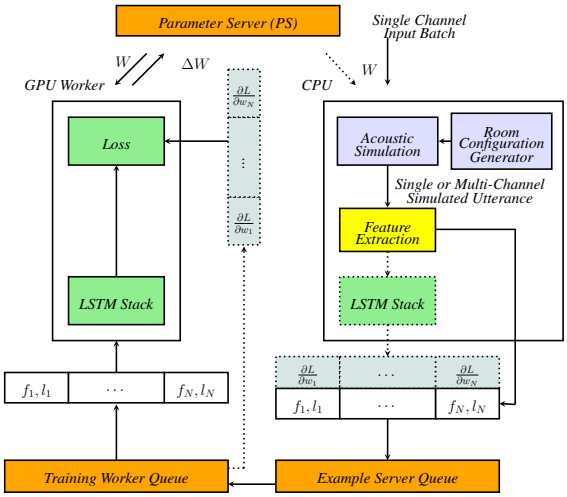


Figure 1: Training architecture using cluster of CPUs and GPUs. [2]. *Room Simulator* is on the CPU side to generate simulated utterances using room configurations. [1].

2. Efficient Implementation of the Room Simulation System

Fig. 1 shows a high level block diagram of the acoustic model training infrastructure [2]. Since it is not easy to run all the front-end processing blocks such as feature extraction and Voice Activity Detection (VAD) on GPUs, they run on CPUs during training. Even though Fast Fourier Transform (FFT) of the *Room Simulator* may be efficiently implemented on GPUs, since *Room Simulator* must feed the simulated utterances to the rest of CPU optimized front-end components, the Room Simulator is also running on CPUs.

2.1. Review of the Room Simulator for Data Augmentation

In this section, we briefly review the structure of the Google Room Simulator for generating simulated utterances to train acoustic models for speech recognition systems [1]. We as-

[†]Work performed while at Google.

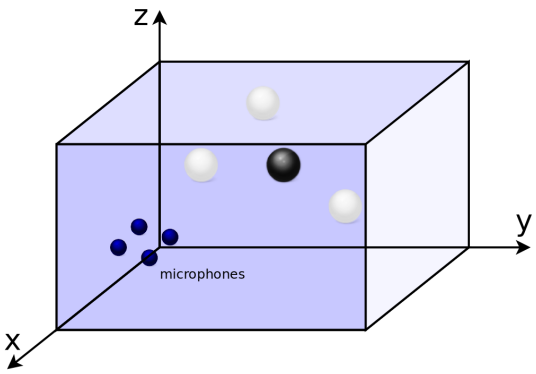


Figure 2: A simulated room: There may be multiple microphones, a single target sound source, multiple noise sources in a cuboid-shape room with acoustically reflective walls [1].

sume a room of a rectangular cuboid-shape as shown in Fig. 2. Assuming that all the walls of a room reflect acoustically uniformly, we use the image method to model the Room Impulse Responses (RIRs) [22, 23, 24, 25]. In the image method, a real room is acoustically mirrored with respect to each wall, which results in grids of virtual rooms. In our work for training the acoustic model for Google Home [1, 9], we consider $17 \times 17 \times 17 = 4913$ virtual rooms for RIR calculation. Following the image method and assuming that there are V virtual sound sources including one real source, the impulse response is calculated using the following equation [22, 23]:

$$h[n] = \sum_{v=0}^{V-1} \frac{r^{g_v}}{d_v} \delta \left[n - \left\lceil \frac{d_v f_s}{c_0} \right\rceil \right], \tag{1}$$

where v is the index of each virtual sound source, and $d_v > 0$ is the distance from that sound source to the microphone, $0 < r < 1$ is the reflection coefficient of the wall, g_v is the number of the reflections to that sound source, f_s is the sampling rate of the RIR, and c_0 is the speed of sound in the air. We use the value of $f_s = 16,000$ *Hz* and $c_0 = 343$ *m/s* for *Room Simulator* [9]. For d_v, r, we use numbers created by a random number generator following specified distributions for each impulse response [1].

Assuming that there are I sound sources including one target source and J microphones, the received signal at microphone j is given by:

$$y_j[n] = \sum_{i=0}^{I-1} \alpha_{ij} \left(h_{ij}[n] * x_i[n] \right). \tag{2}$$

Since we used a two-microphones system in [1, 9] J is two, and for the number of noise sources, we used a value from zero up to three with an average of 1.55. Including the target source, the average number of I is 2.55 [1].

2.2. Efficient Room Impulse Response filtering

When the signal and the room impulse response lengths are N_x and N_h respectively, the number of multiplications C_{td} required for calculating (2) using the time-domain convolution is given by:

$$C_{td} = I \times J \times N_x \times N_h. \tag{3}$$

In our training set used in [1, 9], the average utterance length including *non-speech* portions marked by the Voice Activity Detector (VAD) is 7.31 s. This corresponds to N_x of 116,991 at 16 *kHz*. The average length of the RIR in the room simulator used in [1] is 0.243 s, corresponding to the N_h value of 3893. For Google Home the number of noise sources in our training set is 1.55 on average. Thus I is 2.55 including one target source, and J is two, since we use a two-microphones system in [1, 9]. Thus, if we directly use the linear convolution, it requires 2.32 billion multiplications per utterance from (3), which is prohibitively large. Thus, in the "*Room Simulator*" in [1], we used the frequency domain multiplication using `Kiss FFT` [26]. To avoid time aliasing, the FFT size N must satisfy $N \geq N_x + M - 1$, where M is the length of the impulse response. The j-th microphone channel of the simulated signal $y_j[n]$ in (2) is given by:

$$y_j[n] = \sum_{i=0}^{I-1} FFT^{-1} \left\{ FFT \left\{ h_{ij}[n] \right\} \times FFT \left\{ x_i[n] \right\} \right\}. \tag{4}$$

As shown in (4), we perform two FFTs, one IFFT, and one complex element-wise multiplications between two complex spectra for each convolution term in (2). Assuming that radix-2 FFTs are employed, each FFT or IFFT requires $\frac{N}{2} \log_2(N)$ multiplications [27]. In addition, a single element-wise complex multiplication between two complex spectra is required for each convolution term in (2). For real time-domain signals, we need to perform element-wise multiplications for the lower half-spectrum for Discrete Fourier Transform (DFT) indices $0 \leq k \leq N/2$ since the spectrum has the Hermitian symmetry property [27]. From this discussion, we conclude that the number of real multiplications C_{FFT} for calculating (4) is given by:

$$C_{FFT} = IJ \left(6N \log_2(N) + 2N \right). \tag{5}$$

For the average N_x of 116,991 mentioned above, if we assume that the average N is 2^{17}, (5) requires 69.5 million multiplications per utterance on average. *Room Simulator* for [1, 9] was implemented in C++ using the `Eigen3` linear algebra library [28]. So far we have used the `Kiss FFT` version of FFT in `Eigen3` including acoustic model training for Google Home described in [1, 9]. But, to further speed up frontend computation, we switched from `Kiss FFT` in `Eigen3` to a custom C++ class implementation which internally uses real FFT in `FFTW3`.

A more efficient approach is using the OverLap Add (OLA) FFT filtering [27, 29]. With the OLA FFT filtering, the approximate number of real multiplications is given by:

$$C_{OLA} = IJ \left(\left\lfloor \frac{N_x}{N - N_h + 1} \right\rfloor \left(4N \log_2(N) + 2N \right) + 2N \log_2(N) \right), \tag{6}$$

where N is the FFT size, N_x is the length of the entire signal, and N_h is the length of the impulse response. The term $4N \log_2(N) + 2N$ appears in (6), since there is one FFT, one IFFT, and one element-wise complex multiplication for $0 \leq k \leq N/2$ for each block. As before, we assumed that each FFT or IFFT requires $\frac{N}{2} \log_2(N)$ multiplications and one complex multiplication requires four real multiplications in (6). $\left\lfloor \frac{N_x}{N-M+1} \right\rfloor$ is the number of blocks to process an utterance of length N_x. The $2N \log_2(N)$ term in (6) is required for FFT of the impulse response $h_{ij}[n]$. For the impulse response $h_{ij}[n]$, we do not

Table 1: The profiling result for processing a single utterance using the "Room Simulator" on a local desktop machine

	Avg. Time per Utterance (*ms*)
Original *Room Simulator* in [1]	320.8 *ms*
`FFTW3` Real FFT Filtering	44.2 *ms*
+OLA filtering	19.4 *ms*
+20 dB RIR cut-off	14.3 *ms*

need to repeat it for every block. In our training set consisting of 22 million utterances mentioned above, the average N is 116,991 and the average N_h is 3,893. For this average N and N_h, the minimum value of C_{OLA} in (6) is 50.8 millions of real multiplications when $N = 2^{14}$. The optimal value of N minimizing C_{OLA} is different for different N_h and N_x values. For each filtering, we minimize C_{OLA} by evaluating this equation with different values of $N = 2^m$ where m is an integer.

2.3. Room Impulse Response Length Selection

The average length of the Room Impulse Response in the original room simulator is estimated to be 3893 samples, which corresponds to 0.243 *s*. We perform a simple RIR tail-cutoff by finding the RIR power threshold which is η *dB* below the maximum power of the RIR $h_{ij}[n]$:

$$p_{th} = \max\left\{h_{ij}^2[n]\right\} \times 10^{-\frac{\eta}{10}}. \tag{7}$$

Using p_{th}, we find the cut-off index n_c which is the smallest sample index beyond which all the trailing $h_{ij}[n]$ has power below the threshold p_{th}:

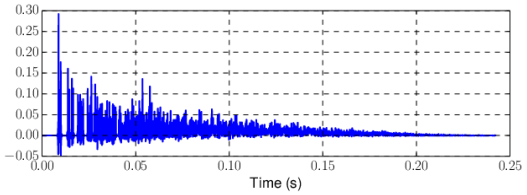
$$n_c = \min_m \left\{ m \middle| \max_{n>m} \left\{h^2[n]\right\} < p_{th} \right\}. \tag{8}$$

Then the final RIR cut-off $\widehat{h}_{ij}[n]$ is given by the following equation:

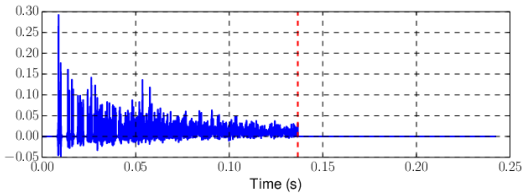
$$\widehat{h}_{ij}[n] = h_{ij}[n], \qquad 0 \le n \le n_c + 1. \tag{9}$$

Fig. 3 shows the original impulse responses and the corresponding RIR when the cutoff threshold of η is used. To reflect the typical case in our training set, we used the average room dimension, average microphone-to-target distance, and average T_{60} value among 3 million room configurations described in [1].

As shown in Table 1 and 2, the RIR tail cut-off at 20 *dB* shows relatively 35.6 % to 69.4 % speed improvement on a local desktop machine and on `Google Borg` cluster [30]. Fig. 4 shows the profiling results on a local machine with different RIR cutoff thresholds η. The local machine we used has a single `Intel(R) Xeon(R) E5-1650 @ 3.20GHz` CPU with 6 cores and 32 GB of memory. In Fig. 4, we observe that the computational cost becomes less for the OLA filtering case as we cut the tail portion of the RIR more. For the full FFT case, theoretically, it should remain almost constant regardless of the impulse response length, but due to variation in profiling measurement, there is some small variation.



(a)



(b)

Figure 3: The simulated room impulse responses generated in the "Room Simulator" described in Sec. 2: (a) The impulse response without the RIR tail cut-off, (b) with the RIR tail cut-off at 20 *dB*.

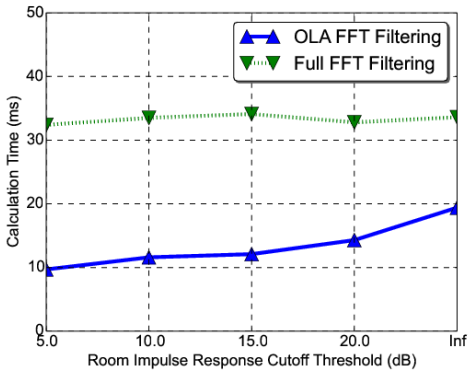


Figure 4: Profiling result on local desktop machine with different RIR cut-off threshold with and without OverLap Addition FIR filtering.

3. Experimental results

In this section, we present speech recognition results and CPU profiling results obtained using *Room Simulator* with different RIR cut-off thresholds. The acoustic modeling structure to obtain experimental results in this section is somewhat different from those described in [1, 9] for faster training. We use a single-channel 128 log-Mel feature whose window size is 32 *ms*. The interval between successive frame is 10 *ms*. The low and upper cutoff frequencies of the Mel filterbank are 125 *Hz* and 7500 *Hz* respectively. Since it has been shown that long-duration features represented by overlapping features are helpful [31], four frames are stacked together. Thus we use a context dependent feature consisting of 512 elements given by 128 (the size of the log-mel feature) x 4 (number of stacked frames). This input is down-sampled by a factor of three [32]. The feature is processed by a typical multi-layer Long Short-Term Memory (LSTM) [33, 34] acoustic model. We use 5-layer LSTMs with 768 units in each layer. The output of the final LSTM layer is passed to a softmax layer. The softmax layer has 8192 nodes corresponding to the number of tied context-dependent phones in our ASR system. The output state label is delayed by five frames, since it was observed that the information about future frames improves the prediction of the cur-

Table 2: The CPU usage portion of the FFT and *Room Simulator* with the respect to the entire CPU pipeline in Fig. 1 and relative speed up measured in terms of execution time with respect to our baseline in [1] on `Google Borg` cluster [30].

	Original system in [1]	`FFTW3` OLA Filter	`FFTW3` OLA Filter with 20 dB RIR Cut-off	`FFTW3` OLA Filter with 10 dB RIR Cut-off
FFT portion (%)	79.27 %	10.11 %	**6.23** %	5.08 %
Relative Speed Up in FFT Portion (%)	-	34.0 times	**57.6 times**	59.6 times
Entire Acoustic Simulation (%)	80.01 %	13.30 %	**9.69** %	8.05 %
Relative Speed Up in Acoustic Simulation Portion (%)	-	26.1 times	**37.3 times**	45.7 times

Table 3: Speech recognition experimental result in terms of Word Error Rates (WERs) with and without MTR using the room simulation system in [1] and different RIR cutoff thresholds.

	Baseline without MTR	MTR using One-time Batch Room Simulation	Baseline (On-the-fly Room Simulation)	20 dB RIR Cut-off	10 dB RIR Cut-off	5 dB RIR Cut-off
Original Test Set	11.70 %	12.07 %	11.95 %	**11.43** %	10.75 %	11.20 %
Simulated Noisy Set A	20.75 %	15.58 %	14.88 %	**13.96** %	13.59 %	14.99 %
Simulated Noisy Set B	50.78 %	22.47 %	20.64 %	**19.58** %	21.37 %	28.27 %
Device 1	52.56 %	22.39 %	21.69 %	**21.35** %	23.20 %	29.53 %
Device 2	51.59 %	22.12 %	21.62 %	**21.26** %	22.90 %	29.39 %
Device 3	54.89 %	23.42 %	22.29 %	**22.86** %	26.47 %	36.71 %
Device 3 (Noisy Condition)	72.09 %	36.21 %	35.88 %	**35.83** %	39.99 %	51.11 %
Device 3 (Multi-Talker Condition)	74.60 %	47.63 %	46.03 %	**46.21** %	48.36 %	58.31 %

Table 4: Average T_{60} Time of the Simulated Training Set and Simulated Test Sets A and B.

	Simulated Training Set	Simulated Noisy Set A	Simulated Noisy Set B
Average T_{60} *(s)*	0.482 *s*	0.167 *s*	0.479 *s*

rent frame [35, 36]. The acoustic model was trained using the Cross-Entropy (CE) loss as the objective function, using precomputed alignments for utterance as targets. To obtain results in Table 3, we trained for about 45 epochs. For training, we used an anonymized and hand-transcribed 22-million English utterances (18,000-hr) set. The training set is the same as what we used in [1, 9]. For evaluation, we used around 15-hour of utterances (13,795 utterances) obtained from anonymized mobile voice search data. We also generate noisy evaluation sets from this relatively clean voice search data. We use both simulated and rerecorded noisy sets. The average reverberation time in T_{60} of the simulated training set and two simulated test sets are shown in Table 4. These two simulated test sets are named *Simulated Noisy Set A* and *Simulated Noisy Set B* respectively.

Since our objective is deploying our speech recognition systems on far-field standalone devices such as Google Home, we rerecorded these evaluation sets using the actual hardware in far-field environment. Note that the actual Google Home hardware has two microphones with microphone spacing of 7.1 cm. In our experiments in this section, we selected the first channel out of two channel data. Three different devices were used in rerecording, and each device was placed in five different locations in an actual room resembling a real living room. These devices are listed in Table 3 as "Device 1", "Device 2", and "Device 3". As shown in Table 3, we observe that the RIR cutoff up to 20 *dB* threshold does not adversely affect the performance. However, if we cut the RIR to 5 *dB* threshold, then the performance under far-field environment becomes significantly worse. This observation also confirms that far-field speech recognition benefits from RIRs with sufficiently long tails in the training set. Table 2 shows how much CPU resource was used when training is done on `Google Borg` cluster [30] using the CPU/GPU training architecture in Fig. 1. We observe that if we use the `FFTW3`-based OLA filter using 20 *dB* RIR cutoff, we may obtain 57.6 times speed-up in the FFT portion. The entire speed-up of *Room Simulator* portion is 37.3 times.

4. Conclusions

In this paper, we describe how to efficiently implement an acoustic room simulator to generate large-scale simulated data for training deep neural networks. We implement an efficient Over-Lap Addition (OLA) based filtering using the open-source `FFTW3` library. We investigate into the effects of the Room Impulse Response (RIR) lengths. We conclude that we can cut the tail portions of RIRs whose power is less than 20 *dB* below the maximum power without sacrificing speech recognition accuracy. However, if we cut off RIR more than that, we observe it adversely affects the performance for reverberant cases. Using the approaches mentioned here, we could reduce the room simulator portion in the CPU usage down to 9.69 % in CPU/GPU training architecture. Profiling result shows that we obtain 22.4 times speed-up on a local desktop machine and 37.3 times speed up on `Google Borg` cluster.

5. References

[1] C. Kim, A. Misra, K.K. Chin, T. Hughes, A. Narayanan, T. Sainath, and M. Bacchiani, "Generation of simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in Google Home," in *INTERSPEECH-2017*, Aug. 2017, pp. 379–383.

[2] E. Variani, T. Bagby, E. McDermott, and M. Bacchiani, "End-to-end training of acoustic models for large vocabulary continuous speech recognition with tensorflow," in *INTERSPEECH-2017*, 2017, pp. 1641–1645. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2017-1284

[3] M. Seltzer, D. Yu, and Y.-Q. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Int. Conf. Acoust. Speech, and Signal Processing*, 2013, pp. 7398–7402.

[4] D. Yu, M. L. Seltzer, J. Li, J.-T. Huang, and F. Seide, "Feature learning in deep neural networks - studies on speech recognition tasks," in *Proceedings of the International Conference on Learning Representations*, 2013.

[5] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," in *Deep Learning and Unsupervised Feature Learning NIPS Workshop*, 2011.

[6] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, Nov.

[7] T. Sainath, R. J. Weiss, K. W. Wilson, B. Li, A. Narayanan, E. Variani, M. Bacchiani, I. Shafran, A. Senior, K. Chin, A. Misra, and C. Kim, "Multichannel signal processing with deep neural networks for automatic speech recognition," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, Feb. 2017.

[8] ——, "Raw Multichannel Processing Using Deep Neural Networks," in *New Era for Robust Speech Recognition: Exploiting Deep Learning*, S. Watanabe, M. Delcroix, F. Metze, and J. R. Hershey, Ed. Springer, Oct. 2017.

[9] B. Li, T. Sainath, A. Narayanan, J. Caroselli, M. Bacchiani, A. Misra, I. Shafran, H. Sak, G. Pundak, K. Chin, K-C Sim, R. Weiss, K. Wilson, E. Variani, C. Kim, O. Siohan, M. Weintraub, E. McDermott, R. Rose, and M. Shannon, "Acoustic modeling for Google Home," in *INTERSPEECH-2017*, Aug. 2017, pp. 399–403.

[10] C. Kim and R. M. Stern, "Power-Normalized Cepstral Coefficients (PNCC) for Robust Speech Recognition," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, pp. 1315–1329, July 2016.

[11] U. H. Yapanel and J. H. L. Hansen, "A new perceptually motivated MVDR-based acoustic front-end (PMVDR) for robust automatic speech recognition," *Speech Communication*, vol. 50, no. 2, pp. 142–152, Feb. 2008.

[12] C. Kim and R. M. Stern, "Feature extraction for robust speech recognition based on maximizing the sharpness of the power distribution and on power flooring," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, March 2010, pp. 4574–4577.

[13] ——, "Nonlinear enhancement of onset for robust speech recognition," in *INTERSPEECH-2010*, Sept. 2010, pp. 2058–2061.

[14] C. Kim, K. Chin, M. Bacchiani, and R. M. Stern, "Robust speech recognition using temporal masking and thresholding algorithm," in *INTERSPEECH-2014*, Sept. 2014, pp. 2734–2738.

[15] T. Nakatani, N. Ito, T. Higuchi, S. Araki, and K. Kinoshita, "Integrating DNN-based and spatial clustering-based mask estimation for robust MVDR beamforming," in *IEEE Int. Conf. Acoust., Speech, Signal Processing*, March 2017, pp. 286–290.

[16] T. Higuchi and N. Ito and T. Yoshioka and T. Nakatani, "Robust MVDR beamforming using time-frequency masks for online/offline ASR in noise," in *IEEE Int. Conf. Acoust., Speech, Signal Processing*, March 2016, pp. 5210–5214.

[17] H. Erdogan, J. R. Hershey, S. Watanabe, M. Mandel, J. Roux, "Improved MVDR Beamforming Using Single-Channel Mask Prediction Networks," in *INTERSPEECH-2016*, Sept 2016, pp. 1981–1985.

[18] C. Kim and K. K. Chin, "Sound source separation algorithm using phase difference and angle distribution modeling near the target," in *INTERSPEECH-2015*, Sept. 2015, pp. 751–755.

[19] C. Kim, K. Kumar, B. Raj, and R. M. Stern, "Signal separation for robust speech recognition based on phase difference information obtained in the frequency domain," in *INTERSPEECH-2009*, Sept. 2009, pp. 2495–2498.

[20] C. Kim, C. Khawand, and R. M. Stern, "Two-microphone source separation algorithm based on statistical modeling of angle distributions," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, March 2012, pp. 4629–4632.

[21] C. Kim, K. Kumar, and R. M. Stern, "Binaural sound source separation motivated by auditory processing," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, May 2011, pp. 5072–5075.

[22] J. Allen and D. Berkley, "Image method for efficiently simulating small-room acoustics," *J. Acoust. Soc. Am.*, vol. 65, no. 4, pp. 943–950, April 1979.

[23] E. A. Lehmann, A. M. Johansson, and S. Nordholm, "Reverberation-time prediction method for room impulse responses simulated with the image-source model," in *2007 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, Oct. 2007, pp. 159–162.

[24] S. G. McGovern. room impulse response generator. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/5116-room-impulse-response-generator

[25] ——, "Fast image method for impulse response calculations of box-shaped rooms," *Applied Acoustics*, vol. 70, no. 1, pp. 182 – 189, 2009.

[26] M. Borgerding, "Kiss fft version 1.2.9," https://sourceforge.net/projects/kissfft/, 2010.

[27] A. V. Oppenheim and R. W.Scafer, with .J. R. Buck, *Discrete-time Signal Processing*, 2nd ed. Englewood-Cliffs, NJ: Prentice-Hall, 1998.

[28] G. Guennebaud, B. Jacob et al., "Eigen v3," http://eigen.tuxfamily.org, 2010.

[29] R. Crochiere, "A weighted overlap-add method of short-time Fourier analysis/synthesis," *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. 28, no. 1, pp. 99–102, feb 1980.

[30] A. Verma, L. Pedrosa, M. R. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at Google with Borg," in *Proceedings of the European Conference on Computer Systems (EuroSys)*, Bordeaux, France, 2015.

[31] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and Accurate Recurrent Neural Network Acoustic Models for Speech Recognition," in *INTERSPEECH-2015*, Sept. 2015, pp. 1468–1472.

[32] G. Pundak and T. N. Sainath, "Lower Frame Rate Neural Network Acoustic Models," 2016, pp. 22–26. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2016-275

[33] S. Hochreiter and Jürgen Schmidhuber, "Long Short-term Memory," *Neural Computation*, no. 9, pp. 1735–1780, Nov. 1997.

[34] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *IEEE Int. Conf. Acoust., Speech and Signal Processing*, Apr. 2015, pp. 4580–4584.

[35] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *INTERSPEECH-2014*, Sept. 2014, pp. 338–342.

[36] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.