Harmonizing Audio and Human Interaction: Enhancement, Analysis, and Application of Audio Signals via Machine Learning Approaches

Ruilin Xu

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2024

Abstract

Harmonizing Audio and Human Interaction: Enhancement, Analysis, and Application of Audio Signals via Machine Learning Approaches

Ruilin Xu

In this thesis, we tackle key challenges in processing audio signals, specifically focusing on speech and music. These signals are crucial for human interaction with both the environment and machines. Our research addresses three core topics: speech denoising, speech dereverberation, and music-dance generation, each of which plays a vital role in enhancing the harmony between audio and human interaction.

Leveraging machine learning and *human-centric* approaches inspired by classical algorithms, we develop methods to mitigate common audio degradations, such as additive noise and multiplicative reverberation, delivering high-quality audio suitable for human use and applications. Furthermore, we introduce a real-time, music-responsive system for generating 3D dance animations, advancing the integration of audio signals with human engagement.

The first focus of our thesis is the elimination of additive noise from audio signals by focusing on short pauses, or *silent intervals*, in human speech. These brief pauses provide key insights into the noise profile, enabling our model to dynamically reduce ambient noise from speech. Tested across diverse datasets, our method outperforms traditional and audiovisual denoising techniques, showcasing its effectiveness and adaptability across different languages and even musical contexts.

In the second work of our research, we address reverberation removal from audio signals, a

task traditionally reliant on knowing the environment's exact impulse response—a requirement often impractical in real-world settings. Our novel solution combines the strengths of classical and learning-based approaches, tailored for online communication contexts. This human-centric method includes a one-time personalization step, adapting to specific environments and human speakers. The two-stage model, integrating feature-based Wiener deconvolution and network refinement, has shown through extensive experiments to outperform current methods, both in effectiveness and user preference.

Transitioning from foundational audio signal enhancement and analysis to a more dynamic realm, our research culminates in a novel, interactive system for real-time 3D human dance generation. Contrasting with the passive human-centric assumptions of our previous works, this final work actively engages users, enabling direct interaction with a system that synchronizes expressive dance movements to live music, spanning various musical elements like type, tempo, and energy. This innovative approach, diverging from traditional choreography methods, leverages spontaneous improvisation to generate unique dance sequences. These sequences, a mix of pre-recorded choreographies and algorithm-generated transitions, adapt to real-time audio inputs, offering customization through personal 3D avatars. This system's user-centric design and interactivity are validated by user studies, confirming its effectiveness in creating an immersive and engaging user experience.

Table of Contents

List of Figures

List of Tables

Acknowledgements

Six years of PhD has been a long and incredible journey. When I think back, I'm amazed at how many significant events have shaped my life during this time. It's been a mix of ups and downs in my personal life, alongside moments of happiness and frustration in my research. Now, as I'm writing this thesis, there is nothing but gratitude remaining in my heart. I could not have reached this point without the consistent support, help, care, and love from everyone I've met along the way. They all hold a special place in my heart and will continue to do so for the rest of my life.

Advisors and mentors

First and foremost, my deepest thanks go to my advisor, Shree K. Nayar. Thank you for introducing me in to the field of Computer Vision, and if not for your class during my Master's, I might never have started this PhD journey. Thank you for spending time with me every day in my early PhD days, showing me how to do research step by step. Thank you for sharing philosophical updates with me. Thank you for teaching me not just about research but also about being a better person. Thank you for telling me how to succeed as a PhD student and how important Diet Coke is in contributing to success (I'm still on team regular Coke, but I might have to switch soon as I'm getting older). Thanks for being there during my personal hard times. I couldn't have made it without your care and support. I'm grateful for your honest and straight-up feedback when I made mistakes. Thank you for willing to spend time with me, giving me all the academic and life advice you can when I was struggling. Thank you for all the restaurant recommendations around the campus such as Thai Market, Pisticci, and La Salle Dumpling Room. And thank you for making Apple Tree the designated cafeteria for our lab. Thanks for trying out the "Coke Chicken Wings"

recipe I shared, and for getting into Lao Gan Ma chili sauce ("Old Godmother" - what a name!). All in all, thank you from the bottom of my heart. It has been an honor to join the research family and become one of your students.

I also want to express my heartfelt thanks to Changxi Zheng for guiding me through the challenges of research. Your constant availability and willingness to help whenever I needed guidance in my work has been incredibly valuable. I have always been amazed by your extensive knowledge across various fields, including Computer Vision, Computer Graphics, Machine Learning, Robotics, Physics, and even more to list. Your passion and boldness in pursuing research ideas have been a great source of inspiration, constantly motivating me to dive deeper into my own research topics.

I want to thank Carl M. Vondrick for your brilliant research ideas and insights. A huge thanks for allowing me to use your servers for my research, which has been immensely helpful.

A special thank to Gurunandan Krishnan, my mentor during my research internships at Snap. Guru, I deeply appreciate all the time and effort you invested in helping me with my research and editing papers. Your industry knowledge and experience have opened my eyes to the possibilities that lie ahead post-PhD.

Lastly, I want to extend my thanks to Mohit Gupta for your invaluable remote advice during our collaboration. Your meticulous approach to research has significantly influenced my research journey, leaving a lasting impression on my work.

Colleagues and friends

Jeremy Klotz, Matthew Beveridge and Nikhil Nanda, it was a pleasure meeting you guys during the final stretch of my PhD journey. You guys are not only smart and dedicated, but also bring a great sense of humor to our lab. I'm truly grateful for your enthusiasm in driving and pushing forward the lab purchases and renovations. And a big thanks for significantly improving the coffee quality in our lab – that made a huge difference! You guys are awesome.

Thank you, Parita Pooj, for being the only accompany in the lab during the early days of my PhD. Our discussions on research and personal stories have been invaluable.

Thank you, Brian A. Smith, for your kind guidance in both lab matters and research. Your mini

"User Study 101" lecture was instrumental in shaping my approach to designing user studies for my research projects.

I'm also grateful to Bing Zhou and Vu An Tran from Snap for your patient assistance in both research and engineering. Your knowledge, responsiveness, and inspiration have been indispensable.

Rundi Wu, thanks for sharing your machine learning expertise and coding tips. Your depth of knowledge and skill always amazed me.

For Physics insights, Ziwei Zhu has been my go-to. Thank you for expanding my understanding with concepts and formulas well beyond my initial grasp.

A special shout-out to Colman Leung and Ziwei Zhu for the music sessions – we should definitely think of a band name!

To my weekend and vacation buddies – Honglin Chen, Peter Yichen Chen, Colman Leung, Ruoshi Liu, Qijia Shao, Rundi Wu, Chang Xiao, Xiaofeng Yan, and Ziwei Zhu – thank you for the adventures in nature, food explorations, and karaoke sessions. You've all brightened my PhD journey with beautiful and colorful memories.

I'd like to extend my gratitude to my fellow Columbia graphics group members: Honglin Chen, Raymond Yun Fei, Dingzeyu Li, Henrique Teles Maia, Oded Stein, Chang Xiao, Rundi Wu, and Ziwei Zhu, for their support and collaboration.

Additionally, I would like to thank Mia Chiquier, Basile Van Hoorick, Ruoshi Liu, and Chengzhi Mao from Carl M. Vondrick's lab.

Family

Mom and Dad, thank you for creating a loving and pressure-free environment where I could grow up surrounded by warmth and support. Thank you for always willing to help out and giving your advice and guidance yet respecting my own decisions wholeheartedly. I cannot be who I am without your influence. I'm deeply grateful and I love you both.

Dad, your passion for math, engineering, and research has been the cornerstone of my journey towards this PhD. From those early days in elementary school, explaining calculus on a philosophical level as a way to view the world, you encouraged me to think deeply and see things from different

angles. Your life stories, both successes and failures, have been invaluable lessons for me. Thank you for always believing in me, especially during times of confusion, struggle and uncertainty. Thank you for always being there, despite the 12-hour time difference. Thank you for being the one who I can always count on and trust. Thank you for encouraging me to be brave enough to embrace my true self.

Mom, thank you for your unconditional love. Thank you for being the one who always care about my health. Thank you for believing in the significance of my research, even though you have always believed that my research can help you speeding up your slow laptop and making it run smoother without any hiccups. Thank you for your dedication in taking me to violin lessons when I was a kid, regardless of the weather. Violin has become an integral part of my life, a faithful companion in times of joy and sorrow.

To my grandparents, who passed away when I was young, your love has left an indelible mark on my heart. I miss you all.

Finally, my sincere thanks go to everyone who has been a part of my life's journey. Each one of you has contributed to shaping the person I am today. Your presence and influence, in all its forms, has been irreplaceable and a guiding force in my journey.

To my beloved family;

To my younger self;

To my violin.

Chapter 1: Introduction

Audio signals, comprising both speech and music, are fundamental to human perception and interaction with the environment. They facilitate the exchange of information and emotions in both human-to-human and human-to-machine communications. In the digital era, automated systems such as Intelligent Personal Assistants (IPAs), whether in the form of mobile applications or hardware appliances like smart speakers, have become integral to daily life, transforming human-machine communication into an essential component of modern existence.

Speech, in particular, serves as a natural medium for communication between humans and machines. Voice-activated digital assistants in smart speakers, integrated within the framework of the Internet of Things (IoT), exemplify this synergy. These assistants, such as Amazon Alexa [1], Google Home Assistant [2], and Apple Siri [3], enable a range of interactions, from playing music to controlling home appliances. However, the effectiveness of human-machine communication hinges on the machine's ability to process and understand human speech — a task complicated by the impurities and distortions inherent in real-world audio signals. Ambient noises, dynamic sound events, structured sounds like music and conversations, and reverberations from enclosed spaces all contribute to this complexity.

As environments become increasingly intricate, classic audio processing methods [4, 5, 6, 7] reveal their limitations in handling complex scenarios and often exhibit slower processing times. In contrast, machine learning techniques [8] have gained immense popularity due to their ability to outperform classical methods when provided with substantial data. These techniques have been widely adopted across various speech-related tasks, including automatic speech recognition (ASR)[9], speech synthesis[10], speech enhancement [11], speaker separation [12], and spoken emotion recognition [13]. Despite their success, these learning-based methods often require vast amounts of training data, which can be challenging to obtain. Furthermore, these methods have not

fully explored the intrinsic properties of speech signals, limiting their effectiveness in more complex real-world scenarios.

Music, another significant type of audio signal, also requires varying degrees of connection between humans and machines. These connections can range from simple interactions such as music playback requests to more complex tasks like music recognition through IPAs. While these interactions may appear straightforward, they demand substantial backend processing, involving algorithms such as music information retrieval [14] and instrument separation [15]. A more complex form of human-machine interaction involving music is the task of 3D dance generation. Traditionally, producing 3D dance animations has been restricted to aligning pre-recorded dances with specific music tracks that convey similar emotions. This limitation arises because classical algorithms for both music/audio signal processing and 3D dance generation [16, 17, 18] are non-real-time, limiting the scope of music-dance generation to pre-choreographed routines. Furthermore, this process can be time-consuming, thereby diminishing the sense of interaction between humans and machines.

Advances in machine learning and the effective utilization of large dance datasets have enabled the development of learning-based approaches [19, 20, 21, 22] that generate new dance motions conditioned on music. These approaches significantly reduce the repetitiveness seen in traditional methods. However, they typically require vast amounts of training data, and the generated dances can sometimes be physically implausible or lack visual appeal. Moreover, the level of human-machine interaction remains limited, as most of these methods cannot operate in real-time.

In short, this growing reliance on data-centric approaches, which are less focused on human interaction, prompts critical reflection. This thesis aims to explore alternative approaches by considering key questions such as, **"Can we draw inspiration from classical methods rather than relying solely on data-centric trends?"** and **"Can we leverage the intrinsic properties of data to reduce dependence on massive datasets, thereby fostering a more human-centric approach?"**

In response to these questions, this thesis revisits popular research areas in audio signal processing through a *human-centric* lens. By integrating insights from classical algorithms with modern machine learning techniques, our goal is to enhance the understanding of audio signals and explore

their application in innovative, interactive ways.

Specifically, this thesis focuses on both speech and music signals, introducing methods that deeply analyze and understand these signals. By combining classical and learning-based approaches, these methods emphasize the integration of human interaction — both passive and active — while reducing reliance on massive datasets. The research spans three distinct fields: speech denoising (**Part I**), speech dereverberation (**Part II**), and music-dance generation (**Part III**).

Part I: Listening to Sounds of Silence for Speech Denoising. The initial focus of our thesis is the elimination of additive noise from audio signals, utilizing a unique approach that capitalizes on dense yet brief pauses commonly found in human speech—termed *silent intervals*. These intervals, often overlooked, are essential in our methodology, providing a window into the underlying noise characteristics within the audio signals. By analyzing these silent intervals, our model learns to dynamically adapt to the time-varying features of ambient noise, effectively suppressing it from the speech signal, as inspired by the classic spectral subtraction algorithm. Our experiments across multiple datasets have validated the crucial role of silent interval detection in speech denoising. Remarkably, this method not only surpasses several state-of-the-art denoising techniques, which rely solely on audio input, but also outperforms methods that utilize additional audiovisual information. Furthermore, our approach demonstrates exceptional generalization capabilities, effectively denoising across various languages and even in musical contexts, highlighting its versatility and robustness.

Part II: Personalized Dereverberation of Speech. The second aspect of our research addresses the challenge of reverberation—a phenomenon akin to the superposition of numerous, minutely delayed copies of a clean audio signal. Traditional dereverberation methods, aimed at removing these reverberations from audio signals, are effective *only* when the exact impulse response of the environment is known. Yet, in real-world settings, obtaining such a precise impulse response is often unfeasible, as it varies with environmental conditions and the speaker's location. Learning-based blind methods, while offering an alternative, fall short in consistently achieving adequate dereverberation across diverse environments. To overcome these limitations, we developed a unique environment- and speaker-specific approach that synthesizes the benefits of both classic Wiener

and learning-based methods. This approach is particularly tailored to the human-centric context of online communications, where individuals frequently engage in virtual meetings in stable office environments. Under this assumption, our method involves a one-time personalization step, enabling the model to generalize a single measured impulse response throughout its spatial vicinity effectively. Our extensive experimental evaluations demonstrate that this approach not only surpasses existing state-of-the-art methods in both quantitative and qualitative measures but also receives overwhelming preference from users, as evidenced by our user study results.

Part III: DanceCraft: A Music-Reactive Real-time Dance Improv System. While the enhancement and analysis of audio signals are foundational elements of our research, the above works predominantly involve passive human-centric assumptions, where human involvement, though essential, remains largely indirect. In contrast, our final work shifts towards a more interactive and genuinely human-centric approach. Here, we actively involve users in real-time with our system, focusing on direct human interaction and engagement. In this work, we harness audio signals, particularly music, to power an innovative system that generates natural and expressive 3D human dances in real-time. This system adeptly synchronizes dance movements with various musical elements, including type, tempo, and energy levels, achieving an integrated audio-visual experience that is both enjoyable and engaging for users.

Inspired by the classic graph-based approach, the dance sequences generated by our system are a unique blend of previously captured dance sequences as well as randomly triggered generative transitions between different dance sequences. Due to these randomized transitions, two generated dances, even for the same music, tend to appear very different. Furthermore, the system's design emphasizes user interactivity and customization, allowing individuals to integrate their personal 3D avatars into the dance experience. The effectiveness and appeal of our system are substantiated by user studies, which highlight its success in delivering an immersive and highly engaging user experience, as reflected in the overwhelmingly positive feedback received.

1.1 Contributions

In summary, this thesis offers the following key contributions:

- Development of an innovative end-to-end deep learning approach for speech denoising that effectively harnesses the inherent physical structure of human speech. Drawing inspiration from spectral subtraction, this approach models comprehensive noise profiles by implicitly learning from non-speech segments densely distributed across human speech. (**Part I**)

- Creation of a large-scale noisy speech dataset compiled from the AVSPEECH, `AudioSet`, and DEMAND datasets. This dataset, designed for training and evaluating speech denoising tasks, features signal-to-noise ratios (SNR) as low as -10, making it the first public dataset of its kind with such intense noise levels. (**Part I**)

- Proposal of a novel, human-centric end-to-end model for speech dereverberation, leveraging the natural constraints inherent in online communications. Integrating classic principles like Wiener deconvolution with advanced deep learning techniques, this model produces exceptional dereverberation tailored to specific users and environments. (**Part II**)

- Introduction of a simple yet innovative personalization step in the dereverberation process that allows the proposed dereverberation method to generalize room impulse responses (RIR) to its spatial neighborhood. (**Part II**)

- Development of the first human-interactive and real-time system for generating 3D improvisational dances responsive to any live music. Moreover, this system is distinguished by its high level of personalization capabilities, allowing users to import their personal 3D avatars and have them dance to any music played in the environment. (**Part III**)

- Compilation of the first comprehensive large-scale 3D dance dataset specifically tailored for improvisational dance. This dataset, covering a diverse array of dance styles and genres, is synchronized with various music styles and performed by numerous dancers, setting a new benchmark in the field. (**Part III**)

1.2 Thesis organization

The structure of the rest of this thesis is organized as follows:

Part I focuses on the task of speech denoising. This part includes modified versions of two key publications:

- R. Xu, R. Wu, Y. Ishiwaka, C. Vondrick, and C. Zheng, "Listening to sounds of silence for speech denoising", in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. Nips'20, Vancouver, BC, Canada: Curran Associates Inc., 2020, ISBN: 9781713829546.

- J. Xiang, Y. Zhu, R. Wu, R. Xu, Y. Ishiwaka, and C. Zheng, "Dynamic sliding window for realtime denoising networks", in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 361–365. DOI: 10.1109/icassp43922.2022.9747168.

Part II delves into the task of speech dereverberation. This section is based on the modified version of the following publication:

- R. Xu, G. Krishnan, C. Zheng, and S. K. Nayar, "Personalized dereverberation of speech", in *Proceedings of INTERSPEECH 2023*, 2023, pp. 3859–3863. DOI: 10.21437/Interspeech.2023-2055.

Part III explores the field of music-dance generation. It is grounded in the modified version of this publication:

- R. Xu, V. A. Tran, S. K. Nayar, and G. Krishnan, "Dancecraft: A music-reactive real-time dance improv system", in *Proceedings of the 9th International Conference on Movement and Computing*, ser. Moco '24, Utrecht, Netherlands: Association for Computing Machinery, 2024, ISBN: 9798400709944. DOI: 10.1145/3658852.3659078.

Finally, Chapter 21 concludes the thesis, reflecting on the contributions and discussing potential future directions for the research presented.

Part I

Listening to Sounds of Silence for Speech Denoising

Chapter 2: Overview

Noise is everywhere. When we listen to someone speak, the audio signals we receive are never pure and clean, always contaminated by all kinds of noises—cars passing by, spinning fans in an air conditioner, barking dogs, music from a loudspeaker, and so forth. To a large extent, people in a conversation can effortlessly filter out these noises [27]. In the same vein, numerous applications, ranging from cellular communications to human-robot interaction, rely on speech denoising algorithms as a fundamental building block.

Despite its vital importance, algorithmic speech denoising remains a grand challenge. Provided an input audio signal, speech denoising aims to separate the foreground (speech) signal from its additive background noise. This separation problem is inherently ill-posed. Classic approaches such as spectral subtraction [4, 5, 28, 29, 30] and Wiener filtering [6, 7] conduct audio denoising in the spectral domain, and they are typically restricted to stationary or quasi-stationary noise. In recent years, the advance of deep neural networks has also inspired their use in audio denoising. While outperforming the classic denoising approaches, existing neural-network-based approaches use network structures developed for general audio processing tasks [31, 32, 33] or borrowed from other areas such as computer vision [34, 35, 36, 37, 38] and generative adversarial networks [39, 40]. Nevertheless, beyond reusing well-developed network models as a black box, a fundamental question remains: *What natural structures of speech can we leverage to mold network architectures for better performance on speech denoising?*

2.1 Key insight: time distribution of silent intervals

Motivated by this question, we revisit one of the most widely used audio denoising methods in practice, namely the spectral subtraction method [4, 5, 28, 29, 30]. Implemented in many commercial software such as Adobe Audition [41], this classical method requires the user to specify

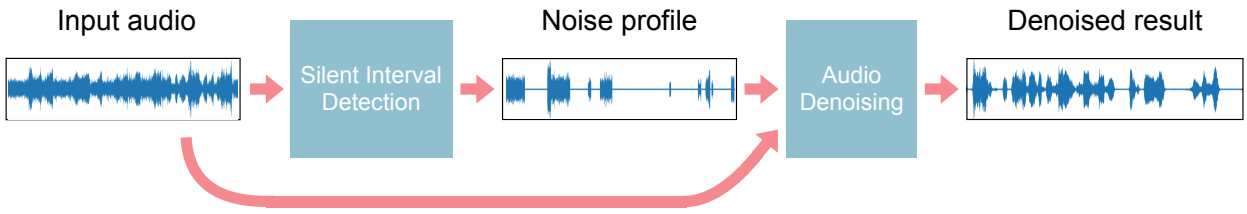


Figure 2.1: **Audio denoising.** We present an audio-only denoising model that, given a noisy audio signal as input, detects out silent intervals, which gives us a partial noise profile. The model then uses the noise profile as a reference to perform the denoising task.

a time interval during which the foreground signal is absent. We call such an interval a *silent interval*. A silent interval is a time window that exposes pure noise. The algorithm then learns from the silent interval the noise characteristics, which are in turn used to suppress the additive noise of the entire input signal (through subtraction in the spectral domain).

Yet, the spectral subtraction method suffers from two major shortcomings: i) it requires user specification of a silent interval, that is, not fully automatic; and ii) the single silent interval, although undemanding for the user, is insufficient in presence of *non-stationary* noise—for example, a background music. Ubiquitous in daily life, non-stationary noise has time-varying spectral features. The single silent interval reveals the noise spectral features only in that particular time span, thus inadequate for denoising the entire input signal. The success of spectral subtraction pivots on the concept of silent interval; so do its shortcomings.

In this part of the thesis, we introduce a deep network for speech denoising that tightly integrates silent intervals, and thereby overcomes many of the limitations of classical approaches. Our goal is not just to identify a single silent interval, but to find as many as possible silent intervals over time. Indeed, silent intervals in speech appear in abundance: psycho-linguistic studies have shown that there is almost always a pause after each sentence and even each word in speech [42, 43]. Each pause, however short, provides a silent interval revealing noise characteristics local in time. All together, these silent intervals assemble a time-varying picture of background noise, allowing the neural network to better denoise speech signals, even in presence of non-stationary noise (see Figure 2.2).

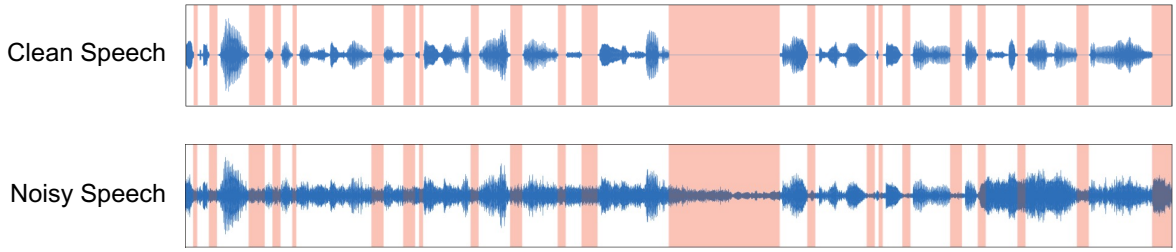


Figure 2.2: **Silent intervals over time. (top)** A speech signal has many natural pauses. Without any noise, these pauses are exhibited as silent intervals (highlighted in red). **(bottom)** However, most speech signals are contaminated by noise. Even with mild noise, silent intervals become overwhelmed and hard to detect. If robustly detected, silent intervals can help to reveal the noise profile over time.

In short, to interleave neural networks with established denoising pipelines, we propose a network structure consisting of three major components (see Figure 4.1): **i)** one dedicated to silent interval detection, **ii)** another that aims to estimate the full noise from those revealed in silent intervals, akin to an inpainting process in computer vision [44], and **iii)** yet another for cleaning up the input signal.

2.2 Summary of results

Our neural-network-based denoising model accepts a single channel of audio signal and outputs the cleaned-up signal. Unlike some of the recent denoising methods that take as input audiovisual signals (i.e., both audio and video footage), our method can be applied in a wider range of scenarios (e.g., in cellular communication). We conducted extensive experiments, including ablation studies to show the efficacy of our network components and comparisons to several state-of-the-art denoising methods. We also evaluate our method under various signal-to-noise ratios—even under strong noise levels that are not tested against in previous methods. We show that, under a variety of denoising metrics, our method consistently outperforms those methods, including those that accept only audio input (like ours) and those that denoise based on audiovisual input.

The pivotal role of silent intervals for speech denoising is further confirmed by a few key results. Even without supervising on silent interval detection, the ability to detect silent intervals naturally emerges in our network. Moreover, while our model is trained on English speech only, with no

additional training it can be readily used to denoise speech in other languages (such as Chinese, Japanese, and Korean).

Chapter 3: Related Work

3.1 Speech denoising

Speech denoising [45] is a fundamental problem studied over several decades. Spectral subtraction [4, 5, 28, 29, 30] estimates the clean signal spectrum by subtracting an estimate of the noise spectrum from the noisy speech spectrum. This classic method was followed by spectrogram factorization methods [46]. Wiener filtering [6, 7] derives the enhanced signal by optimizing the mean-square error. Other methods exploit pauses in speech, forming segments of low acoustic energy where noise statistics can be more accurately measured.For example, minima-tracking algorithms [47, 48] perform the denoising task by finding the minimum of the noisy speech power to track the aforementioned low acoustic segments. Methods such as [49, 50, 51, 52, 53] improve the noise estimation by using recursive averaging, which takes both the noise segments and the speech segments into account, resolving the issue of noise power overestimation. Statistical model-based methods [54, 55] and subspace algorithms [56, 57] are also studied.

Applying neural networks to audio denoising dates back to the 80s [58, 59]. With increased computing power, deep neural networks are often used [60, 61, 62, 63]. Long short-term memory networks (LSTMs) [64] are able to preserve temporal context information of the audio signal [65], leading to strong results [31, 32, 33]. Leveraging generative adversarial networks (GANs) [66], methods such as [39, 40] have adopted GANs into the audio field and have also achieved strong performance.

Audio signal processing methods operate on either the raw waveform or the spectrogram by Short-time Fourier Transform (STFT). Some work directly on waveform [67, 68, 69, 70], and others use Wavenet [10] for speech denoising [71, 11, 72]. Many other methods such as [73, 74, 75, 76, 77, 78, 79] work on audio signal's spectrogram, which contains both magnitude and phase information. There are works discussing how to use the spectrogram to its best potential [80, 81], while one of the

disadvantages is that the inverse STFT needs to be applied. Meanwhile, there also exist works [82, 83, 84, 85, 86, 87, 88] investigating how to overcome artifacts from time aliasing.

Speech denoising has also been studied in conjunction with computer vision due to the relations between speech and facial features [89]. Methods such as [34, 35, 36, 37, 38] utilize different network structures to enhance the audio signal to the best of their ability. Adeel et al. [90] even utilize lip-reading to filter out the background noise of a speech.

3.2 Deep learning for other audio processing tasks

Deep learning is widely used for lip reading, speech recognition, speech separation, and many audio processing or audio-related tasks, with the help of computer vision [91, 92, 93, 94]. Methods such as [95, 96, 97] are able to reconstruct speech from pure facial features. Methods such as [98, 99] take advantage of facial features to improve speech recognition accuracy. Speech separation is one of the areas where computer vision is best leveraged. Methods such as [100, 91, 101, 102] have achieved impressive results, making the previously impossible speech separation from a single audio signal possible. Recently, Zhang et al. [103] proposed a new operation called Harmonic Convolution to help networks distill audio priors, which is shown to even further improve the quality of speech separation.

Chapter 4: Learning Speech Denoising

We present a neural network that harnesses the time distribution of silent intervals for speech denoising. The input to our model is a spectrogram of noisy speech [104, 105, 106], which can be viewed as a 2D image of size $T \times F$ with two channels, where T represents the time length of the signal and F is the number of frequency bins. The two channels store the real and imaginary parts of STFT, respectively. After learning, the model will produce another spectrogram of the same size as the noise suppressed.

We first train our proposed network structure in an end-to-end fashion, with only denoising supervision (Section 4.2); and it already outperforms the state-of-the-art methods that we compare against. Furthermore, we incorporate the supervision on silent interval detection (Section 4.3) and obtain even better denoising results (see Chapter 5).

4.1 Network structure

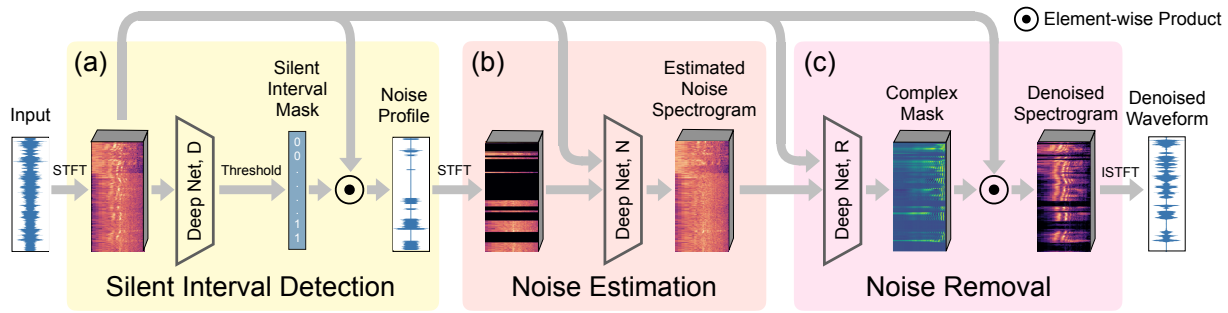


Figure 4.1: **Our audio denoise network.** Our model has three components: **(a)** one that detects silent intervals over time, and outputs a noise profile observed from detected silent intervals; **(b)** another that estimates the full noise profile, and **(c)** yet another that cleans up the input signal.

Classic denoising algorithms work in three general stages: silent interval specification, noise feature estimation, and noise removal. We propose to interweave learning throughout this process: we rethink each stage with the help of a neural network, forming a new speech denoising model. Since we can chain these networks together and estimate gradients, we can efficiently train the model with large-scale audio data. Figure 4.1 illustrates this model, which we describe below.

4.1.1 Silent interval detection

The first component is dedicated to detecting silent intervals in the input signal. The input to this component is the spectrogram of the input (noisy) signal $\boldsymbol{x}$. The spectrogram $\boldsymbol{s_x}$ is first encoded by a 2D convolutional encoder into a 2D feature map, which is in turn processed by a bidirectional LSTM [64, 107] followed by two fully-connected (FC) layers (see network details in Section 4.4). The bidirectional LSTM is suitable for processing time-series features resulting from the spectrogram [108, 109, 110, 101], and the FC layers are applied to the features of each time sample to accommodate variable length input. The output from this network component is a vector $\mathrm{D}(\boldsymbol{s_x})$. Each element of $\mathrm{D}(\boldsymbol{s_x})$ is a scalar in [0,1] (after applying the sigmoid function), indicating a confidence score of a small time segment being silent. We choose each time segment to have $1/30$ second, small enough to capture short speech pauses and large enough to allow robust prediction (see Section 4.3).

The output vector $\mathrm{D}(\boldsymbol{s_x})$ is then expanded to a longer mask, which we denote as $\boldsymbol{m}(\boldsymbol{x})$. Each element of this mask indicates the confidence of classifying each sample of the input signal $\boldsymbol{x}$ as pure noise (see Figure 4.2-e). With this mask, the noise profile $\tilde{\boldsymbol{x}}$ exposed by silent intervals are estimated by an element-wise product, namely $\tilde{\boldsymbol{x}} \coloneqq \boldsymbol{x} \odot \boldsymbol{m}(\boldsymbol{x})$.

4.1.2 Noise estimation

The signal $\tilde{\boldsymbol{x}}$ resulted from silent interval detection is noise profile exposed only through a series of time windows (see Figure 4.2-e)—but not a complete picture of the noise. However, since the input signal is a superposition of clean speech signal and noise, having a complete noise profile would ease the denoising process, especially in presence of nonstationary noise. Therefore, we also

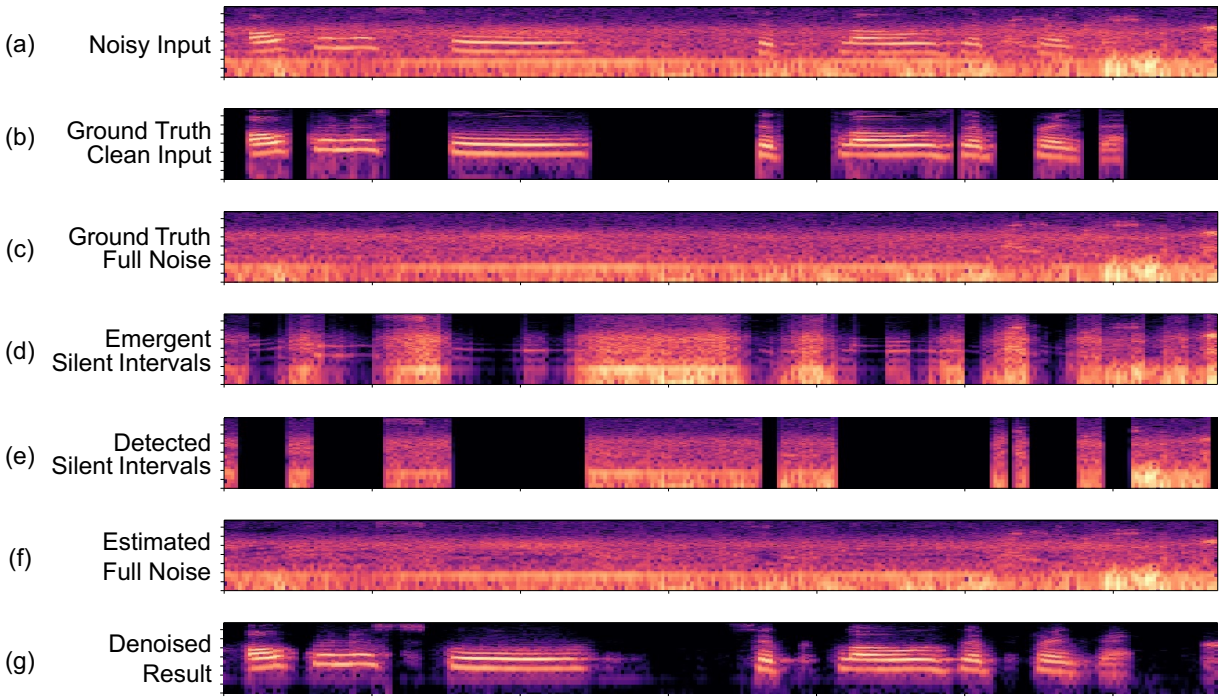


Figure 4.2: **Example of intermediate and final results.** **(a)** The spectrogram of a noisy input signal, which is a superposition of a clean speech signal **(b)** and a noise **(c)**. The **black** regions in (b) indicate ground-truth silent intervals. **(d)** The noise exposed by automatically emergent silent intervals, i.e., the output of the silent interval detection component when the entire network is trained without silent interval supervision (recall Section 4.2). **(e)** The noise exposed by detected silent intervals, i.e., the output of the silent interval detection component when the network is trained with silent interval supervision (recall Section 4.3). **(f)** The estimated noise profile using subfigure (a) and (e) as the input to the noise estimation component. **(g)** The final denoised spectrogram output.

estimate the entire noise profile over time, which we do with a neural network.

Inputs to this component include both the noisy audio signal $\boldsymbol{x}$ and the incomplete noise profile $\tilde{\boldsymbol{x}}$. Both are converted by STFT into spectrograms, denoted as $\boldsymbol{s_x}$ and $\boldsymbol{s_{\tilde{x}}}$, respectively. We view the spectrograms as 2D images. And because the neighboring time-frequency pixels in a spectrogram are often correlated, our goal here is conceptually akin to the image inpainting task in computer vision [44]. To this end, we encode $\boldsymbol{s_x}$ and $\boldsymbol{s_{\tilde{x}}}$ by two separate 2D convolutional encoders into two feature maps. The feature maps are then concatenated in a channel-wise manner and further decoded by a convolutional decoder to estimate the full noise spectrogram, which we denote as $\mathrm{N}(\boldsymbol{s_x}, \boldsymbol{s_{\tilde{x}}})$. A

result of this step is illustrated in Figure 4.2-f.

4.1.3 Noise removal

Lastly, we clean up the noise from the input signal $\boldsymbol{x}$. We use a neural network R that takes as input both the input audio spectrogram $\boldsymbol{s_x}$ and the estimated full noise spectrogram $\mathsf{N}(\boldsymbol{s_x}, \boldsymbol{s_{\tilde{x}}})$. The two input spectrograms are processed individually by their own 2D convolutional encoders. The two encoded feature maps are then concatenated together before passing to a bidirectional LSTM followed by three fully connected layers (see details in Section 4.4). Like other audio enhancement models [101, 111, 112], the output of this component is a vector with two channels which form the real and imaginary parts of a complex ratio mask $\mathrm{c} \coloneqq \mathsf{R}\big(\boldsymbol{s_x}, \mathsf{N}(\boldsymbol{s_x}, \boldsymbol{s_{\tilde{x}}})\big)$ in frequency-time domain. In other words, the mask c has the same (temporal and frequency) dimensions as $\boldsymbol{s_x}$.

In the final step, we compute the denoised spectrogram $\boldsymbol{s_x^*}$ through element-wise multiplication of the input audio spectrogram $\boldsymbol{s_x}$ and the mask c (i.e., $\boldsymbol{s_x^*} = \boldsymbol{s_x} \odot \mathrm{c}$). Finally, the cleaned-up audio signal is obtained by applying the inverse STFT to $\boldsymbol{s_x^*}$ (see Figure 4.2-g).

4.2 Loss functions

Since a subgradient exists at every step, we are able to train our network in an end-to-end fashion with stochastic gradient descent. We optimize the following loss function:

$$\mathcal{L}_0 = \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\Big[\, \big\|\mathsf{N}(\boldsymbol{s_x}, \boldsymbol{s_{\tilde{x}}}) - \boldsymbol{s_n^*}\big\|_2 + \beta \big\|\boldsymbol{s_x} \odot \mathsf{R}\big(\boldsymbol{s_x}, \mathsf{N}(\boldsymbol{s_x}, \boldsymbol{s_{\tilde{x}}})\big) - \boldsymbol{s_x^*}\big\|_2 \,\Big], \tag{4.1}$$

where the notations $\boldsymbol{s_x}$, $\boldsymbol{s_{\tilde{x}}}$, $\mathsf{N}(\cdot,\cdot)$, and $\mathsf{R}(\cdot,\cdot)$ are defined in Section 4.1; $\boldsymbol{s_x^*}$ and $\boldsymbol{s_n^*}$ denote the spectrograms of the ground-truth foreground signal and background noise, respectively. The first term penalizes the discrepancy between estimated noise and the ground-truth noise, while the second term accounts for the estimation of foreground signal. These two terms are balanced by the scalar β ($\beta = 1.0$ in our experiments).

4.2.1 Natural emergence of silent intervals

While producing plausible denoising results (see Section 5.4), the end-to-end training process has no supervision on silent interval detection: the loss function (4.1) only accounts for the recoveries of noise and clean speech signal. But somewhat surprisingly, the ability of detecting silent intervals automatically emerges as the output of the first network component (see Figure 4.2-d as an example, which visualizes $\boldsymbol{s}_{\tilde{\boldsymbol{x}}}$). In other words, the network automatically learns to detect silent intervals for speech denoising without this supervision.

4.3 Silent interval supervision

As the model is learning to detect silent intervals on its own, we are able to directly supervise silent interval detection to further improve the denoising quality. Our first attempt was to add a term in (4.1) that penalizes the discrepancy between detected silent intervals and their ground truth. But our experiments show that this is not effective (see Section 5.4). Instead, we train our network in two sequential steps.

First, we train the silent interval detection component through the following loss function:

$$\mathcal{L}_1 = \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\Big[\ell_{\mathrm{BCE}}\big(\boldsymbol{m}(\boldsymbol{x}), \boldsymbol{m}^*_{\boldsymbol{x}}\big)\Big], \tag{4.2}$$

where $\ell_{\mathrm{BCE}}(\cdot,\cdot)$ is the binary cross entropy loss, $\boldsymbol{m}(\boldsymbol{x})$ is the mask resulted from silent interval detection component, and $\boldsymbol{m}^*_{\boldsymbol{x}}$ is the ground-truth label of each signal sample being silent or not—the way of constructing $\boldsymbol{m}^*_{\boldsymbol{x}}$ and the training dataset will be described in Section 5.1.

Next, we train the noise estimation and removal components through the loss function (4.1). This step starts by neglecting the silent detection component. In the loss function (4.1), instead of using $\boldsymbol{s}_{\tilde{\boldsymbol{x}}}$, the noise spectrogram exposed by the estimated silent intervals, we use the noise spectrogram exposed by the ground-truth silent intervals (i.e., the STFT of $\boldsymbol{x} \odot \boldsymbol{m}^*_{\boldsymbol{x}}$). After training using such a loss function, we fine-tune the network components by incorporating the already trained silent interval detection component. With the silent interval detection component fixed, this fine-tuning step optimizes the original loss function (4.1) and thereby updates the weights of the noise estimation

and removal components.

4.4 Network details and training

We now present the details of our network structure and training configurations.

4.4.1 Silent interval detection

The silent interval detection component of our model is composed of 2D convolutional layers, a bidirectional LSTM, and two FC layers. The parameters of the convolutional layers are shown in Table 4.1. Each convolutional layer is followed by a batch normalization layer with a ReLU activation function. The hidden size of bidirectional LSTM is 100. The two FC layers, interleaved with a ReLU activation function, have hidden size of 100 and 1, respectively.

Table 4.1: **Architecture of the silent interval detection component.**

	conv1	conv2	conv3	conv4	conv5	conv6	conv7	conv8	conv9	conv10	conv11	conv12
Num Filters	48	48	48	48	48	48	48	48	48	48	48	8
Filter Size	(1,7)	(7,1)	(5,5)	(5,5)	(5,5)	(5,5)	(5,5)	(5,5)	(5,5)	(5,5)	(5,5)	(1,1)
Dilation	(1,1)	(1,1)	(1,1)	(2,1)	(4,1)	(8,1)	(16,1)	(32,1)	(1,1)	(2,2)	(4,4)	(1,1)
Stride	1	1	1	1	1	1	1	1	1	1	1	1

4.4.2 Noise estimation

The noise estimation component of our model is fully convolutional, consisting of two encoders and one decoder. The two encoders process the noisy signal and the incomplete noise profile, respectively; they have the same architecture (shown in Table 4.2) but different weights. The two feature maps resulted from the two encoders are concatenated in a channel-wise manner before feeding into the decoder. In Table 4.2, every layer, except the last one, is followed by a batch normalization layer together with a ReLU activation function. In addition, there is skip connections between the 2nd and 14-th layer and between the 4-th and 12-th layer.

Table 4.2: **Architecture of the noise estimation component.** 'C' indicates a convolutional layer, and 'TC' indicates a transposed convolutional layer.

	Encoder			**Decoder**												
ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Layer Type	C	C	C	C	C	C	C	C	C	C	C	TC	C	TC	C	C
Num Filters	64	128	128	256	256	256	256	256	256	256	256	128	128	64	64	2
Filter Size	5	5	5	3	3	3	3	3	3	3	3	3	3	3	3	3
Dilation	1	1	1	1	1	2	4	8	16	1	1	1	1	1	1	1
Stride	1	2	1	2	1	1	1	1	1	1	1	2	1	2	1	1

4.4.3 Noise removal

The noise removal component of our model is composed of two 2D convolutional encoders, a bidirectional LSTM, and three FC layers. The two convolutional encoders take as input the input audio spectrogram $\boldsymbol{s_x}$ and the estimated full noise spectrogram $\mathsf{N}(\boldsymbol{s_x}, \boldsymbol{s_{\tilde{x}}})$, respectively. The first encoder has the network architecture listed in Table 4.3, and the second has the same architecture but with half of the number of filters at each convolutional layer. Moreover, the bidirectional LSTM has the hidden size of 200, and the three FC layers have the hidden size of 600, 600, and $2F$, respectively, where F is the number of frequency bins in the spectrogram. In terms of the activation function, ReLU is used after each layer except the last layer, which uses sigmoid.

Table 4.3: **Convolutional encoder for the noise removal component.** Each convolutional layer is followed by a batch normalization layer with ReLU as the activation function.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15
Num Filters	96	96	96	96	96	96	96	96	96	96	96	96	96	96	8
Filter Size	(1,7)	(7,1)	(5,5)	(5,5)	(5,5)	(5,5)	(5,5)	(5,5)	(5,5)	(5,5)	(5,5)	(5,5)	(5,5)	(5,5)	(1,1)
Dilation	(1,1)	(1,1)	(1,1)	(2,1)	(4,1)	(8,1)	(16,1)	(32,1)	(1,1)	(2,2)	(4,4)	(8,8)	(16,16)	(32,32)	(1,1)
Stride	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

4.4.4 Training details

We use PyTorch platform to implement our speech denoising model, which is then trained with the Adam optimizer. In our end-to-end training without silent interval supervision (referred to as "`Ours w/o SID loss`" in Chapter 5; also recall Section 4.2), we run the Adam optimizer for 50 epochs with a batch size of 20 and a learning rate of 0.001. When the silent interval supervision is incorporated (recall Section 4.3), we first train the silent interval detection component with the following setup: run the Adam optimizer for 100 epochs with a batch size of 15 and a learning rate of 0.001. Afterwards, we train the noise estimation and removal components using the same setup as the end-to-end training of "`Ours w/o SID loss`".

Chapter 5: Experiments

This chapter presents the major evaluations of our method, comparisons to several baselines and prior works, and ablation studies.

5.1 Experiment setup

5.1.1 Dataset construction

To construct training and testing data, we leverage publicly available audio datasets. We obtain clean speech signals using AVSPEECH [101], from which we randomly choose 2448 videos (4.5 hours of total length) and extract their speech audio channels. Among them, we use 2214 videos for training and 234 videos for testing, so the training and testing speeches are fully separate. All these speech videos are in English, selected on purpose: as we show in Section 5.7, our model trained on this dataset can readily denoise speeches in other languages.

We use two datasets, DEMAND [113] and Google's `AudioSet` [114], as background noise. Both consist of environmental noise, transportation noise, music, and many other types of noises. DEMAND has been used in previous denoising works (e.g., [39, 72, 32]). Yet `AudioSet` is much larger and more diverse than DEMAND, thus more challenging when used as noise. Figure 5.1 shows some noise examples. Our evaluations are conducted on both datasets, separately.

Due to the linearity of acoustic wave propagation, we can superimpose clean speech signals with noise to synthesize noisy input signals (similar to previous works [39, 72, 32]). When synthesizing a noisy input signal, we randomly choose a signal-to-noise ratio (SNR) from seven discrete values: -10dB, -7dB, -3dB, 0dB, 3dB, 7dB, and 10dB; and by mixing the foreground speech with properly scaled noise, we produce a noisy signal with the chosen SNR. For example, a -10dB SNR means that the power of noise is ten times the speech (see Figure 5.2). The SNR range in our evaluations (i.e., [-10dB, 10dB]) is significantly larger than those tested in previous works.

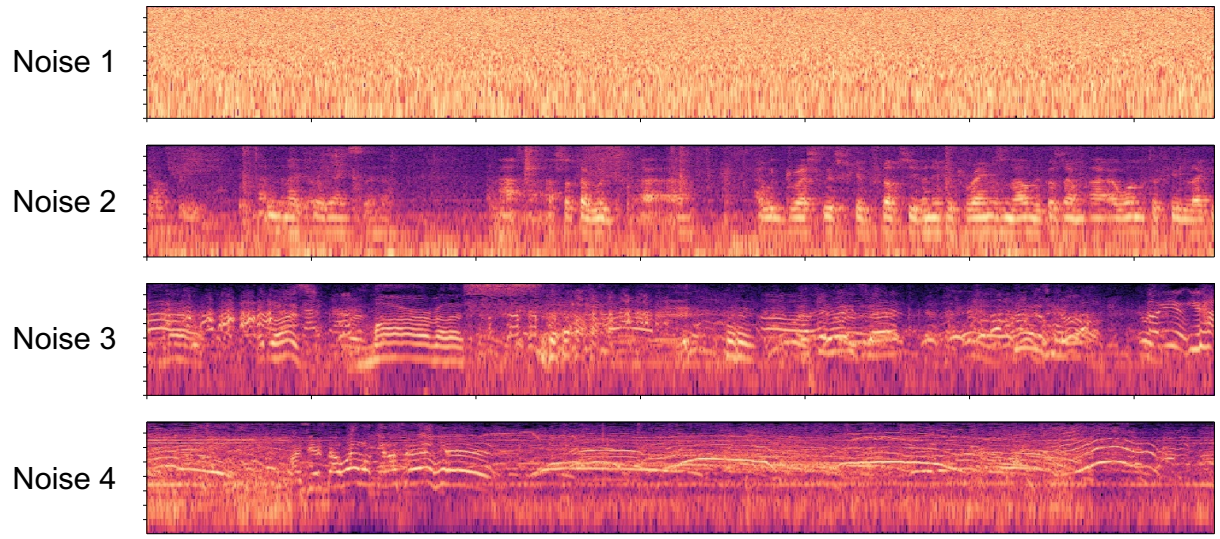


Figure 5.1: **Noise gallery.** We show four examples of noise from the noise datasets. Noise 1) is a stationary (white) noise, and the other three are not. Noise 2) is a monologue in a meeting. Noise 3) is party noise from people speaking and laughing with background noise. Noise 4) is street noise from people shouting and screaming with additional traffic noise such as vehicles driving and honking.

To supervise our silent interval detection (recall Section 4.3), we need ground-truth labels of silent intervals. To this end, we divide each clean speech signal into time segments, each of which lasts $1/30$ seconds. We label a time segment as silent when the total acoustic energy in that segment is below a threshold. Since the speech is clean, this automatic labeling process is robust.

Remarks on creating our own datasets. Unlike many previous models, which are trained using existing datasets such as Valentini's VoiceBank-DEMAND [32], we choose to create our own datasets because of two reasons. First, Valentini's dataset has a noise SNR level in [0dB, 15dB], much narrower than what we encounter in real-world recordings. Secondly, although Valentini's dataset provides several kinds of environmental noise, it lacks the richness of other types of structured noise such as music, making it less ideal for denoising real-world recordings (see discussion in Section 5.7).

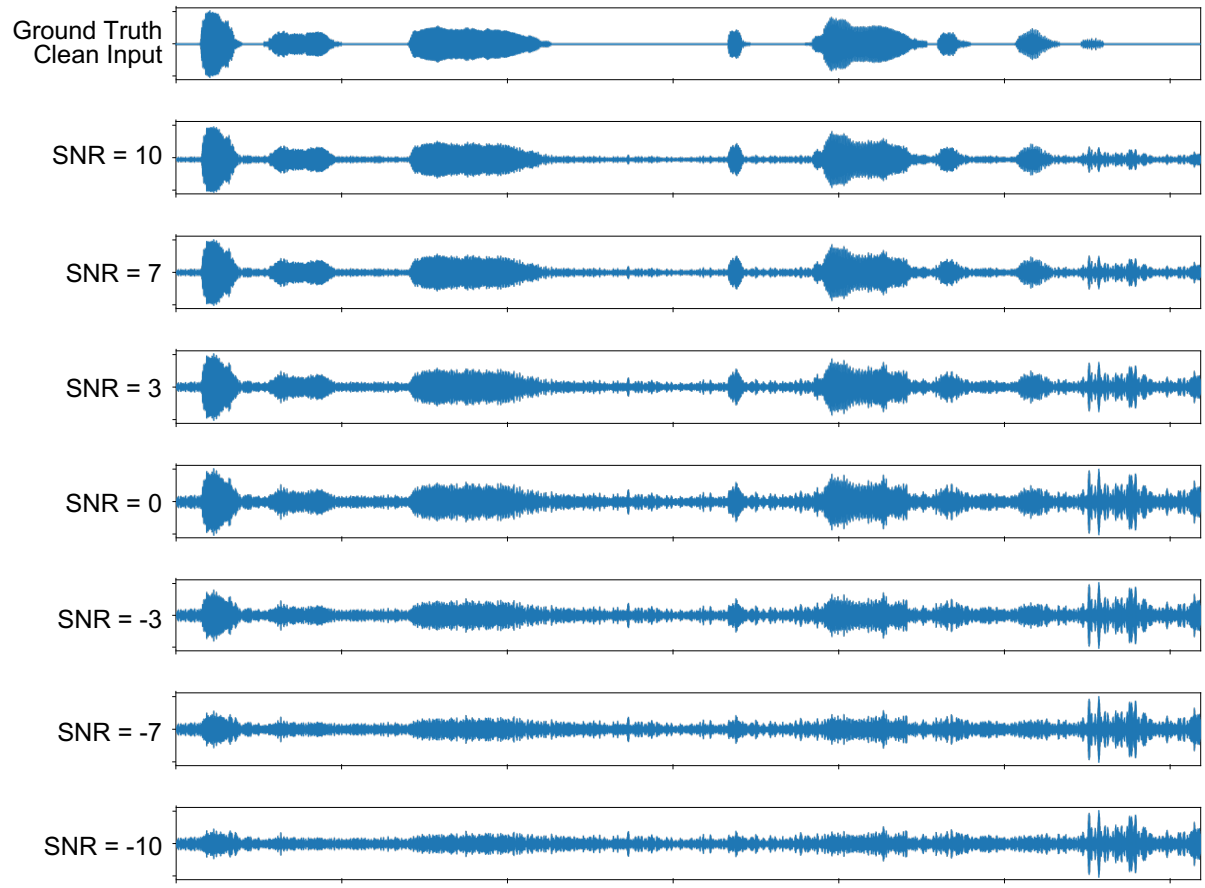


Figure 5.2: **Constructed noisy audio based on different SNR levels.** The first row shows the waveform of the ground truth clean input.

5.1.2 Data processing details

Our model is designed to take as input a mono-channel audio clip of an arbitrary length. However, when constructing the training dataset, we set each audio clip in the training dataset to have the same 2-second length, to enable batching at training time. To this end, we split each original audio clip from AVSPEECH, DEMAND, and `AudioSet` into 2-second long clips. All audio clips are then downsampled at 16kHz before converting into spectrograms using STFT. To perform STFT, the Fast Fourier Transform (FFT) size is set to 510, the Hann window size is set to 28ms, and the hop length is set to 11ms. As a result, each 2-second clip yields a (complex-valued) spectrogram with a resolution 256×178, where 256 is the number of frequency bins, and 178 is the temporal

resolution. At inference time, our model can still accept audio clips with arbitrary length.

Both our clean speech dataset and noise datasets are first split into training and test sets, so that no audio clips in training and testing are from the same original audio source—they are fully separate.

To supervise our silent interval detection, we label the clean audio signals in the following way. We first normalize each audio clip so that its magnitude is in the range [-1,1], that is, ensuring the largest waveform magnitude at -1 or 1. Then, the clean audio clip is divided into segments of length $1/30$ seconds. We label a time segment as a "silent" segment (i.e., label 0) if its average waveform energy in that segment is below 0.08. Otherwise, it is labeled as a "non-silent" segment (i.e., label 1).

5.1.3 Method comparison

We compare our method with several existing methods that are also designed for speech denoising, including both the classic approaches and recently proposed learning-based methods. We refer to these methods as follows: **i)** `Ours`, our model trained with silent interval supervision (recall Section 4.3); **ii)** `Baseline-thres`, a baseline method that uses acoustic energy threshold to label silent intervals (the same as our automatic labeling approach in Section 5.1 but applied on noisy input signals), and then uses our trained noise estimation and removal networks for speech denoising. **iii)** `Ours-GTSI`, another reference method that uses our trained noise estimation and removal networks, but hypothetically uses the ground-truth silent intervals; **iv)** `Spectral Gating`, the classic speech denoising algorithm based on spectral subtraction [30]; **v)** `Adobe Audition` [41], one of the most widely used professional audio processing software, and we use its machine-learning-based noise reduction feature, provided in the latest Adobe Audition CC 2020, with default parameters to batch process all our test data; **vi)** SEGAN [39], one of the state-of-the-art audio-only speech enhancement methods based on generative adversarial networks. **vii)** `DFL` [72], a recently proposed speech denoising method based on a loss function over deep network features; [1]

[1]This recent method is designed for high-noise-level input, trained in an end-to-end fashion, and as their paper states, is "particularly pronounced for the hardest data with the most intrusive background noise".

viii) VSE [35], a learning-based method that takes both video and audio as input, and leverages both audio signal and mouth motions (from video footage) for speech denoising. We could not compare with another audiovisual method [101] because no source code or executable is made publicly available.

For fair comparisons, we train all the methods (except `Spectral Gating` which is not learning-based and `Adobe Audition` which is commercially shipped as a black box) using the same datasets. For SEGAN, DFL, and VSE, we use their source codes published by the authors. The audiovisual denoising method VSE also requires video footage, which is available in AVSPEECH.

5.2 Evaluation on speech denoising

5.2.1 Metrics

Due to the perceptual nature of audio processing tasks, there is no widely accepted single metric for quantitative evaluation and comparisons. We therefore evaluate our method under six different metrics, all of which have been frequently used for evaluating audio processing quality. Namely, these metrics are: **i)** Perceptual Evaluation of Speech Quality (PESQ) [115], **ii)** Segmental Signal-to-Noise Ratio (SSNR) [116], **iii)** Short-Time Objective Intelligibility (STOI) [117], **iv)** Mean opinion score (MOS) predictor of signal distortion (CSIG) [118], **v)** MOS predictor of background-noise intrusiveness (CBAK) [118], and **vi)** MOS predictor of overall signal quality (COVL) [118].

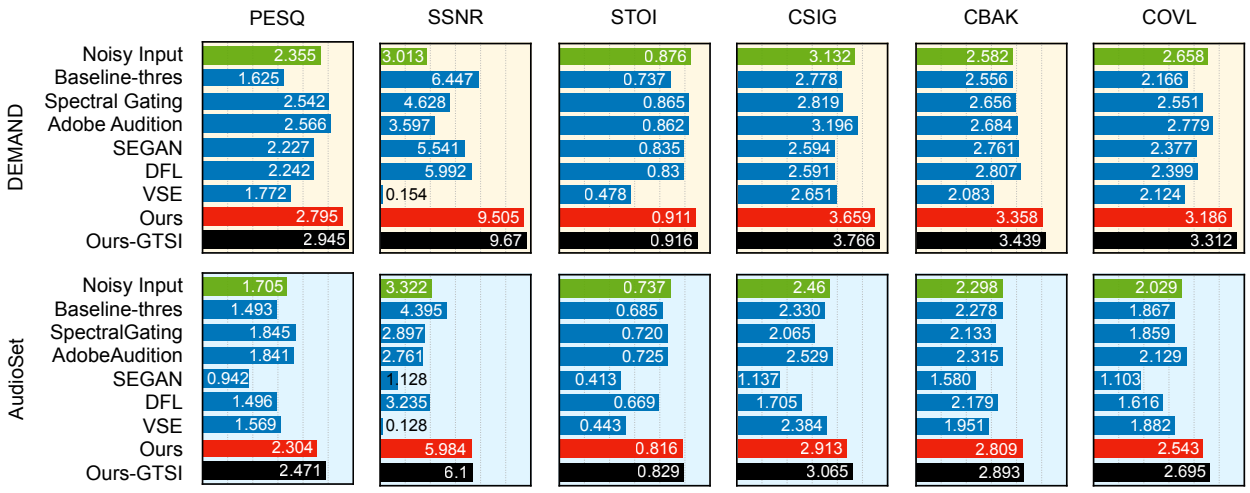


Figure 5.3: **Quantitative comparisons.** We measure denoising quality under six metrics (corresponding to columns). The comparisons are conducted using noise from DEMAND and `AudioSet` separately. `Ours-GTSI` (in black) uses ground-truth silent intervals. Although not a practical approach, it serves as an upper-bound reference of all methods. Meanwhile, the green bar in each plot indicates the metric score of the noisy input without any processing.

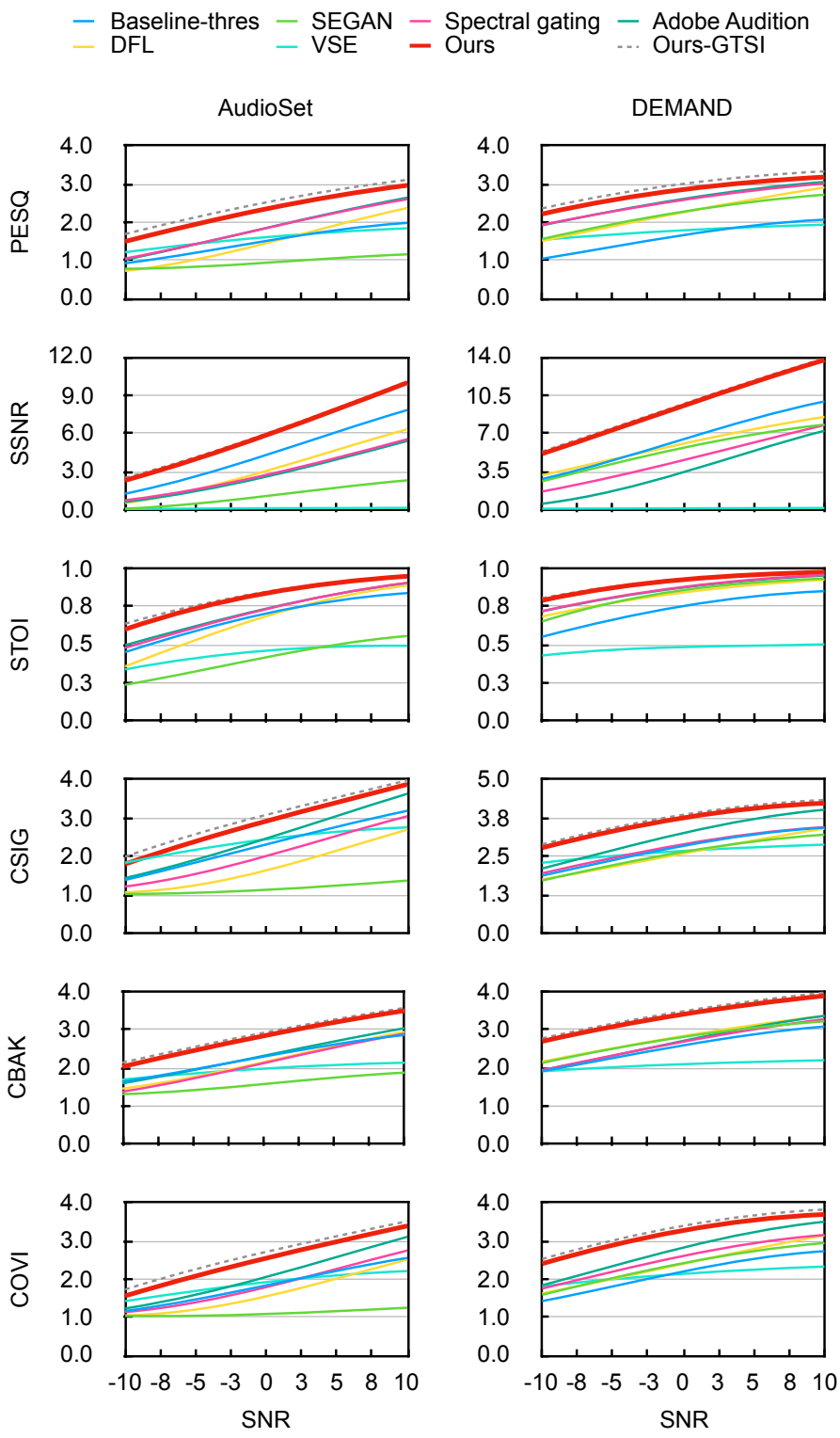


Figure 5.4: **Denoise quality w.r.t input SNRs.** Denoise results for each method w.r.t different input SNRs under all six metrics described in Section 5.2.

5.2.2 Results

We train two separate models using DEMAND and `AudioSet` noise datasets respectively, and compare them with other models trained with the same datasets. We evaluate the average metric values and report them in Figure 5.3. Under all metrics, our method consistently outperforms others.

We breakdown the performance of each method with respect to SNR levels from -10dB to 10dB on both noise datasets. The results are reported in Figure 5.4. In the previous works that we compare to, no results under those low SNR levels (at < 0 dBs) are reported. Nevertheless, across all input SNR levels, our method performs the best, showing that our approach is fairly robust to both light and extreme noise.

From Figure 5.4, it is worth noting that `Ours-GTSI` method performs even better. Recall that this is our model but provided with ground-truth silent intervals. While not practical (due to the need of ground-truth silent intervals), `Ours-GTSI` confirms the importance of silent intervals for denoising: a high-quality silent interval detection helps to improve speech denoising quality.

5.3 Evaluation on silent interval detection

Due to the importance of silent intervals for speech denoising, we also evaluate the quality of our silent interval detection, in comparison to two alternatives, the baseline `Baseline-thres` and a Voice Activity Detector (VAD) [119]. The former is described above, while the latter classifies each time window of an audio signal as having human voice or not [120, 121]. We use an off-the-shelf VAD [122], which is developed by Google's WebRTC project and reported as one of the best available. Typically, VAD is designed to work with low-noise signals. Its inclusion here here is merely to provide an alternative approach that can detect silent intervals in more ideal situations.

We evaluate these methods using four standard statistic metrics: the precision, recall, F1 score, and accuracy. We follow the standard definitions of these metrics, which are summarized in Appendix A. These metrics are based on the definition of positive/negative conditions. Here, the positive condition indicates a time segment being labeled as a silent segment, and the negative condition indicates a non-silent label. Thus, the higher the metric values are, the better the detection

approach.

Table 5.1: **Results of silent interval detection.** The metrics are measured using our test signals that have SNRs from -10dB to 10dB. Definitions of these metrics are summarized in Appendix A.

Noise Dataset	Method	Precision	Recall	F1	Accuracy
DEMAND	`Baseline-thres`	0.533	0.718	0.612	0.706
	VAD	0.797	0.432	0.558	0.783
	`Ours`	**0.876**	**0.866**	**0.869**	**0.918**
Audioset	`Baseline-thres`	0.536	0.731	0.618	0.708
	VAD	0.736	0.227	0.338	0.728
	`Ours`	**0.794**	**0.822**	**0.807**	**0.873**

Table 5.1 shows that, under all metrics, our method is consistently better than the alternatives. Between VAD and `Baseline-thres`, VAD has higher precision and lower recall, meaning that VAD is overly conservative and `Baseline-thres` is overly aggressive when detecting silent intervals (see Figure 5.5 below). Our method reaches better balance and thus detects silent intervals more accurately.

In Figure 5.5, we present an example of silent interval detection results in comparison to two alternative methods, referred to as `Baseline-thres` and VAD, respectively. Figure 5.5 echos the quantitative results in Table 5.1: VAD tends to be overly conservative, even in the presence of mild noise; and many silent intervals are ignored. On the other hand, `Baseline-thres` tends to be too aggressive; it produces many false intervals. In contrast, our silent interval detection maintains a better balance, and thus predicts more accurately.

5.4 Ablation studies

In addition, we perform a series of ablation studies to understand the efficacy of individual network components and loss terms. In Table 5.2, "`Ours`" refers to our proposed network structure and training method that incorporates silent interval supervision (recall Section 4.3). Details are described in Section 4.4. "`Ours w/o SID loss`" refers to our proposed network structure

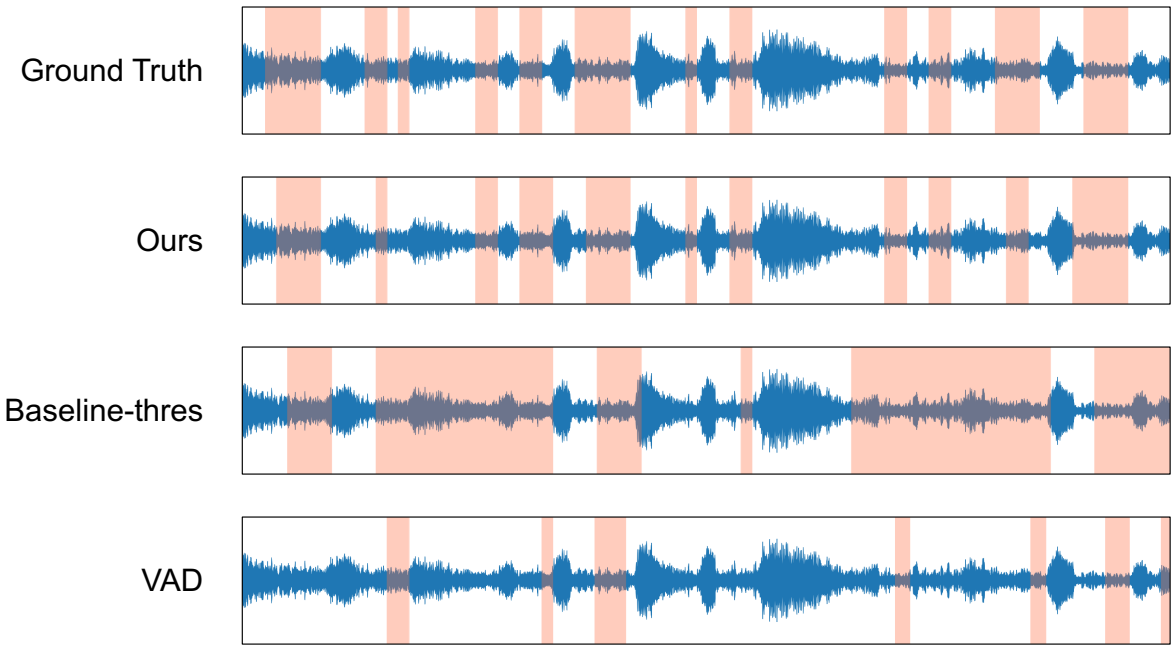


Figure 5.5: **An example of silent interval detection results.** Provided an input signal whose SNR is 0dB (top row), we show the silent intervals (in red) detected by three approaches: our method, `Baseline-thres`, and VAD. We also show ground-truth silent intervals in the top row.

but optimized by the training method in Section 4.2 (i.e. an end-to-end training without silent interval supervision). This ablation study is to confirm that silent interval supervision indeed helps to improve the denoising quality. "`Ours Joint loss`" refers to our proposed network structure optimized by the end-to-end training approach that optimizes the loss function (4.1) with the additional term (4.2). In this end-to-end training, silent interval detection is also supervised through the loss function. This ablation study is to confirm that our two-step training (Section 4.3) is more effective. term (4.2). "`Ours w/o NE loss`" uses our two-step training (in Section 4.3) but without the loss term on noise estimation—that is, without the first term in (4.1). This ablation study is to examine the necessity of the loss term on noise estimation for better denoising quality. In comparison to these alternative training approaches, our two-step training with silent interval supervision (referred to as "`Ours`") performs the best. We also note that "`Ours w/o SID loss`"—i.e., without supervision on silent interval detection—already outperforms the methods

Table 5.2: **Ablation studies.** We alter network components and training loss, and evaluate the denoising quality under various metrics. Our proposed approach performs the best.

Noise Dataset	Method	PESQ	SSNR	STOI	CSIG	CBAK	COVL
DEMAND	`Ours w/o SID comp`	2.689	9.080	0.904	3.615	3.285	3.112
	`Ours w/o NR comp`	2.476	0.234	0.747	3.015	2.410	2.637
	`Ours w/o SID loss`	2.794	6.478	0.903	3.466	3.147	3.079
	`Ours w/o NE loss`	2.601	9.070	0.896	3.531	3.237	3.027
	`Ours Joint loss`	2.774	6.042	0.895	3.453	3.121	3.068
	`Ours`	**2.795**	**9.505**	**0.911**	**3.659**	**3.358**	**3.186**
Audioset	`Ours w/o SID comp`	2.190	5.574	0.802	2.851	2.719	2.454
	`Ours w/o NR comp`	1.803	0.191	0.623	2.301	2.070	1.977
	`Ours w/o SID loss`	**2.325**	4.957	0.814	2.814	2.746	2.503
	`Ours w/o NE loss`	2.061	5.690	0.789	2.766	2.671	2.362
	`Ours Joint loss`	2.305	4.612	0.807	2.774	2.721	2.474
	`Ours`	2.304	**5.984**	**0.816**	**2.913**	**2.809**	**2.543**

we compared to in Figure 5.3, and "`Ours`" further improves the denoising quality. This shows the efficacy of our proposed training approach.

We also experimented with two variants of our network structure. The first one, referred to as "`Ours w/o SID comp`" turns off silent interval detection: the silent interval detection component always outputs a vector with all zeros. As a result, the input noise profile to the noise estimation component N is made precisely the same as the original noisy signal. This ablation study is to examine the effect of silent intervals for speech denoising. The second, referred as "`Ours w/o NR comp`" uses a simple spectral subtraction to replace our noise removal component; the other components remain unchanged. This ablation studey is to examine the efficacy of our noise removal component. Table 5.2 shows that, under all the tested metrics, both variants perform worse than our method, suggesting our proposed network structure is effective.

5.4.1 The influence of silent interval detection on denoising quality

A key insight of our neural-network-based denoising model is the leverage of silent interval distribution over time. The experiments above have confirmed the efficacy of our silent interval

detection for better speech denoising. We now report additional experiments, aiming to gain some empirical understanding of how the quality of silent interval prediction would affect speech denoising quality.

Table 5.3: **Silent interval detection vs. denoising quality.** Results on how silent interval detection quality affects the speech denoising quality.

Shift	No Shift	1/30	1/10	1/5	1/2
PESQ	**2.471**	1.932	1.317	1.169	1.094

(a) Effect of shifting silent intervals.

Shrink	No Shrink	20%	40%	60%	80%
PESQ	**2.471**	2.361	2.333	2.283	2.249

(b) Effect of shrinking silent intervals.

First, starting with ground-truth silent intervals, we shift them on the time axis by $1/30$, $1/10$, $1/5$, and $1/2$ seconds. As the shifted time amount increases, more time segments become incorrectly labeled: both the numbers of false positive labels (i.e., non-silent time segments labeled silent) and false negative labels (i.e., silent time segments are labeled non-silent) increase. After each shift, we feed the silent interval labels to our noise estimation and removal components and measure the denoising quality under the PESQ score.

In the second experiment, we again start with ground-truth silent intervals; but instead of shifting them, we shrink each silent interval toward its center by 20%, 40%, 60%, and 80%. As the silent intervals become more shrunken, fewer time segments are labeled as silent. In other words, only the number of false negative predictions increases. Similar to the previous experiment, after each shrink, we use the silent interval labels in our speech denoising pipeline, and meausure the PESQ score.

The results of both experiments are reported in Table 5.3. As we shrink the silent intervals, the denoising quality drops gently. In contrast, even a small amount of shift causes a clear drop of denoising quality. These results suggest that in comparison to false negative predictions, false positive predictions affect the denoising quality more negatively. On the one hand, reasonably conservative predictions may leave certain silent time segments undetected (i.e., introducing some false negative labels), but the detected silent intervals indeed reveal the noise profile. On the other

Table 5.4: **Comparisons on VoiceBank-DEMAND corpus.**

Method	PESQ	CSIG	CBAK	COVL	STOI
Noisy Input	1.97	3.35	2.44	2.63	0.91
WaveNet [10]	–	3.62	3.24	2.98	–
SEGAN [39]	2.16	3.48	2.94	2.80	0.93
DFL [72]	2.51	3.79	3.27	3.14	–
MMSE-GAN [123]	2.53	3.80	3.12	3.14	0.93
MetricGAN [124]	2.86	3.99	3.18	3.42	–
SDR-PESQ [125]	3.01	4.09	3.54	3.55	–
T-GSA [126]	3.06	4.18	3.59	3.62	–
Self-adapt. DNN [127]	2.99	4.15	3.42	3.57	–
RDL-Net [128]	3.02	4.38	3.43	3.72	0.94
Ours	3.16	3.96	3.54	3.53	0.98

hand, even a small amount of false positive predictions causes certain non-silent time segments to be treated as silent segments, and thus the observed noise profile through the detected silent intervals would be tainted by foreground signals.

5.5 Comparison with state-of-the-art benchmark

Many state-of-the-art denoising methods, including MMSE-GAN [123], Metric-GAN [124], SDR-PESQ [125], T-GSA [126], Self-adaptation DNN [127], and RDL-Net [128], are all evaluated on Valentini's VoiceBank-DEMAND [32]. We therefore compare ours with those methods on the same dataset. We note that DEMAND consists of audios with SNR in [0dB, 15dB]. Its SNR range is much narrower than what our method (and our training datasets) aims for (e.g., input signals with -10dB SNR). Nevertheless, trained and tested under the same setting, our method is highly competitive to the best of those methods under every metric, as shown in Table 5.4. The metric scores therein for other methods are numbers reported in their original papers.

5.6 Generalization across datasets

To evaluate the generalization ability of our model, we performed *three* cross-dataset tests reported in Table 5.5. The experiments are set up in the following way.

- "Test **i**": We train our model on our own AVSPEECH + Audioset (AA) dataset but evaluate on Valentini's VoiceBank-DEMAND (VD) testset. The result is shown in the first row of the "Test **i**" section in Table 5.5. In comparison, the second row shows the result of training on VD and testing on VD.

- "Test **ii**": We train our model on our own AA dataset but evaluate on our second AVSPEECH + DEMAND (AD) testset. The result is shown in the first row of the "Test **ii**" section in Table 5.5. In comparison, the second row shows the result of training on AD and testing on AD.

- "Test **iii**": We train our model on our own AD dataset but evaluate on AA testset. The result is shown in the first row of the "Test **iii**" section of the table. In comparison, the second row shows the result of training on AA and testing on AA.

The small degradation in each cross-dataset test demonstrates the great generalization ability of our method. We could not directly compare the generalization ability of our model with existing methods, as no previous work reported cross-dataset evaluation results.

5.7 Tests on real-world data

We also test our method against real-world data. Quantitative evaluation on real-world data, however, is not easy because the evaluation of nearly all metrics requires the corresponding ground-truth clean signal, which is not available in real-world scenario. Instead, we collected a good number of real-world audios, either by recording in daily environments or by downloading online (e.g., from YouTube). These real-world audios cover diverse scenarios: in a driving car, a café, a park, on the street, in multiple languages (Chinese, Japanese, Korean, German, French, etc.), with different genders and accents, and even with singing songs. None of these recordings is cherry picked.

Table 5.5: **Generalization across datasets.**

Test	Trainset	Testset	PESQ	CSIG	CBAK	COVL	STOI
Test **i**	AA	VD	3.00	3.78	3.08	3.34	0.98
	VD	VD	3.16	3.96	3.54	3.53	0.98
Test **ii**	AA	AD	2.65	3.48	3.21	3.01	0.90
	AD	AD	2.80	3.66	3.36	3.17	0.91
Test **iii**	AD	AA	2.12	2.71	2.65	2.34	0.79
	AA	AA	2.30	2.91	2.81	2.54	0.82

Furthermore, we use real-world data to test our model trained with different datasets, including our own dataset (recall Section 5.1) and the existing DEMAND [32]. We show that the network model trained by our own dataset leads to much better noise reduction (see details in Section 5.7.1 below). This suggests that our dataset allows the denoising model to better generalize to many real-world scenarios.

5.7.1 Generalization on real-world data

We conduct experiments to understand the extent to which the model trained with different datasets can generalize to real-world data. We train two versions of our model using our AVSPEECH + Audioset dataset and Valentini's VoiceBank-DEMAND, respectively (denoted as "Model AA" and "Model VD", respectively, in Table 5.6), and use them to denoise our collected real-world recordings. For the denoised real-world audios, we measure the noise level reduction in detected silent intervals. This measurement is doable, since it requires no knowledge of noise-free ground-truth audios. As shown in Table 5.6, in terms of noise reduction, the model trained with our own AA dataset outperforms the one trained with the public VoiceBank-DEMAND dataset by a significant amount for *all* tested real-world recordings. On average, it produces **22.3 dB** noise reduction in comparison to **12.6 dB**, suggesting that our dataset allows the denoising model to better generalize to many real-world scenarios.

Table 5.6: **Real-world recording noise reduction comparison.**

Real-world Recording	Model AA noise reduction (dB)	Model VD noise reduction (dB)
Song Excerpt 1	**22.38**	9.22
Song Excerpt 2	**21.16**	17.65
Chinese	**18.99**	15.65
Japanese	**16.95**	9.39
Korean	**25.90**	9.76
German	**14.54**	7.29
French	**21.95**	17.16
Spanish 1	**33.34**	11.00
Spanish 2	**31.66**	14.09
Female 1	**17.64**	7.79
Female 2	**30.21**	8.64
Female 3	**19.15**	6.70
Male 1	**24.81**	14.44
Male 2	**24.92**	13.35
Male 3	**13.10**	10.99
Multi-person 1	**32.00**	21.60
Multi-person 2	**15.10**	15.07
Multi-person 3	**18.71**	10.68
Street Interview	**20.54**	18.94
AVERAGE	**22.27**	12.60

Chapter 6: Summary

Speech denoising has been a long-standing challenge. We present a new network structure that leverages the abundance of silent intervals in speech. Even without silent interval supervision, our network is able to denoise speech signals plausibly, and meanwhile, the ability to detect silent intervals automatically emerges. We reinforce this ability. Our explicit supervision on silent intervals enables the network to detect them more accurately, thereby further improving the performance of speech denoising. As a result, under a variety of denoising metrics, our method consistently outperforms several state-of-the-art audio denoising models. In the future, the idea of harnessing silent intervals in a neural network model can be extended to denoise other types of audio signals, beyond speech.

6.1 Broader impact

High-quality speech denoising is desired in a myriad of applications: human-robot interaction, cellular communications, hearing aids, teleconferencing, music recording, filmmaking, news reporting, and surveillance systems to name a few. Therefore, we expect our proposed denoising method—be it a system used in practice or a foundation for future technology—to find impact in these applications.

In our experiments, we train our model using English speech only, to demonstrate its generalization property—the ability of denoising spoken languages beyond English. Our demonstration of denoising Japanese, Chinese, and Korean speeches is intentional: they are linguistically and phonologically distant from English (in contrast to other English "siblings" such as German and Dutch). Still, our model may bias in favour of spoken languages and cultures that are closer to English or that have frequent pauses to reveal silent intervals. Deeper understanding of this potential bias requires future studies in tandem with linguistic and sociocultural insights.

Lastly, it is natural to extend our model for denoising audio signals in general or even signals beyond audio (such as Gravitational wave denoising [129]). If successful, our model can bring in even broader impacts. Pursuing this extension, however, requires a judicious definition of "silent intervals". After all, the notion of "noise" in a general context of signal processing depends on specific applications: noise in one application may be another's signals. To train a neural network that exploits a general notion of silent intervals, prudence must be taken to avoid biasing toward certain types of noise.

Chapter 7: Implementation of Speech Denoising in Real-time

7.1 Overview

In Part I, our focus has been on achieving the highest possible quality in speech denoising, which necessitates knowing the entire track of the target noisy speech, thereby making our method offline. As discussed in Chapter 3, the majority of speech denoising techniques also operate within the offline domain, aiming to eliminate noise as effectively as possible. However, in practical applications, the absence of real-time capabilities can significantly limit the usefulness of these methods. In real-world scenarios, such as online communications, it is impractical to ask users to wait for a conversation to be processed before hearing the denoised speech. Regardless of the denoising quality, this delay results in an unacceptable user experience.

Real-time speech denoising is thus a highly demanded audio processing task, especially in our current context where online meetings have become a "new normal" due to the ongoing COVID pandemic. Recent advancements in state-of-the-art real-time denoising techniques are predominantly based on neural networks[130, 131, 132, 133, 134, 135, 136, 137]. These methods explore novel network architectures to balance achieving high-quality denoising with maintaining network simplicity, thereby reducing processing time and enabling real-time performance.

Unlike offline speech denoising, an audio signal in real-time setting is provided in a streaming fashion. The network must process signal samples as soon as they arrive, and generate output samples in the shortest possible delay. As a result, in almost all real-time techniques, a common strategy is to use a sliding window buffer of fixed length.

This seems a natural choice: the network waits until the sliding window buffer is fulfilled with incoming audio samples, and then denoises the data in that buffer. Afterwards, the resulting signal data are fed into a real-time audio player. In this way, the network expects an input signal with a fixed length L. As long as the network processing time is always less than L, the total lag from the

arrival of a signal sample to the playing time of the corresponding denoised sample is bounded, larger than L but less than $2L$ (see analysis in Section 7.2.1).

In practice, however, it is difficult, if not impossible, to choose a buffer length L that ensures real-time performance. A large L causes a long lag; a small L may lead a network processing time longer than L, resulting in choppy audio playback. This is because in a real computing environment, there are always other background processes (e.g., 4K video playback and games). The shorter the L is, the more prone is the denoising network to the CPU occupancy of other processes. In short, the sliding window strategy, albeit fundamental to real-time denoising, remains elusive from careful examination.

We propose a different sliding window strategy, namely the dynamic sliding window. In our approach, the input buffer length is not fixed. The network takes in currently buffered data regardless of its length, and starts to process it with no wait. While the network is running, the newly received data are accumulated in a buffer, ready to be processed when the network finishes its current round of denoising. This sliding window strategy, although conceptually simple, is more robust against CPU occupancy of other processes, and thereby resulting in shorter and more stable lag. To show this advantage, we formally analyze the audio playback delays caused by our approach and the commonly used fixed-size sliding window. To our knowledge, this is the first time the network delays under different sliding window strategies are examined.

Many existing real-time denoising networks (i.e., [130, 131, 132]) are not able to incorporate dynamic sliding window easily (see discussion in Section 7.2.2). We therefore propose a lightweight denoising network tailored for real-time setting: accepting streaming signals and processing them using the dynamic sliding window. By carefully padding and reusing data across sliding windows, our network undergoes *no* degradation on denoising quality comparing to the offline non-streaming case.

We conduct extensive experiments comparing the proposed model with several state-of-the-art real-time denoising methods. Results suggest that our proposed model produces the smallest playback lag amongst all compared methods while obtaining on-par denoising quality on all the

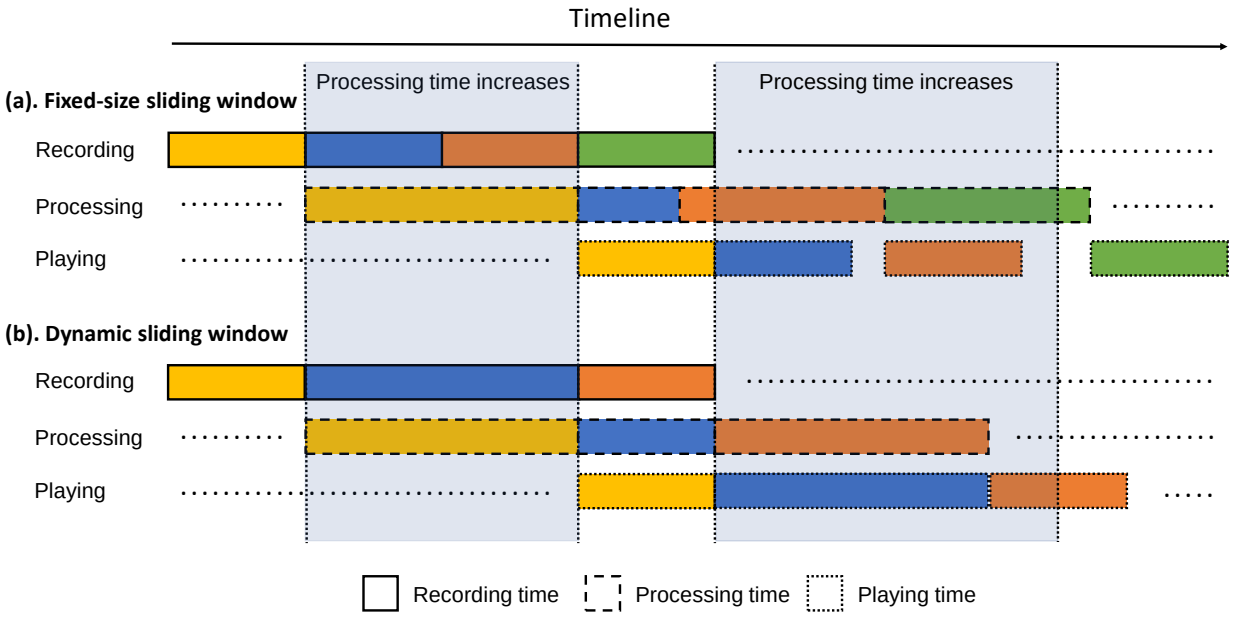


Figure 7.1: **Dynamic vs. fixed-size sliding windows.** Different windows are indicated by different colors. Each window is processed by the network individually. The gray areas indicate the time period in which CPU cycles are occupied by other processes and thus the network processing time increases.

quality metrics. Most importantly, compared with the prior real-time denoising method [130], our model is more robust to maintaining real-time performance in real-world scenarios, where other background tasks such as Zoom conferences, 4K video editing, and video games may preempt CPU cycles.

7.2 Method

We start by analyzing the audio playback lag when the fixed-size sliding window is used. This is compared to the lag using our proposed dynamic sliding window (Section 7.2.1). Motivated by our analysis, we then propose a lightweight denoising network that leverages dynamic sliding window for faster and more robust denoising (Section 7.2.2).

7.2.1 Sliding window for streaming data

Fixed-size sliding window. In almost all existing network-based denoising models, the incoming audio signal is treated as a time series of non-overlapping windows $[X_1, X_2, \cdots]$. Each window hosts a constant length L of audio samples (i.e., $X_i \in \mathbb{R}^L$), filled in a streaming fashion. The network F takes in the latest unprocessed window X_i, outputs denoised result $F(X_i)$, and then waits until the next window, X_{i+1}, is fulfilled (see Figure 7.1-a). Let t_k denote the network processing time for window X_k. Note that although the window size is fixed, in practice t_k varies over time due to other background processes' CPU occupancy. Our analysis shows that the delay (or lag) d_i from the moment of receiving X_i to the moment of network outputting $F(X_i)$ is expressed as

$$d_i = 2L + \max_{1 \le p \le q \le i} \sum_{k=p}^{q} (t_k - L). \tag{7.1}$$

The detailed derivation of (7.1) is included in Appendix B.

Our analysis (7.1) is revealing: in the ideal case wherein $t_i < L$ is always satisfied, the denoising network runs smoothly in real-time with no accumulating lag; the playback delay is upper-bounded by $2L$. However, in reality, the network execution is often affected by other background computing processes, and its processing time t_i may become larger than L. At the same time, a length t_i of audio samples arrives, accumulated in the buffer. To process this amount of data, the network needs to run $\lceil \frac{t_i}{L} \rceil > 1$ times. This can in turn cause audio playback lag to accumulate (and hence the summation term in (7.1)).

Dynamic sliding window. To overcome the above shortcomings of using a fixed-size sliding window, We argue that the size of the window should be dynamically changing. We propose to dynamically adjust the sliding window size. Immediately after the network finishes processing a data window X_i, the newly received data in the buffer have a length t_i, which may or may not be larger than L. Regardless of the buffer length, we denoise the available buffered data with no wait. Figure 7.1-b illustrates the process. Under this strategy, the delay d_i for playing back the window X_i

is:

$$d_i = \max_{k \le i} (t_{k-1} + t_k), \quad i \ge 2. \tag{7.2}$$

When $i = 1$, the delay for the first window is $d_1 = L_0 + t_1$, where L_0 is an initial window size to start the denoising process at the beginning. The detailed derivation of (7.2) is included in Appendix B.

This analysis shows that d_i depends only on the network processing time of two consecutive windows X_{k-1} and X_k. In contrast to the fixed-size sliding window (shown in (7.1)), There is no accumulating delay. Thus, our approach is more robust against computing power fluctuations. This is a remarkable advantage, since In a real computing environment, the computing power for denoising constantly varies (see Section 7.3.3).

7.2.2 Network structure and data padding

While the dynamic sliding window strategy is independent of specific network structures, many existing real-time denoising networks [130, 131, 132, 133, 134] can not be easily adapted to utilize it. Some of them require a predetermined sliding window size prior to the network execution [130]. Others focus on reducing the network inference cost, but how they handle the incoming data stream remains unclear [131, 132, 133, 134].

Proposed network structure. We propose a lightweight denoising network using the dynamic sliding window. Our network is built upon the noise removal component in [23] (and thus much simpler than the denoising model therein). Input to our network is a spectrogram $\mathrm{s_x}$ obtained by applying STFT on a data window X_i. The spectrogram $\mathrm{s_x}$ is first processed by a 2D convolutional layer with kernel size $(5, 5)$ and dilation $(1, 1)$ in time-frequency domain. The resulting feature map is fed into a unidirectional LSTM [64] of hidden size 400. Finally, three fully-connected layers of hidden size $(400, 600, 512)$ are applied for each time bin. Similar to other speech enhancement models [23, 101, 112], our network outputs a complex-valued mask c of the same dimension as

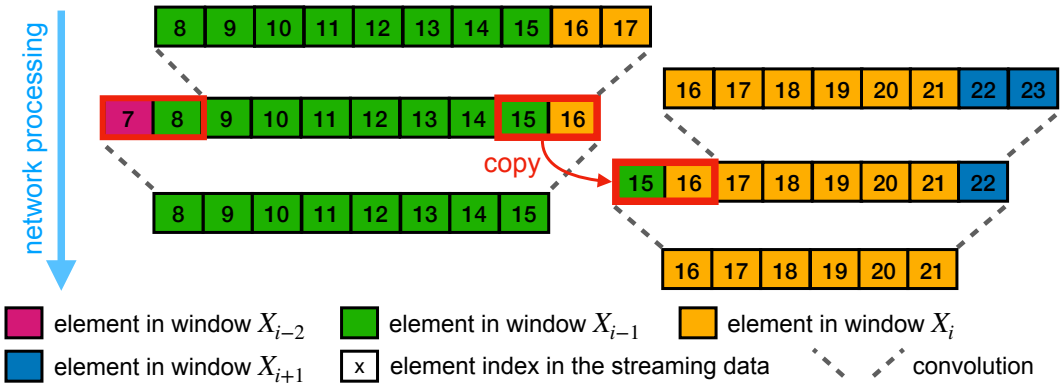


Figure 7.2: **Illustration of data padding.** Two consecutive windows of size 8 (green) and 6 (yellow) are processed by two 1D convolutions of kernel size 3. In this case, we pad two future elements for each window (22, 23 for window X_i) and reuse the two elements of the first convolution results from previous window (15, 16 obtained from processing window X_{i-1}).

s_x. Lastly, the denoised audio signal is obtained by applying the inverse STFT to $c \odot s_x$, where $\odot$ denote the Hadamard product.

At training time, we optimize the following loss function:

$$\mathcal{L} = \mathbb{E}_{x \sim p(x)} ||c \odot s_x - s_x^*||_2, \tag{7.3}$$

where s_x^* denotes the ground truth spectrogram of a clean audio. When computing the STFT, we set the number of FFT bins to 510, Hann window size to 400, and hop length to 128.

Data padding. Our network (and also many others) has convolutional layers, which require padding to process data on the boundary. To handle streaming data in a sliding window fashion, this means we need to buffer enough "future" data when a data window is processed (e.g., block 16 and 17 for window X_{i-1} in Figure 7.2). Waiting for the arrival of padding data introduces an additional lag (48ms in our practice). But we can reuse the convolution results of the padding data in the next sliding window (see illustration in Figure 7.2). Apart from saving some computational cost, reuse of padding data is critical to the network's denoising quality. It ensures that our network maintains the same denoising quality as if it takes in the entire signal all at once. Perhaps surprisingly,

such a guarantee remains lacking in existing real-time denoising networks (see experiments in Section 7.3.2).

7.3 Experiments

Our experiments are twofold: we evaluate our network's denoising quality by comparing it to the state-of-the-art real-time denoising models (Section 7.3.2). We then demonstrate the performance advantages of the dynamic sliding window over the conventional fixed-size sliding window approach (Section 7.3.3).

7.3.1 Experiment setup

Datasets. We conduct experiments on two publicly available datasets. The first, provided by Xu et al. [23], has clean audios selected from AVSPEECH [101] and noises from `AudioSet` [114] and DEMAND [113]. We refer to this dataset as AAD dataset. In addition, we also test on `Valentini` [32] benchmark, which contains audio clips from 28 speakers; each clip has its corresponding clean and noisy versions.

Evaluation metrics. To evaluate the denoising quality, we use the following widely used objective metrics: (i) STOI: Short-Time Objective Intelligibility [117]; (ii) PESQ: Perceptual evaluation of speech quality (we use narrow-band version) [115]; (iii) CSIG: MOS predictor of signal distortion [118]; (iv) CBAK: MOS prediction of the intrusiveness of background noise [118]; (v) COVL: MOS predictor of overall quality [118]; (vi) SSNR: Segmental Signal-to-Noise Ratio [116].

We use two metrics to evaluate the networks' real-time performance: the average network processing time (D_{N}) and the maximum audio playback lag (D_{A}). D_{N} is defined as $\frac{1}{M}\sum_{i=1}^{M} t_i$, where M is the total number of sliding windows, and t_i is the processing time for window X_i. D_{A} measures, as the audio samples come in, the maximum lag between the arrival time of an audio sample and its playback time (after denoising). Unless otherwise stated, the denoising networks are run on CPU (3.60GHz Intel 8-Core i7-9700K), and the metrics are measured in milliseconds.

Table 7.1: **Quantitative comparisons.** Denoising quality on AAD dataset in both offline (top) and real-time (bottom) settings. All scores are the average results for input audios with SNR [-10, -7, -3, 0, 3, 7, 10]. Second best is highlighted in cyan.

Mode	Methods	STOI	CSIG	CBAK	COVL	PESQ	SSNR
Non-real-time	noisy	0.73	2.46	2.29	2.03	1.70	-1.99
	`Demucs48`	0.78	2.38	2.39	2.06	1.83	1.28
	`FullSub`	**0.80**	**2.89**	**2.77**	**2.50**	**2.25**	**2.62**
	`RNN-Mod`	0.69	2.49	2.42	2.07	1.72	0.73
	`Ours`	0.77	2.68	2.63	2.31	2.04	1.92
Real-time	`Demucs48`	**0.78**	2.35	2.37	2.04	1.81	1.20
	`FullSub`	0.76	2.60	2.59	2.24	2.00	1.65
	`RNN-Mod`	0.72	1.79	2.32	1.77	1.88	-1.82
	`Ours`	0.77	**2.68**	**2.63**	**2.31**	**2.04**	**1.92**

7.3.2 Real-time speech denoising quality

We compare our denoising model against several recently proposed real-time denoising networks, including `Demucs48` [130], `FullSub` [131], and `RNN-Mod` [132]. We train these models on the same aforementioned datasets, and evaluate denoising quality in both real-time and offline settings. In offline setting, the audio signal is provided at once, and thus no sliding window is needed. By comparing denoising quality of the two settings from the same network, we wish to understand to what extent a sliding window strategy affects the denoising quality.

For `Demucs48`, we use their provided sliding window implementation. `FullSub` and `RNN-Mod` do not provide streaming implementations, and we found that their denoising quality becomes unstable when dynamic sliding window is added. Therefore we adopt for them the fixed-size sliding window of size 80ms with 16ms padding on the past and future ends. We choose this sliding window configuration because our experiments show that it leads to the best possible real-time denoising quality while keeping an acceptable delay.

Table 7.1 summarizes the evaluation results on AAD dataset. Our model has the best or on-par real-time denoising quality for all the quality metrics. Also worth noting is that our model is the

only one that ensures the real-time denoising quality the same as its offline counterpart, thanks to the data padding strategy. All other models experience quality degradation when they are switched from offline setting to real-time setting. Moreover, the results of real-time denoising quality on `Valentini` benchmark are reported in Table 7.2.

7.3.3 Real-time performance

Controlled experiments. First, we measure D_{N} and D_{A} of different network models (see Table 7.3). Since these models take in different lengths of input data, we also measure their network running time for processing a 200ms signal at once without splitting it into multiple windows. All the measurements are done without heavy background processes. The results indicate that in a dedicated computing environment, our network is as fast as the state-of-the-art models.

Next, we perform controlled experiments to understand the real-time performance of dynamic sliding window and fixed-size sliding window in presence of CPU resource fluctuation. To this end, we create two denoise models: both use the same denoising network (in Section 7.2.2) trained on AAD dataset; the first one uses dynamic sliding window (referred as D-model) while the second uses fixed-size sliding window (referred as F-model). For fair comparison, the initial window size L_0 in D-model is set the same as the fixed window size in F-model (L_0 = 16ms). We use the two models to denoise the same set of audios, each of which has a length of 5s. To impose computing power fluctuation, after 2 seconds we deliberately delay the network processing by a factor of s where s is randomly chosen from $[1, 7]$. This is to simulate CPU occupancy by background processes in a controlled way, as we ensure the same amount of delays are added to both D-model and F-model.

Figure 7.3 shows the measured D_{N} and D_{A} as the audio stream arrives over time. At the beginning, both can run smoothly in real-time (with the playback lag $N_{\mathrm{A}} < 100$ms). After 2 seconds, the computing power starts to fluctuate, causing some sliding windows not to be processed in time. As a result, the playback lag D_{A} of the F-model accumulates, and the output audio playback becomes choppy. In contrast, D_{A} of the D-model stays stable, because it can dynamically increase window size to catch up the delay. This experiment confirms our theoretical analysis in (7.1) and (7.2).

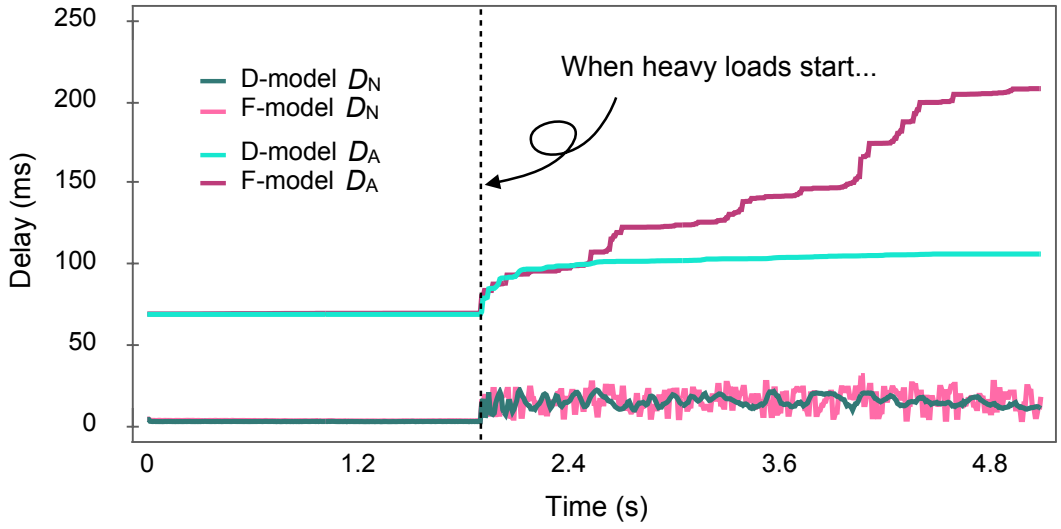


Figure 7.3: **Comparison of real-time performance between dynamic and fixed sliding window approaches.** We artificially increase the network processing time randomly by 1-7 times after 2 seconds to simulate CPU power fluctuation. The curves are averaged results over 100 trials with a fixed random seed.

Table 7.2: **Real-time quantitative comparisons.** Real-time denoising quality on `Valentini`.

Methods	STOI	CSIG	CBAK	COVL	PESQ	SSNR
`Demucs48`	**0.94**	**4.46**	**3.63**	**3.92**	**3.34**	7.77
`FullSub`	0.92	3.76	3.51	3.46	3.22	8.40
`RNN-Mod`	0.92	3.96	3.02	3.48	3.02	0.04
`Ours`	0.93	4.09	3.62	3.67	3.27	**9.56**

Real-world experiments. We then examine the real-time performance of our model in real scenarios wherein the background processes may preempt CPU cycles. Here, we denoise the same set of audios while different software is running on the background, including 4K video playing, Zoom conference, iMovie video editing, and a video game *Apex*. To run Apex game, we use a Windows10 PC with an 8-core Intel CPU (3.60GHz i7-9700K) and a GPU (NVIDIA GeForce RTX 2070 SUPER); tests with other software are performed on a Macbook Pro (2.3GHz Intel Quad-Core i5). We choose these software because they demand increasingly more CPU cycles.

Table 7.3: **Comparison of timings.** In addition to D_{N} and D_{A}, we also report cost of network inference for 200ms audio (S_{N}). Here the numbers include the mean and std. of the timings.

Methods	D_{N}	D_{A}	S_N
`Demucs48`	8.9 ± 0.1	**58.4** ± 0.7	23.7 ± 1.2
`FullSub`	64.7 ± 0.8	182.0 ± 6.3	96.6 ± 9.6
`RNN-Mod`	3.3 ± 0.1	101.8 ± 0.9	**4.5** ± 0.8
`Ours`	**2.7** ± 0.1	61.6 ± 0.9	9.0 ± 1.0

Table 7.4: **Real-time denosing lag** for `Demucs48` and `Ours` when running other background software. Each cell shows $D_{\mathrm{A}}(D_{\mathrm{N}})$ with both mean and std. Numbers are measured using a 20s audio.

Software	`Demucs48`	`Ours`
None	67.5±2.7 (9.7±0.5)	**65.3±1.2 (3.6±0.0)**
4K Video	69.2±4.8 (12.0±0.4)	**67.8±6.92 (3.6±0.1)**
Zoom	74.7±16.1 (13.2±0.4)	**65.8±3.0 (3.9±0.1)**
iMovie	3938.5±1167.3 (19.1±0.9)	**82.4±3.7 (6.3±0.3)**
Apex	15789.3±9501.6 (28.1±8.2)	**167.2±28.7 (11.4±2.4)**

We run these software individually, and meanwhile measure D_{N} and D_{A} using our model and `Demucs48`, respectively. We compare with `Demucs48` because it has its own streaming implementation and offers comparable denoising quality to ours. The results are reported in Table 7.4. As the background process becomes much computationally intensive (e.g., iMovie and Apex), the playback lag of `Demucs48` increases drastically, whereas the lag of our model stays mild and stable. This is clear evidence indicating the performance advantage of our model and the dynamic sliding window strategy.

7.4 Summary

We have proposed a dynamic sliding window strategy for real-time speech denoising. Through careful analysis and experiments, we demonstrated its advantages over the widely used fixed-size

sliding window strategy. Our denoising network achieves realtime denoisnig quality comparable to the SOTAs, while keeping the lag low by utilizing the dynamic sliding window. Remarkably, it allows our model to run robustly in real-world scenarios in presence of other background tasks.

Part II

Personalized Dereverberation of Speech

Chapter 8: Overview

Today, companies and organizations have adopted remote work as a major component of how they function and operate. As a result, online communication has become an integral part of our daily routine. This form of communication, however, tends to be less personal due to several technological limitations. One of these is the fact that the participants/users often sound like they are located in very different spaces. This is because the environment that each user resides in (room, office, classroom, etc.) ends up having a distinct signature on the user's speech. The user's speech is distorted by echoes (reverberations) that depend on the acoustic properties of their environment as well as their location within it [138]. For example, a person's voice in a small tiled kitchen would sound very different from their voice in a large concrete lecture room. Furthermore, the person's voice can sound quite different (less or more tinny, for instance) as they move from one side of the tiled kitchen to the other. This is one reason why online communication lacks the intimacy of an in-person conversation. A critical step towards making users in different locations feel like they reside in the same "space" is to dereverberate their speech—i.e., remove the effects of their respective environments.

In theory, dereverberation requires a precise impulse response. Many traditional dereverberation methods assume prior knowledge of the impulse response [139, 140, 141]. When the impulse response is known, a deconvolution process can be applied to the captured audio signal to remove the reverberations [6]. A classical approach to deconvolving a signal is Wiener filtering [142], as it yields the best results in terms of mean-squared error.

In virtually any real-world scenario, however, it is impractical to obtain the precise impulse response. This is because the impulse response depends on i) the surrounding environment, ii) the sound source (user) location, and iii) the receiver's (microphone) location. The acoustic property of the environment is affected by many physical factors, including the room's geometry and material

properties. In the case of online communication, we would need to measure impulse responses for all possible pairs of microphone and user locations within an environment, which is simply impractical. For all these reasons, almost all the recent approaches have focused on blind dereverberation, requiring no prior knowledge of the impulse response.

Unfortunately, blind dereverberation remains rather challenging due to its limited ability to generalize to *all* environments. Our examples in the supplementary material demonstrate that existing blind techniques can produce strong artifacts in many real-world scenarios. Often, not only does some of the reverberation persist in the processed signal, but it also suffers from the removal of high frequencies, which causes muffling of the original speech.

In this part of the thesis, we revisit the dereverberation problem from the perspective of online communication. We observe that people often conduct their meetings from one or a small number of spaces, such as a study room, office, lecture room, etc. The listener in these cases is the microphone on a computer monitor or a display, which usually stays more or less fixed—sitting on a desk or attached to a wall—within the environment. The sound source (user) is in front of the computer monitor but can move around within a region while maintaining a view of the monitor. In this setting, two of the three factors—the environment and the microphone—are more or less fixed, and the impulse response is mainly impacted by the location of the source (user). This allows us to rethink the classical Wiener filtering approach for speech dereverberation and develop a new method.

The proposed dereverberation method requires a simple, one-time personalization step. We use an automated method for measuring the impulse response for a single location of the user and then have the user read a short paragraph from within the region they are expected to occupy. This information enables our network to generalize the measured impulse response to all locations within the region of interest. The end result is high-quality dereverberation of the user's speech while enabling them to move within their environment.

The proposed personalized approach quantitatively outperforms the state-of-the-art dereverberation methods on both synthetic data and real-world recordings. User studies show that our system is highly preferred over other methods and baselines.

Chapter 9: Related Work

9.1 Speech dereverberation

Speech dereverberation [143] has been studied since the 1970s [144, 145], and it remains one of the most challenging problems in audio processing. From a signal processing perspective, acoustic reverberation can be viewed as filtering, that is, a clean signal convolved with the room impulse response (RIR). Classical dereverberation methods use a known RIR to deconvolve the reverberant signal [139, 140, 141]. Among them, the most widely used method is the Wiener filter [6], which enhances speech signals by optimizing the mean-squared error. However, all the deconvolution approaches mentioned above require the RIR to be precisely known. Many works have shown that a slight change in RIR can lead to a dramatic drop in dereverberation quality [146]. This is why classical deconvolution-based dereverberation methods often generate sub-optimal results in real-world scenarios.

In order to be more robust to RIR variations, techniques such as sub-band envelope deconvolution [147, 148] and envelope expansion [149] have been developed, which aim to increase the modulation depth of the reverberant speech signals. Years later, the concept of blind deconvolution was proposed [150, 151], where the goal is to jointly estimate the RIR and the dereverberated signal.

Numerous approaches to blind speech dereverberation have been proposed. In [152], the idea of sparsifying the linear prediction (LP) residual of the reverberant speech is introduced. In [153], the LP residual of the clean speech is reconstructed by finding the extrema of wavelet coefficients. In [150], the authors sparsify the LP residual by kurtosis optimization. In [154], a maximum-likelihood sparsification method for LP residual is proposed. Several other works [155, 156, 157] also utilize the LP residual to perform speech dereverberation.

Spectral subtraction [28] is also commonly used for speech dereverberation [158]. Works such as [157, 156] combine the use of spectral subtraction with the LP residual. In [159, 160, 161],

spectral subtraction is extended to the multi-channel case, and the statistical model of the late reverberation is assumed to be Gaussian noise modulated by a decaying exponential function.

The weighted prediction error (WPE) algorithm [162, 163] is a well-established approach to speech dereverberation. WPE is an iterative method that estimates a filter to predict the current reverberation tail at each time frame. It is well known that WPE can suppress late reverberations to a large extent. As a result, many extensions to WPE have been proposed [164, 165, 166, 167].

Another class of methods uses microphone arrays for speech dereverberation. Among them, beamforming [168] is one of the most popular multi-channel processing approaches. The objective of beamforming is to enhance the speech signals arriving from one direction (the user, in our case) while suppressing the signals from other directions. Since sound signals propagate as plane waves, they arrive at each microphone in the array at a slightly different time. These time delays are exploited to enhance or suppress a received reverberant signal's components to achieve dereverberation [169, 160, 170].

More recently, deep learning has been applied to speech dereverberation. Using the plethora of speech data available today, waveform-based networks [130, 171, 172, 173, 174, 175] learn a direct mapping from the reverberant and noisy speech signal to its corresponding clean speech signal. However, waveform-based methods often introduce distortions in the output speech signals. Sometimes the higher frequencies of the human voice get suppressed, creating a muffling effect. Instead of working with waveforms, some methods operate in the time-frequency domain. Methods such as [176, 177, 178, 179, 77] consider both the magnitude and phase information of the Short-time Fourier Transform (STFT) as inputs to the network. Unfortunately, time-frequency-based methods also suffer from artifacts such as phase-induced artifacts and time-aliasing artifacts [84, 87, 88]. Generative adversarial networks (GANs) improve the generated speech signals by utilizing adversarial loss from discriminators. GAN-based methods such as [180, 181, 182, 124, 183] have achieved impressive results given that they are addressing a hard problem — blind dereverberation. However, from the perspective of our goal of achieving the level of quality that is required to enhance online communication, the quality of the processed speech remains inadequate.

9.2 Room acoustic estimation

Many works focus on estimating the room acoustics to help improve the quality of downstream applications such as speech enhancement, speech recognition, speech separation, audio forensics, etc. The works reported in [184, 185] synthesize RIRs by learning from real RIRs. Image source [186] and path tracing [187] are used to generate synthetic room acoustic information (RIRs). In [188], the volume of a room is estimated from noisy speech signals. Methods such as [189, 190, 191] blindly estimate reverberation times and direct-to-reverberant ratios from reverberant speech signals. As we will demonstrate, our *personalized* dereverberation method produces results that are a significant improvement over existing approaches.

Chapter 10: Method

Both the environment and the location of the speaker within it can affect the reverberation of a speech signal. In our setting, the microphone is fixed in the environment, and the speaker (user) is assumed to be within a region of usage. By assuming a single environment and a single user within it, our goal is to be able to dereverberate the user's speech in a way that is insensitive to both the person's location with the region of usage as well as to the expected variability in the user's voice. The latter is important as a person's voice can change over the course of a day due to health conditions such as colds and sore throats.

Our personalization step is simple and efficient—it is done before using the system and does not need to be repeated. We measure a single representative room impulse response (RIR) and capture a few minutes of reverberant audio along with its corresponding clean audio.

This chapter is organized as follows: Section 10.1 describes the formulation of reverberation and the reasons for which a general blind dereverberation method cannot be expected to work well for all types of environments. Section 10.3 to Section 10.4 details the personalization process and describes our dereverberation method that generalizes the representative RIR to the spatial neighborhood.

10.1 Background on dereverberation

The sound waves emitted by a source (user) travel along multiple paths before arriving at the receiver (microphone). The microphone receives waves not only directly from the user but also indirectly after reflections and diffractions due to the environment. The end result is a reverberated received signal. From a signal processing perspective, given a source signal x, the signal y received by the microphone can be modeled as

$$y = x * h + n, \tag{10.1}$$

where h is the RIR between the user and the microphone, and n is the additive noise due to background sound from other weaker sources in the environment and/or the noise introduced by the microphone itself.

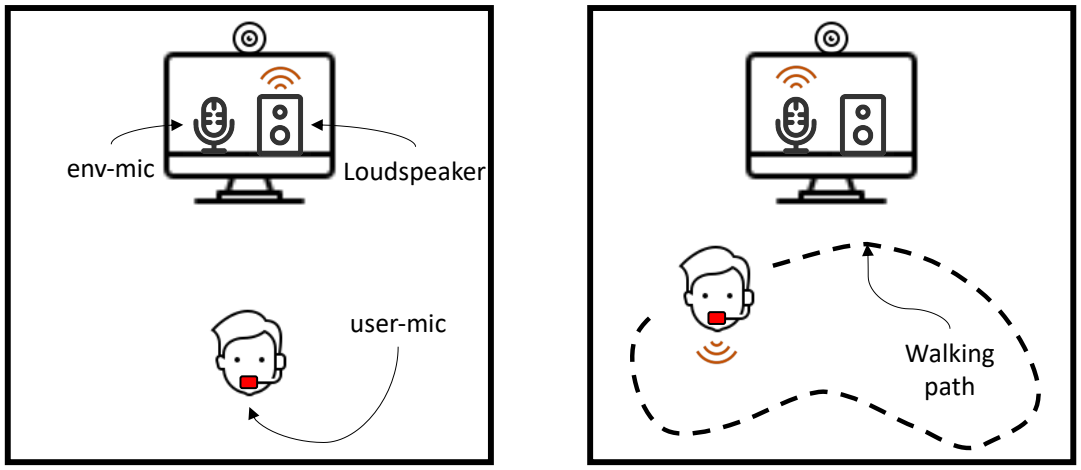
Dereverberation is the inverse of this process. From a received signal y, we wish to recover the clean source signal x. Many traditional methods, such as Wiener filtering, assume that the RIR h is known. Even with this assumption, x may not be precisely recovered from y. This is because the convolution of x with h in (10.1) is a multiplication in the frequency domain and may result in the loss of some frequencies in y (e.g., when h at certain frequencies is close to zero). Nevertheless, if h is precisely known, the dereverberation results using traditional methods can yield good results.

Unfortunately, in practice, it is hard, if not impossible, to know h *a priori* for all user locations of interest. Therefore, most of the recent works aim to perform dereverberation without knowing h, i.e., blind dereverberation. Blind dereverberation is even more challenging than the traditional approach of using a known RIR because it has to (implicitly) infer h from the signal y.

Today, blind dereverberation is most often implemented using neural networks. Training a neural network for blind dereverberation requires an enormous amount of data due to the large space of possible RIRs a source signal may be subjected to—the RIR depends on the room's geometry and material properties, as well as the source and receiver locations. As a result, the current neural-network-based dereverberation methods have limited generalization ability; when dereverberating signals in the wild, they suffer from artifacts.

10.2 Representative RIR

Let the user stand roughly at the center of the space within which they would conduct online communication. The representative RIR (`rRIR`) is the RIR from the user at this location, denoted by P_{user}, to the location of the microphone (`env-mic`), denoted by $P_{\text{env-mic}}$. The typical method to measure RIRs requires the audio source to emit a sine-sweep and record it using the microphone. Please note that we choose the sine-sweep method (ESS) over the Maximum Length Sequence (MLS) because of its robustness as described in [192, 193]. However, emitting a sine-sweep is



(a) (b)

Figure 10.1: **The personalization step.** (a) The user wears a head-worn microphone `user-mic` and stands approximately at the center of the usage region. The computer's loudspeaker emits a sine-sweep signal received by both `user-mic` and `env-mic`. The representative RIR (`rRIR`) is calculated from the signals captured by the two microphones. (b) The user then moves around the region while reading a brief passage, which is recorded by both microphones.

beyond human capability. Thus, we exploit the *reciprocity of RIR*.

Let $RIR(P_{\text{user}} \mid P_{\text{env-mic}})$ be the RIR from the user to `env-mic`. According to acoustic reciprocity, the RIR between a source and a receiver remains the same if the two are swapped [194, 195, 196]. That is,

$$RIR(P_{\text{user}} \mid P_{\text{env-mic}}) = RIR(P_{\text{env-mic}} \mid P_{\text{user}}). \tag{10.2}$$

We require the user to wear a head-worn microphone (`user-mic`) during this phase as shown in Figure 10.1. Furthermore, we assume that the loudspeaker on the computer is roughly co-located with the `env-mic`. Then,

$$RIR(P_{\text{user}} \mid P_{\text{env-mic}}) \approx RIR(P_{\text{ls}} \mid P_{\text{user-mic}}), \tag{10.3}$$

where P_{ls} is the location of the loudspeaker and $P_{\text{user-mic}}$ is the location of the `user-mic`.

We now emit a sine-sweep over the loudspeaker and record it simultaneously by both the

`env-mic` and the `user-mic`. `rRIR` is then calculated using the method described in [192]. In our implementation, the sine-sweep frequency ranges from 70 Hz to 20 kHz and is repeated three times for 2 seconds each.

While `rRIR` is accurate for the measured location, it can be an imprecise substitute for the spatial neighborhood.

10.3 Personalization

Since we cannot measure the RIR at every spatial location, we instead capture the *effects* of the spatially varying RIR. We ask the user to read a brief passage for five minutes while freely moving within the vicinity. We record the user's voice using both the `env-mic` and the `user-mic`. The audio recorded by `env-mic` has reverberations while the audio recorded by `user-mic` is treated as clean audio as its signal-to-reverberation ratio is very high.

10.4 Wiener Meets Deep Learning

Recall the reverberation model in (10.1). If the precise RIR h and approximate noise characteristics are known, Wiener deconvolution [142] estimates the dereverberated signal $\hat{x}$ as,

$$\hat{x} = w * y, \tag{10.4}$$

where w is the Wiener filter. For practical reasons, this is usually computed in the frequency domain as,

$$\hat{X} = WY, \qquad W = \frac{H^*}{|H|^2 + \mathrm{NSR}}, \tag{10.5}$$

where $\hat{X}$, W, Y, and H are the Fourier transforms of $\hat{x}$, w, y, and h, respectively, and H^* is the complex conjugate of H. NSR is the expected noise-to-signal ratio, usually set empirically.

However, `rRIR` is only measured at one user location and cannot be generalized to the spatial neighborhood. Performing the naïve Wiener deconvolution of audio captured in the vicinity of

rRIR leads to a substantial and irrecoverable loss in speech information.

The data captured during personalization contains implicit information about how the RIR varies spatially. Leveraging this insight, our proposed dereverberation technique combines the advantages of both the Wiener filter and neural networks in a GAN framework [66]. Figure 10.2 illustrates our proposed model.

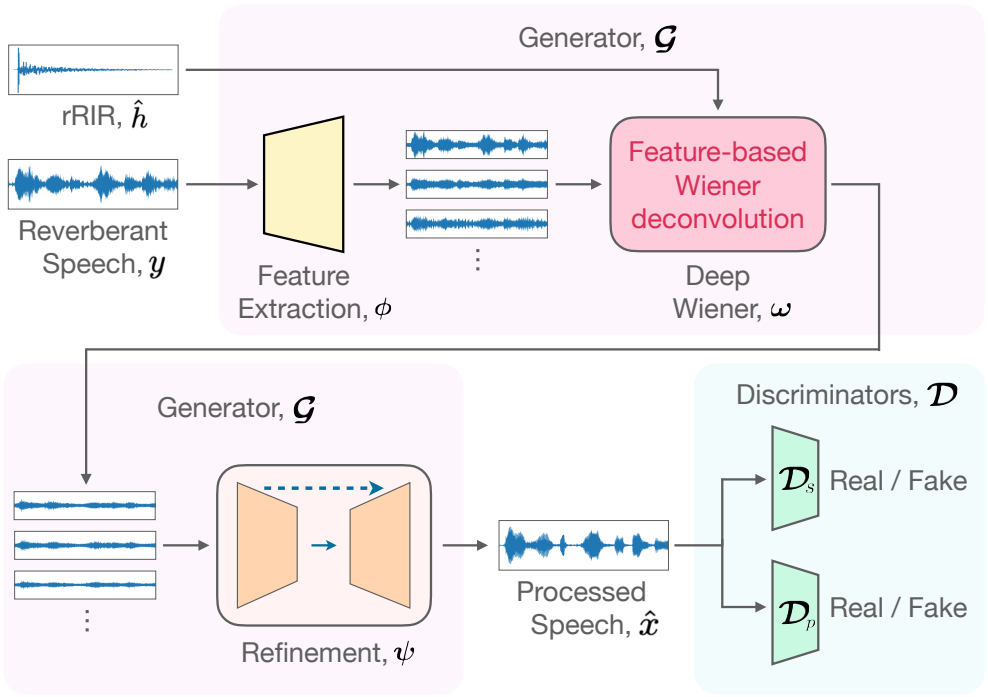


Figure 10.2: **Proposed dereverberation network.** The generator $\mathcal{G}(y, \hat{h})$ accepts both a reverberant signal and an approximate RIR to perform feature-based Wiener deconvolution, which is then enhanced by the refinement module for removing any remaining noises and artifacts. The discriminator $\mathcal{D}$ contains a multi-scale discriminator $\mathcal{D}_s$, and a multi-period discriminator $\mathcal{D}_p$.

Our proposed generator $\mathcal{G}$ accepts a reverberant signal y and rRIR $\hat{h}$ as inputs, extracts features of y in a high-dimensional latent space (feature extraction, $\boldsymbol{\phi}$), performs feature-based Wiener deconvolution (deep Wiener, $\boldsymbol{\omega}$), and finally refines and projects features back to a 1-dimensional signal (refinement, $\boldsymbol{\psi}$).

10.4.1 Feature extraction

Feature extraction $\boldsymbol{\phi}$ takes a reverberant signal y as input, learns to extract useful features, and projects them to an n-dimensional latent space ($n = 32$ in our implementation), formulated as,

$$\boldsymbol{\phi}(y) = \left\{\boldsymbol{\phi}(y)_i\right\}_{i=1}^{n} : \mathbb{R}^1 \rightarrow \mathbb{R}^n. \tag{10.6}$$

Intuitively, we seek to extract representations of the audio signal that explicitly support Wiener deconvolution using $\texttt{rRIR}$ $\hat{h}$, but also generalize to the spatial neighborhood.

This module consists of one Conv1d layer (kernel size $k = 15$, stride $s = 1$ and padding $p = 7$) with ReLU followed by two residual blocks ($k = 15$, $s = 1$ and $p = 7$).

10.4.2 Deep Wiener

Partially inspired by [197], we propose our feature-based deep Wiener operation $\boldsymbol{\omega}$, which is analogous to (10.4), as,

$$\boldsymbol{\omega}(y, \hat{h}) = \hat{w} * \boldsymbol{\phi}(y)_i, \quad \forall i = 1, \ldots, n. \tag{10.7}$$

Here we perform *channel-wise* Wiener deconvolution with $\boldsymbol{\phi}(y)_i$, each of the n extracted features of y, and $\hat{w}$, the Wiener filter obtained by utilizing $\hat{h}$, the imprecise $\texttt{rRIR}$. In frequency domain, $\hat{w}$ can be written as,

$$\hat{W} = \frac{\hat{H}^*}{|\hat{H}|^2 + \mathsf{NSR}}, \tag{10.8}$$

where $\hat{H}$ is the Fourier transform of $\hat{h}$. We use $\mathsf{NSR} = 0.1$ regardless of the noise level of the reverberant signal.

In short, this module deconvolves the input reverberant signal with $\texttt{rRIR}$, aiming to remove most of the reverberation existing in the signal. Since (1) $\texttt{rRIR}$ is *not* the exact RIR used to generate the input signal, and (2) NSR is set at a *fixed* value, this process leads to residual reverberations and

artifacts, which are handled by the refinement module.

10.4.3 Refinement

Now that we obtain $\boldsymbol{\omega}(y, \hat{h})$, we perform a refinement process $\boldsymbol{\psi}$ that enhances the signal in the latent space and then projects back to a 1-dimensional audio signal as,

$$\boldsymbol{\psi}\big(\boldsymbol{\omega}(y, \hat{h})\big) = \hat{x}. \tag{10.9}$$

Specifically, $\boldsymbol{\psi}$ is a U-Net-based multi-layer convolutional encoder and decoder with skip connections. The encoder gets as input the 32-dimensional signal from the previous module and outputs its latent representation. Each of the 5 layers of the encoder consists of a Conv1d layer ($k = 8$, $s = 4$) followed by ReLU, another Conv1d layer ($k = 1$, $s = 1$), and finally a GLU activation. Then the latent representation is fed into a unidirectional LSTM network with 2 layers followed by a linear layer. Finally, the 5-layer decoder network outputs the dereverberated signal. Each decoder layer consists of a Conv1d layer ($k = 1$, $s = 1$) followed by a GLU activation, another Conv1d layer ($k = 8$, $s = 4$), and finally a ReLU function.

In summary, the proposed generator $\boldsymbol{\mathcal{G}}$ described above and shown in Figure 10.2 can be mathematically described as,

$$\begin{aligned} \hat{x} &= \boldsymbol{\mathcal{G}}(y, \hat{h}) \\ &= \boldsymbol{\psi}\big(\boldsymbol{\omega}(y, \hat{h})\big) \\ &= \boldsymbol{\psi}\left[\left(\hat{w} * \boldsymbol{\phi}(y)_i\right)_{i=1}^{n}\right]. \end{aligned} \tag{10.10}$$

10.4.4 Discriminators

The discriminator module $\boldsymbol{\mathcal{D}}$ essentially evaluates whether the generated audio has reverberation or not. We use the multi-scale discriminator proposed in MelGAN [198], which evaluates audio samples at different scales. In our work, the discriminator operates at three scales: audio down-

sampled by a factor of 1, 2, and 4. We also leverage the multi-period discriminator proposed in HiFi-GAN [199], which evaluates audio based on implicit structures at different periodic segments. As suggested by the paper, we set the periods to $[2, 3, 5, 7, 11]$.

10.4.5 Training objectives

The objectives for our GAN model are:

$$\mathcal{L}_{\mathcal{G}} = \mathcal{L}_{GAN}(\mathcal{G}, \mathcal{D}) + \lambda_{FM}\mathcal{L}_{FM}(\mathcal{G}, \mathcal{D}) + \lambda_{MEL}\mathcal{L}_{MEL}(\mathcal{D}), \tag{10.11}$$

$$\mathcal{L}_{\mathcal{D}} = \mathcal{L}_{GAN}(\mathcal{D}, \mathcal{G}), \tag{10.12}$$

where $\mathcal{L}_{GAN}(\mathcal{G}, \mathcal{D})$ and $\mathcal{L}_{GAN}(\mathcal{D}, \mathcal{G})$ are the standard GAN losses [66].

The feature matching loss, $\mathcal{L}_{FM}(\mathcal{G}, \mathcal{D})$, is a similarity metric between the clean and generated signals, defined as,

$$\mathcal{L}_{FM}(\mathcal{G}, \mathcal{D}) = \sum_{i=1}^{T} \frac{1}{N_i} \| \mathcal{D}^{(i)}(\hat{x}) - \mathcal{D}^{(i)}(\mathcal{G}(y, \hat{h})) \|_1, \tag{10.13}$$

where $\mathcal{D}^{(i)}$ denotes the i-th layer feature map output of the discriminator; T is the number of layers in the discriminator; N_i denotes the number of features in the i-th layer.

The Mel-spectrogram loss, $\mathcal{L}_{MEL}(\mathcal{G})$, compares the spectrogram of the clean signal with that of the generated signal.

$$\mathcal{L}_{MEL}(\mathcal{G}) = \| \delta(\hat{x}) - \delta(\mathcal{G}(y, \hat{h})) \|_1, \tag{10.14}$$

where δ is the function that transforms a waveform to a Mel-spectrogram. This ensures that the generator produces realistic audio signals that match the clean ones across all frequencies.

Chapter 11: Implementation

As described in Section 10.4, our model accepts as inputs a reverberant speech signal y, and an imprecise RIR captured in the vicinity of the speaker, $\hat{h}$. Both are in the waveform domain sampled at 16 kHz. The length of y is at least as long as h. To train the model in a supervised manner, we need a clean (noise and reverberation-free) copy of the reverberant signal. We take a two-stage approach to train the model: (1) pre-train with synthetic data and (2) fine-tune with personalization data.

11.1 Pre-training with synthetic data

We construct a large synthetic dataset using publicly available datasets with the following steps: (1) We randomly select 21,600 1-second speech clips (sampled at 16 kHz) for a total duration of 6 hours from LibriSpeech [200] as clean audio signals, of which 18,000 are sampled from LibriSpeech's training set and used to build our training set, and 3,600 are sampled from LibriSpeech's test set for testing. (2) RIRs (1-second long at 16 kHz) are sampled from the Aachen Impulse Response (AIR) dataset [201]. This dataset consists of real-world RIRs measured in various environments, such as a booth, office, meeting room, etc., with the reverberation time RT_{60} ranging from 0.08s to 0.83s. We split the dataset by environments so that the set of environments for testing is different from the set for training. For each environment, AIR provides multiple RIRs measured at different locations in close proximity, from which we randomly select two different RIRs – one to construct a synthetic reverberant signal and the other to use as the corresponding `rRIR`. Such RIR pairs are sampled for each of the 21,600 speech clips. (3) We also add background noises randomly chosen from the BUT Reverb dataset [202] (also at 16 kHz) with an SNR between 10 dB and 30 dB. As a result, we obtain 21,600 pairs of reverberant signals and `rRIRs` for training and testing without any overlap.

Pre-training using such a large synthetic dataset enables the model to generalize the input `rRIR` to its vicinity and treat dereverberation as a general problem.

11.2 Fine-tuning with personalization data

After pre-training with the synthetic data, we fine-tune (personalize) our network per user, per environment using user- and environment-specific reverberant signals and RIRs. We follow the steps below to get training data: (1) One `rRIR` is measured following the procedure in Section 10.2. This is used for both the training and testing tasks. (2) We collect a pair of the reverberant signal (recorded by the `env-mic`) and its corresponding clean signal (recorded by the `user-mic`) for each user within each environment for about 5 minutes (described in Section 10.3). (3) We collect an additional 3 minutes of data for testing and reporting metrics. Please note that since the user is moving around during this step, the underlying RIR for the captured speech keeps varying and does not match the `rRIR` captured above. The user also reads different text excerpts during fine-tuning and testing and ensures that the network never encounters testing data during the training phase of the personalization step. Once the network has been fine-tuned with this data, the user no longer needs to wear the `user-mic`, and only the `env-mic` is used. With this **one-time** personalization process, our method is able to learn the implicit acoustic information of the user's voice and environment and produce high-quality dereverberation results when speaking.

In addition, to ensure data diversity for method comparison, we have recorded 4 individuals—two males and two females, native and non-native speakers—in 5 distinct environments, including a conference room, a tiled kitchen, a wood-floored bedroom, a glass-walled office, and a carpeted study room. Therefore, we have obtained $3 \times 4 \times 5 = 60$ minutes of real-world audio for method evaluation. Results are shown in Table 12.1 and Table 12.2.

Chapter 12: Evaluation

We now present our evaluation details, including **i)** comparisons to classic and network-based dereverberation methods (on both synthetic and real data), **ii)** user studies, and **iii)** ablation studies. We refer the reader to the supplementary materials for audio effects and the demonstration of our method in real-world scenarios.

12.1 Method comparison.

We compare our method with several existing speech dereverberation methods, including both the classic and the learning-based approaches. We refer to these methods as follows: **i)** `Ours`, our proposed method. **ii)** `Wiener` [142], the classic Wiener filtering method which takes as input both the reverberant speech and an RIR. `Wiener` provides the optimal result when the RIR is precise. **iii)** `WPE` [203], weighted prediction error, is another widely used classic dereverberation method based on linear prediction. **iv)** `DEMAND` [130], a recently proposed learning-based speech enhancement method that can handle the dereverberation task well because of its reverberant data augmentation in training. **v)** `HiFi-GAN` [180], one of the state-of-the-art speech dereverberation methods based on generative adversarial networks.

We compare all learning-based methods by training and testing on the same dataset. For classic methods, we directly feed in required data as input using the same testing dataset.

12.2 Quantitative evaluations

We use the following widely used metrics to evaluate the quality of dereverberated audio. **i)** Perceptual Evaluation of Speech Quality (PESQ) [115], **ii)** Short-Time Objective Intelligibility (STOI) [117], **iii)** Frequency-Weighted Segmental Signal-to-Noise Ratio (FW-SSNR) [118], and **iv)** Speech-to-Reverberation Modulation Energy Ratio (SRMR) [204]. These metrics have been widely

Table 12.1: **Comparisons on synthetic dataset.** We measure dereverberation quality on the synthetic dataset under four metrics. The best score under each metric is highlighted.

Method	PESQ	STOI	FW-SSNR	SRMR
`Clean`	4.64	1.00	35.00	7.98
`Noisy`	1.82	0.90	7.72	6.85
`Wiener` [142]	1.32	0.79	6.17	4.72
`WPE` [203]	1.85	0.90	7.86	7.37
DEMAND [130]	1.54	0.87	9.03	7.34
`HiFi-GAN` [180]	2.11	0.90	10.06	7.41
`Ours`	**2.40**	**0.92**	**11.03**	**7.56**

used for evaluating speech processing quality.

12.2.1 Evaluation on synthetic dataset

The metric values on our synthetic dataset are reported in Table 12.1. Note that `Wiener` requires both the reverberant signal and the RIR as inputs; and for a fair comparison, we feed the same inputs as the ones to our proposed method (i.e., one reverberant signal and `rRIR`). The other methods are blind dereverberation methods, and therefore only a reverberant signal is used as the input. Under all metrics, our method consistently outperforms others. In particular, according to SRMR scores, our method results in the least amount of reverberation among all the methods.

12.2.2 Evaluation on real recordings

We also compare our method against others using real recordings, and the results are reported in Table 12.2. For a fair comparison, we fine-tune all learning-based methods, including ours, on the same real recordings obtained from the personalization step and then evaluate on different real recordings of the same speaker. `Wiener` and `Ours` require an RIR as a part of the input. For both methods, we use `rRIR`, which is not necessarily the precise RIR from the speaker to the receiving microphone. Our method outperforms all other methods by an even more significant margin than in the synthetic case, demonstrating its robustness and practicality in real scenarios.

Table 12.2: **Comparisons on real recordings.** All learning-based methods are fine-tuned on the recorded speech signals from the personalization step.

Method	PESQ	STOI	FW-SSNR	SRMR
`Clean`	4.64	1.00	35.00	8.58
`Noisy`	1.60	0.67	7.71	4.30
`Wiener` [142]	1.59	0.67	6.55	4.59
`WPE` [203]	1.70	0.71	7.42	5.91
DEMAND [130]	1.03	0.63	4.31	4.72
`HiFi-GAN` [180]	1.67	0.77	8.12	6.53
`Ours`	**2.06**	**0.88**	**9.53**	**8.46**

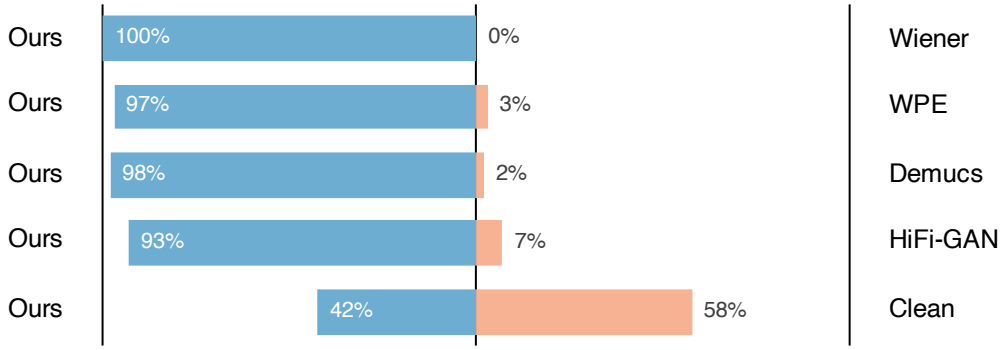


Figure 12.1: **User study results.** We conduct a preference test between the dereverberation results of our method and those of the comparison techniques. Untrained listeners overwhelmingly favor our results. Even when compared to clean speech, our results are preferred nearly half of the time.

12.3 Qualitative user study

We conduct a user study to evaluate the perceptual quality of our method with untrained listeners. We randomly choose three clips from our dataset of real recordings and create pairs of results. Each pair contains audio processed by our method and by one of the comparison methods. Additionally, we create pairs of our result and the corresponding clean audio. In total, we create 15 pairs and invite 33 participants, including 18 male and 15 female, to choose a preferred audio within each pair in a blind study analogous to [180].

As shown in Figure 12.1 the listeners unambiguously favor the results from our method. Interestingly, although WPE does not remove reverberation well enough (according to Table 12.1 and Table 12.2), it preserves the overall clarity of the reverberant signals, and thus it is slightly preferred over `Wiener` and DEMAND. `HiFi-GAN` produces speech clips with fewer artifacts than `Wiener` and DEMAND, but it does not remove reverberations as well as our method does. It is also noteworthy that, compared to clean speech, our method is preferred nearly half of the time. This indicates that our method produces results that are generally perceived as clean as the original signals.

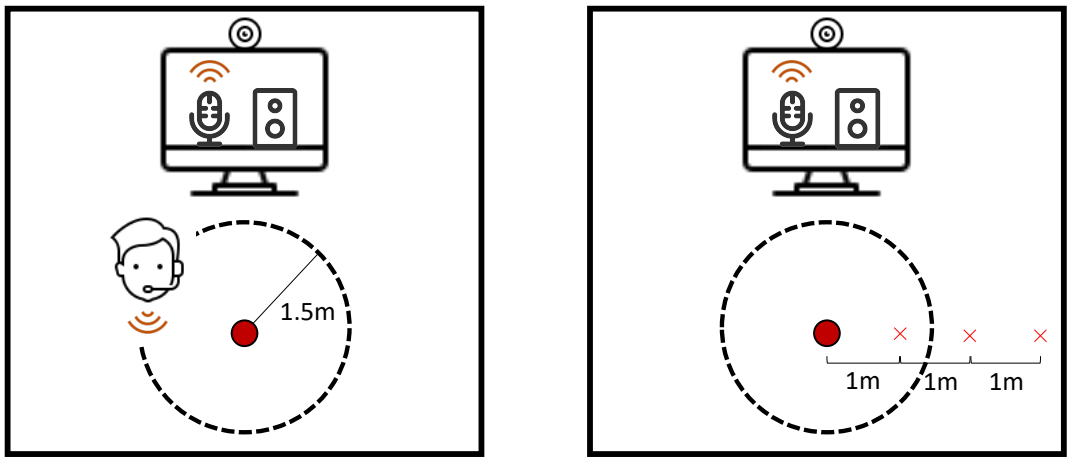
12.4 Robustness to voice variation

Our voices may vary slightly from day to day. Any slight change in conditions of vocal cords (such as a sore throat) results in vocal variation. As everyone has probably experienced, even nasal congestion can make a person sound different. In this experiment, the user first conducts a personalization step without stuffing up his nose. In other words, our proposed network is trained to learn only the user's regular voice. At the inference time, the user is asked to stuff up both of his nostrils. The resulting speech signals are then fed into the previously trained network for dereverberation. The quantitative results are shown in Table 12.3.

Table 12.3: **Robustness to voice variation.** Even with the user's nose congested, the dereverberation quality remains relatively the same, showing our method's robustness to voice variation.

Method	PESQ	STOI	FW-SSNR	SRMR
`Ours-Congested`	2.00	**0.88**	9.45	8.30
`Ours`	**2.06**	**0.88**	**9.53**	**8.46**

In Table 12.3, `Ours` is the result when the user speaks normally, while `Ours-Congested` is the result after stuffing up their nose. As shown by the results, nose congestion, though slightly altering the user's voice, does not cause a significant drop in speech enhancement qualities. This experiment suggests that our method is robust to voice variation.



(a) Personalization step (b) After personalization

Figure 12.2: **Experiment configuration for robustness to location variation.** (a) During personalization, the user is asked to read the passage and walk around the location of the measured RIR in a circular pattern. (b) After the personalization step, the user is asked to talk only at locations at 0m, 1m, 2m, and 3m away from the location where `rRIR` is measured.

12.5 Robustness to location variation

In the personalization step (in Section 10.3), `rRIR` is measured at the center of the region of usage, and then the user is asked to read a passage while walking around in the region. Here we validate the robustness of our method to user location variation. In this experiment, in the personalization step, the user walks in a circular pattern, about 1.5m away from the location where `rRIR` is measured. After the personalization step, the user speaks only at locations that are 0m, 1m, 2m, and 3m away from the location where `rRIR` is measured (see Figure 12.2).

As reported in Table 12.4, `Ours-0m`, `Ours-1m`, `Ours-2m`, and `Ours-3m` indicate results when the user stands 0m, 1m, 2m, and 3m away from the location where `rRIR` is measured. As expected, `Ours-0m` is the best since the user talks right at `rRIR`'s measurement location. The measured RIR is almost identical to `rRIR` for the user's speech. `Ours-1m` and `Ours-2m` receive sightly lower scores than `Ours-0m`, but they are not far off, suggesting our network's strong generalization to locations around `rRIR`'s measurement location. `Ours-3m` shows a larger

Table 12.4: **Robustness to location variation.** The dereverberation quality retains well when the user stays close to the region of usage. But if the user stays too far away from the region of usage, the dereverberation quality largely decreases.

Method	PESQ	STOI	FW-SSNR	SRMR
`Ours-0m`	**2.17**	**0.91**	**9.64**	**8.58**
`Ours-1m`	2.08	0.89	9.56	8.37
`Ours-2m`	1.93	0.88	8.53	7.89
`Ours-3m`	1.74	0.80	7.29	6.65

Table 12.5: **Effectiveness of the personalization step.** Without the personalization step, the dereverberation quality decreases by a large margin. It is clear that the personalization step is the key to a high-quality dereverberation.

Method	PESQ	STOI	FW-SSNR	SRMR
`Ours-NoP`	1.87	0.74	6.72	**8.61**
`Ours`	**2.06**	**0.88**	**9.53**	8.46

decrease in the dereverberation quality on all four metrics since the user talks at a location far from both `rRIR`'s measurement location and the circular region of personalization. We therefore conclude that our method is robust to the user's location variation within the personalization region.

12.6 Effectiveness of personalization

The personalization step is meant to collect data for fine-tuning the network specific to the environment and the user's voice. Here, we compare the dereverberation results by our network with and without fine-tuning on the personalization data.

In Table 12.5, `Ours` and `Ours-NoP` indicate results with and without fine-tuning, respectively. The results show that the dereverberation quality (PESQ, STOI, FW-SSNR) improves drastically with the fine-tuning step. The SRMR score of `Ours` remains on the par with but slightly lower than `Ours-NoP`. This is because the clean speech signals used to fine-tune the network are recorded

Table 12.6: **Ablation studies.** We alter network components and evaluate the dereverberation quality on the synthetic dataset. Each component has its significant contribution to the overall dereverberation quality, showing that each proposed component is necessary.

Method	PESQ	STOI	FW-SSNR	SRMR
Ours-NoDW	2.17	0.90	9.14	6.63
Ours-NoRF	2.10	0.89	8.97	7.38
Ours-NoGAN	2.26	0.89	10.02	7.49
Ours	**2.40**	**0.92**	**11.03**	**7.56**

by the microphone close to the user's mouth, which, as discussed in Section 10.3, might capture a slight amount of reverberation.

12.7 Ablation study

We also perform a series of ablation studies, trained on the synthetic dataset, to understand the efficacy of various components of our proposed network. In Table 12.6, Ours-NoDW refers to the network without the feature-based Wiener deconvolution module (recall Figure 10.2). The network then becomes a blind dereverberation method with only one input (i.e., a reverberant signal) which is directly fed into the refinement UNet. Ours-NoRF refers to the network without the refinement module. The second feature extraction module directly outputs the dereverberated speech signal. Ours-NoGAN refers to the network without the GAN loss. Only the generator network of the proposed GAN-based method is used.

As shown in Table 12.6, the GAN loss mainly helps to improve the sound quality of the speech signals (PESQ, STOI, FW-SSNR), but not so much in terms of reducing reverberation (according to SRMR values). On the other hand, the deep Wiener deconvolution module helps to significantly reduce reverberation, resulting in a large increase in SRMR scores. The refinement module further refines the signals output from the deep Wiener deconvolution module, contributing to a notable increase in the overall quality of the results. Compared to these alternative approaches, our full GAN-based method (referred to as Ours) performs the best.

Chapter 13: Summary

Speech dereverberation is one of the most challenging problems in the speech enhancement community because of reverberation's multiplicative nature within the environment. In this work, we revisit the speech dereverberation problem from the perspective of online communication, where the user is assumed to be within a region of usage. With a simple personalization step, our proposed method learns the room acoustics of the entire region of interest. Our GAN-based network is able to utilize this information and consistently outperform the state-of-the-art dereverberation methods on both synthetic data and real-world recordings. People's strong preferences for our results over the others also confirm the quality of our method.

While our method produces state-of-the-art dereverberation results, we believe certain improvements can make it a practical go-to method. One key aspect is that our current approach is *speaker-specific*. This design choice was made to ensure the highest possible dereverberation quality. However, introducing a *speaker-independent* fine-tuning option would make our system more versatile, though it may involve a trade-off in dereverberation performance.

Another critical factor is the need for real-time, causal audio processing in online communications. Most existing speech dereverberation methods are not capable of real-time operation because they rely on extensive datasets of paired clean and reverberant signals recorded in various environments. These methods often struggle to accurately model the physical properties of the surrounding environment. In contrast, our proposed method, with its simple personalization step, can learn the reverberation characteristics of a specific environment. This capability opens the door to converting our method into a real-time solution by processing streaming audio within the environment where the personalization step was conducted. A potential approach involves using a sliding window technique, where the real-time delay is dominated by the inference time of our method for each window.

However, the sliding window length cannot be too short — typically not less than 1 second — because the reverberation times (RT_{60}) generally range from 0.08 to 0.83 seconds. In larger or highly reverberant spaces like the Sonic Sphere [205], reverberation times can exceed 5 seconds, making real-time dereverberation in such environments potentially unfeasible. Nonetheless, for most typical environments, our method, when combined with a sliding window approach, can function effectively in real-time. A notable drawback is that this real-time method is computationally intensive, particularly on lower-end devices, necessitating further optimization. Additionally, there is an inherent trade-off between dereverberation quality and sliding window size, directly impacting the processing delay: better dereverberation quality leads to more delay. Exploring and optimizing real-time speech processing, particularly in minimizing delay while maintaining quality, is a promising direction for future research.

Part III

DanceCraft: A Music-Reactive Real-time Dance Improv System

Chapter 14: Overview

In the dynamically evolving domain of entertainment, Virtual Reality (VR) and Augmented Reality (AR) technologies are rapidly gaining prominence, revolutionizing everyday activities into immersive digital experiences. This transformation is particularly evident in the sphere of virtual concerts and dance performances, as platforms such as Fortnite, Snapchat, and TikTok spearhead this digital revolution. Users, engaging through their virtual avatars, dynamically interact with music, necessitating the creation of realistic and responsive dance movements in real-time.

Traditionally, dance animation production has been a manual and labor-intensive process. Choreographers and dancers work in tandem to create dance routines that resonate with various music genres, tempos, and styles. These performances are then captured and digitized by specialized teams. Although this method can yield high-quality results, it is both resource-intensive and time-consuming, highlighting the need for more efficient solutions in the swiftly-paced digital entertainment landscape.

Recent advancements in deep learning have enabled the synthesis of music-conditioned dances, as demonstrated in studies such as [206, 207, 22, 208, 209, 210]. These methods replicate the traditional choreography process, aiming to align dance movements with musical rhythms and styles [211, 209, 212, 213]. However, these approaches often lack realism and tend to overlook crucial human elements of dance, such as variety, expressiveness, spontaneity, and personal style, thus diminishing user enjoyment. Our research addresses this gap by focusing on these essential elements, fostering a more personal and immersive connection between users and their dancing avatars. We explore the potential of improvised dance to enrich user experience, posing the question: *What forms of dance, beyond strictly choreographed routines, can enhance the personalization and interactivity of user experiences?*

Improvisational dance, known for its spontaneous and unstructured nature, offers a wide range

of unique interpretations to the same piece of music, thereby enriching the digital user experience [214, 215, 216]. This form of dance, deeply embedded in everyday practices, not only promotes artistic expression but also strengthens the connection between performer and audience [217]. In contrast, choreographed dance involves a precise and methodical process, where dancers strictly adhere to predefined movements, limiting spontaneous expression [218].

This part of the thesis introduces a novel system designed for synthesizing improvisational dance in real-time to live music, with the following key contributions:

1. **Hybrid Modeling Approach:** We introduce a hybrid model that merges data-driven graph-based methods with deep learning techniques for creating music-conditioned 3D improvisational dance. This process begins with the extraction of various music descriptors in response to a music signal. These descriptors guide the selection of dance segments from a database, from which we randomly choose one to match the music's characteristics.

   The selected dance segment is then fused into the ongoing animation using a lightweight state-of-the-art motion in-betweening network. This approach ensures fluid transitions between dance routines, allowing for a seamless and dynamic flow of movements that adapt in real-time to the music.

   The resulting dances are not only spontaneous and engaging but also maintain a strong coherence with the music's genre, tempo, and energy. This balance of unpredictability and musical alignment reflects the essence of improvisational dance.

2. **Comprehensive 3D Dance Dataset:** To tackle the scarcity of diverse dance data, we have compiled an extensive dataset of high-quality dance animations. This collection spans over 8 hours and encompasses a broad spectrum of musical tempos, energies, and genres. Furthermore, our dataset is enriched with natural idle animations and a variety of facial expressions, significantly boosting the avatars' expressiveness and emotional connection during their dance performances.

3. **Interactivity and Customization:** The music that animates the dances can be sourced

from various inputs, such as the end user or a DJ. When users select their own music, they transform from passive viewers into active participants, having direct control over the music and, consequently, the dance animations. For DJs or artists, our system offers the ability to associate popular and recognizable choreographed dance routines with specific music segments, thereby heightening the user's personal engagement and experience.

4. **Personal Avatar Integration:** Our system stands out by allowing users to import their personal 3D avatars from popular AR/VR platforms, instead of limiting them to a few predefined options. We currently support Bitmojis [219], a 3D avatar used by over 250 million Snapchat users [220]. We believe this fosters a deeper connection between users and their dancing avatars.

5. **Integrated Production-Ready System:** We detail both the hardware setup and software implementation of our improvisational dance system. We personalize the user experience by allowing users to cast their Bitmoji 3D avatars by scanning a QR code. Our system is versatile – designed as a web service, it is suitable for deployment on individual kiosks or for large online virtual gatherings. We will release the code and datasets in the future.

Empirical evidence from extensive user studies confirm the effectiveness of our system in delivering an enjoyable, engaging, and highly personalized virtual dance experience.

Chapter 15: Related Work

15.1 Human motion synthesis

Human motion synthesis is a pivotal task in computer animation, characterized by its complexity due to the non-linear and stochastic nature of human movements. Initial methods in this field utilized classic techniques like hidden Markov models [221, 222, 223, 224] and statistical models [225, 226, 227]. These approaches provided valuable insights but were limited in capturing the full dynamics of human motion.

The integration of machine learning has notably advanced this domain. Modern approaches predominantly employ neural networks, which are trained on comprehensive 3D human motion datasets to enhance the realism of synthesized movements. Various neural network architectures have been explored, including CNNs [19, 228, 229], GANs [230], RNNs [20, 231, 232, 233, 234, 235, 236, 237, 238, 207], and transformers [239, 240, 241, 242, 243, 244]. These advancements demonstrate a significant progression from earlier methods, offering more sophisticated tools for generating lifelike human motion.

15.2 Graph-based motion synthesis

Graph-based approaches are a significant category in human motion synthesis. Lamouret et al. [16] introduced the first graph-based method, which generates new motions by assembling existing motion segments from a pre-constructed database. This "cutting and pasting" concept opened up opportunities for synthesizing realistic and physically plausible human motions by reusing and reconfiguring existing motion data to create novel sequences.

Building on this foundational work, subsequent research [245, 246, 247, 248, 249, 250, 251] formalized the concept of motion graphs. In a motion graph, mathematically represented as a directed graph $G = \langle V, E \rangle$, V denotes the set of nodes, and E denotes the set of directed edges. Each node

$v \in V$ represents an individual motion segment, while each directed edge $e \in E$ indicates possible transitions between these segments. Transition probabilities between nodes are often determined based on the similarity of motion segments at the connecting keyframes, ensuring smooth and natural transitions. This structure allows for the automatic generation of motion sequences by traversing the graph, $S = (v_1, v_2, ..., v_n)$, starting from an arbitrary motion segment node v_1 and following directed edges to generate complex and diverse motions from a finite set of base animations.

A key advancement in graph-based motion synthesis was the integration of rhythmic elements into the framework [252, 253, 18]. These methods incorporated beats and musical patterns as constraints, often formulated as optimization objectives, expanding the applicability of motion graphs to rhythmic and dance movements. This alignment between motion and rhythm is crucial in applications involving dance and performance arts, where timing and synchronization with music are essential.

Further innovations in this domain have introduced complex choreographic rules and constraints into the graph-based framework [254, 213, 255]. These enhancements allow for the generation of sophisticated and varied motion sequences that adhere to specific choreographic principles, enabling the creation of motions that are not only physically plausible but also artistically expressive. For instance, modern systems can generate dance movements that follow specific stylistic guidelines or maintain a consistent emotional tone throughout the performance [213].

A principal advantage of graph-based methods is their ability to ensure smooth transitions between motion segments, which is critical for producing seamless and realistic human movements. This capability is particularly valuable in applications such as animation and virtual reality, where motion quality significantly impacts the user's experience. Additionally, the modular nature of motion graphs allows for easy updates and expansions, enabling the addition of new motion segments and transition rules without requiring a complete system overhaul. This flexibility makes graph-based approaches a powerful tool for ongoing developments in human motion synthesis. Even with the advent of learning-based methods mentioned in Section 15.1, which synthesize motion sequences from large datasets, researchers continue to turn to this simple yet elegant class of methods for their

robustness and effectiveness.

15.3 Motion in-betweening

Motion in-betweening, a crucial task in motion generation, involves creating intermediate frames between existing motion frames. Initially, this task was approached through keyframe interpolation using techniques like linear or spline interpolation [256, 257] and parameterization [258]. Advancements in this area included the integration of static models to enhance the interpolation process [224, 259].

The advent of deep learning has significantly impacted motion in-betweening. Recurrent Neural Networks (RNNs), in particular, have become a predominant tool in this domain. Harvey et al. [260] introduced the Recurrent Transition Network (RTN), a novel RNN architecture, to improve representation learning for 3D human motions. This model was further refined with the addition of time-to-arrival embedding [242]. Similarly, Tang et al. [261] utilized RNNs alongside conditional variational autoencoders (CVAE) for real-time motion transitions, offering enhanced responsiveness and accuracy.

Other approaches have explored convolutional autoencoders, as demonstrated in studies by Kaufmann et al. [262], Ruiz et al. [230], and Zhou et al. [263], for filling in missing frames and partial poses. The exploration of transformer architectures in motion in-betweening has also yielded promising results, with works such as Duan et al. [241], Kim et al. [243], and Oreshkin et al. [264] showcasing the capabilities of these models in handling complex motion sequences [242, 265].

Recently, diffusion models [266, 267] have emerged as a new approach in motion generation, achieving impressive performance in studies like Tevet et al. [268], Zhang et al. [269], and Kim et al. [270]. These models offer a novel paradigm for motion synthesis, though their real-time application is currently limited by computational intensity [271, 272].

15.4 Dance motion synthesis with music

Dance motion synthesis, a specialized branch of human motion synthesis, deals with the challenge of aligning dance movements with music. Early efforts, such as those by Lee et al. [17]

and others [18], focused on similarity-based matching to generate dance motions, but these methods often lacked the ability to produce diverse and lifelike dances.

The incorporation of deep learning, particularly LSTM networks, marked a significant advance in this field. Crnkovic-Friis et al. [21] were pioneers in using LSTM for dance motion synthesis, a trend that was further expanded in subsequent studies [206, 207, 273, 274]. These approaches enhanced the synchronization of dance with music, resulting in more fluid and coherent sequences.

Explorations with other neural architectures, such as GANs [22, 208, 275] and transformers [240, 209, 210, 276, 277], have further diversified the range of generated dance movements. GrooveNet [207] represents a notable contribution for its real-time music-driven synthesis. However, it faced limitations in user engagement due to its training scope and reliance on skeletal data.

15.5 3D dance dataset

The availability of publicly accessible 3D motion datasets, such as AMASS [278], Human3.6M [279], CMU motion capture [280], SFU motion capture [281], and Mixamo [282], has significantly contributed to research in human motion analysis. These datasets primarily focus on everyday human activities like walking, running, and sitting, offering valuable resources for a wide range of motion studies. However, their applicability to dance motion research is limited due to the distinct nature of dance movements compared to daily activities.

Recognizing this gap, several researchers have developed specialized dance datasets to support dance-related studies. These datasets are tailored to capture the nuances of dance movements, addressing the specific needs of dance motion synthesis research [22, 283, 208, 240, 210]. AIST++ [240], built upon the foundational AIST dataset [284], stands out as one of the most extensive 3D dance datasets, encompassing over 1000 dance sequences across 10 genres and amassing more than 5 hours of dance data. PhantomDance [210] further expands this collection, offering over 9 hours of dance content covering more than 13 genres.

However, a notable challenge in these datasets is the method of creation, often involving the extraction of 2D keypoints from videos and their subsequent conversion to 3D poses. While this

approach enables the generation of large-scale datasets, it can lead to compromises in pose accuracy and occasionally results in physically implausible motions. To address this, some studies have employed motion capture technology for direct 3D keypoint capture [207, 206, 285]. Despite providing more accurate motion data, these motion capture-based datasets have their limitations in terms of diversity and scope, as seen in GrooveNet [207] and other similar works.

Given the specific requirements of our research on improvisational dance, which necessitates a dataset encompassing a wide array of dance styles and genres, aligned with diverse music tracks and performed by numerous dancers, existing 3D dance datasets fall short. This gap underscores the need for the collection of a new, more comprehensive dataset that aligns with our research objectives in dance motion synthesis.

Chapter 16: Experience Design

In this chapter, we detail the foundational principles of our proposed dance experience. In envisioning a generative dance system, our goals are aligned with the principles of improvisational dance, while striving to create a highly engaging and interactive user experience. These goals include:

- **Enhanced User Interactivity:** The system should allow users to influence the dance animations by controlling the music, encouraging active engagement.
- **Real-Time Spontaneity:** The ideal system must react instantaneously to user interactions, requiring real-time music analysis and dynamic dance synthesis.
- **Element of Surprise:** Incorporating unpredictability and uniqueness in each dance sequence to reflect the essence of improvisation.
- **Physical Realism:** Generated dance motions should be realistic and fluid, free from artifacts typical to deep learning based systems.
- **Natural Avatar Behavior:** In the absence of music, avatars should exhibit natural, idle behaviors to enhance realism.
- **Ease of Extensibility:** The system should be easily updated with new dance routines without retraining.
- **Dance Authoring:** The system should allow artists to inject specific dance routines to particular musical segments. Ex: Macarena dance, Gangnam Style dance.
- **Avatar Personalization:** The system should support a user's personal 3D avatar from various platforms, deepening user connection and immersion in the virtual dance environment.

- **Animation Personalization:** Avatars come in various sizes and shapes. The synthesized dance should preserve the motion semantics between various avatars while minimizing artifacts.

Chapter 17: Method

In this chapter, we present `Dance Synthesizer`, our proposed approach that takes user's music of selection as input and generates music-conditioned 3D improvisational dance.

17.1 Overview

Currently, a prevalent class of methods is to generate dance frames by sampling from a learned dance motion manifold conditioned on the entirety of the input music [207, 22, 208, 211, 209]. However, these methods heavily depend on the quality of learned manifold which requires a substantial amount of data. Even then, generated dances often lack of realism and tend to be repetitive. In addition, none of these methods can perform in real-time, reactive to user interactions. To overcome these limitations, our `Dance Synthesizer` is divided into two distinct components: music analysis and dance synthesis. Each of these components is discussed in details below.

17.2 Music analysis

Upon user selection, the chosen music track is fed into the Music Analysis component for real-time analysis. We employ a moving window approach with a 3-second window size and 0.3-second increments. Within each window, the system analyzes the music across three key dimensions: energy, tempo, and musical type. These dimensions are explained in detail below.

Music energy. We define the music energy as the loudness of the music signal. According to Steven's power law [286], the loudness of a music signal is its intensity raised to the power of 0.67. Please note that this is a rough estimation of the music's energy, as it is computationally inexpensive and sufficient for our needs.

Tempo estimation. We utilize BeatNet [287], a state-of-the-art method, for real-time tempo estimation. Specifically, we apply BeatNet's "offline mode" on each 3-second music window,

updated every 0.3 seconds. Finally, we employ a filtering process to minimize abrupt variations in the final tempo estimation results.

Music type estimation. We categorize music into three types: idling (absence of music), slow-paced, and fast-paced music, similar to [285]. To identify these three music types in real-time, we utilize YAMNet [288], a light-weight network known for its great performance on sound event classification. We train the network on a dataset created using Jingle Punks [289], with labels corresponding to the three defined categories: 0 for idling, 1 for slow-paced, and 2 for fast-paced music. Furthermore, we integrated YAMNet's original version on sound event classification to help distinguish between music, background noise, and human speech. With these two versions of YAMNet together, our system is able to accurately classify both music existence and music types in real-time.

17.3 Dance synthesis

Subsequently, the Dance Synthesis component of our system generates dance movements in real-time, aligning with the input music's characteristics determined by the Music Analysis component. Unlike the limitations of existing methods discussed in Section 17.1, our approach instead takes inspiration from the classic graph-based methodologies.

17.3.1 Inspiration: classic graph-based approaches.

Introduced in Section 15.2, this class of methods synthesize motion sequence by finding an optimal path within a pre-constructed motion graph. In such a graph, nodes represent individual motion segments from a database, and edges denote the probabilities of transitioning between these segments. The calculation of these probabilities could involve, for instance, assessing the similarity between ending motion segments of the two connected nodes.

However, a limitation of this approach lies in the potential sparsity of the motion graph. A sparse graph, with insufficient amount of edges connecting motion segments, can lead to repetitive dance sequences, contrary to our goal of generating diverse and spontaneous dances. Additionally, no music information can be incorporated in these methods. Therefore, to overcome these issues, our Dance Synthesis component employs a hybrid approach, combining the concept of a classic motion

graph with a learning-based method.

17.3.2 Our hybrid approach.

Aligning with classic graph-based methods, our approach synthesizes dance movements by connecting various motion segments. Traditional methods typically establish connections based on criteria like similarity scores, derived from joint position, orientation, and timing. In contrast, our system dynamically creates these connections using real-time music data, as detailed in Section 17.2.

When a user starts a music track, our Dance Synthesis component quickly analyzes the track's tempo, rhythm, and energy level. Based on this analysis, the system connects the current idle motion to a database segment that best matches the music's characteristics. Unlike traditional approaches, which focus on mechanical fit, our method ensures transitions that are not only smooth but also musically synchronized, enhancing the overall coherence of the dance.

The result is a dense motion graph where each node (a dance segment) connects to many others based on various musical contexts. The richness of this graph provides a vast array of potential paths for dance synthesis, allowing the system to produce non-repetitive, musically aligned dance sequences with nearly limitless possibilities.

By transforming the traditional motion graph into a dynamic network of connections, our approach effectively turns the synthesis process into a sophisticated motion in-betweening task. This method focuses on creating fluid, expressive transitions between motion segments, ensuring that each dance sequence is uniquely tailored to the music.

The outcome is a significant enhancement in the realism and variability of dance performances, greatly improving user interaction. Each sequence reflects both the user's music choice and the system's ability to craft seamless, compelling transitions, delivering a uniquely engaging dance experience that pushes the boundaries of traditional motion synthesis.

17.3.3 Dataset

Since our hybrid approach adapts a modified version of the motion graph, it is necessary to create a comprehensive and diverse dataset for the approach to run effectively. We engaged professional

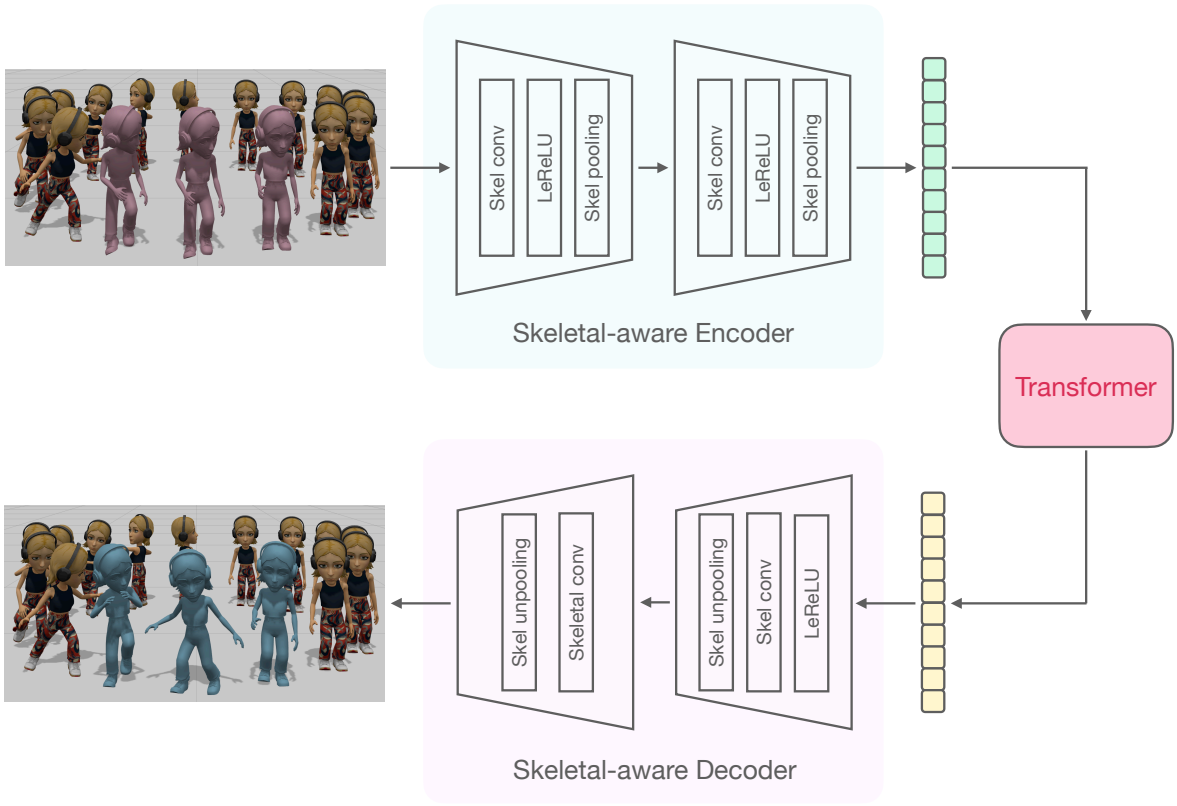


Figure 17.1: **Our dance transition generator network.** The transition generator, $\mathcal{G}_T$, has three components: (a) a skeletal-aware encoder using skeletal operators; (b) a transformer subnetwork that operates in the latent space generated by the encoder; and (c) a skeletal-aware decoder that projects latent vectors back to 3D skeletons.

dancers to perform improvisational dances to an extensive selection of over 200 songs. These tracks were meticulously chosen from a variety of genres, including hip-hop, country, folk, jazz, and disco, to ensure a broad representation of musical styles. The tempos of these songs varied from 80 BPM to 160 BPM, to accommodate a diverse set of dance rhythms and patterns.

In order to capture the dance movements with high fidelity, we equipped the dancers with the Smartsuit Pro II from Rokoko [290]. This motion capture suit allowed us to record the dances directly in 3D, ensuring the accuracy and quality of the captured motions. The result of this process is a comprehensive collection of over 6 hours of dance movements, stored in the Mixamo format [282].

A crucial aspect of our dataset is its versatility to accommodate a wide range of body shapes,

from skinny to obese. Leveraging retargeting techniques inspired by [291], all captured dance movements to six distinct body shapes. This flexibility ensures seamless integration of different Bitmojis into our system.

17.3.4 Approach details.

Here we introduce the implementation details of our hybrid approach, specifically, the transition generation network which we denote as $\mathcal{G}_T$.

$\mathcal{G}_T$ vs. typical motion in-betweening. Typical motion in-betweening methods such as [242, 243], concentrate on generating transitions from an initial motion sequence to a designated *target pose*. In this case, these methods only need to ensure the smoothness between the initial motion sequence to the transition, as long as the transition ends in the target pose. In contrast, $\mathcal{G}_T$ aims to ensure smooth motion continuity not only from the initial sequence to the transition movements but also from the transition to subsequent dance motions which could originate from a different dance track with potentially contrasting physical movements.

Network input. The input for the proposed $\mathcal{G}_T$ consists of a sequence of motion frames divided into three parts: (1) an initial dance segment from the source track, (2) placeholders for intermediate transition frames, and (3) a dance segment from the target track, which may differ from the source. We set the length of the input sequence as $L = M_1 + T + M_2 = 160$ frames, with $M_1 = M_2 = 65$ frames each for the two dance segments, and $T = 30$ frames for the transition, equivalent to one second of dance motion at the system's frame rate of 30 FPS. These 30 placeholder frames are initially generated using Linear Interpolation (LERP) for the global root position and Spherical Linear Interpolation (SLERP) for each joint's rotation between the last frame of the initial segment and the first frame of the subsequent one. $\mathcal{G}_T$ then learns to refine these placeholder frames, ensuring a seamless and natural connection between the dance segments on each end.

Network architecture. The network architecture of $\mathcal{G}_T$ draws inspiration from the framework of a denoising autoencoder. In this configuration, the network is trained to accurately predict a ground-truth motion sequence from an input sequence that has been altered or perturbed. As depicted

in Figure 17.1, our network features a skeletal-aware encoder consisting of two skeletal blocks [292]. Each block is composed of a skeletal convolution, an activation function, and skeletal pooling, which collectively map each input motion frame into a corresponding latent vector. The skeletal-aware decoder, mirroring the encoder, also comprises two skeletal blocks. These blocks function to project the latent vectors back into the domain of 3D skeletal motions, thereby reconstructing the motion sequence.

A key distinction of our network model from the conventional denoising autoencoders is the integration of a transformer subnetwork positioned between the encoder and decoder. This transformer subnetwork, based on the structure from [243], has been adapted to accept inputs from our skeletal-aware encoder and produce outputs compatible with our skeletal-aware decoder. The inclusion of this transformer subnetwork is critical in our architectural design, as it is essential in generating an intermediary motion sequence that is both engaging and relevant to the ongoing dance synthesis. This is confirmed by the results of the ablation study discussed in Section 19.1.

17.3.5 Data and training.

Here we discuss the data and representation used to train $\mathcal{G}_T$ and its training objectives.

Data. To train $\mathcal{G}_T$, we randomly selected approximately one-tenth of the total number of dance tracks from the comprehensive dataset introduced in Section 17.3.3.

Data representation. In our system, the 3D human skeleton motions is represented as two components: a static component and a dynamic component. The static component, denoted as $\mathrm{S} \in \mathbb{R}^{J \times S}$, encapsulates the armature information, where J represents the number of armatures and $S = 3$ corresponds to the 3D spatial coordinates. The dynamic component, symbolized as $\mathrm{Q} \in \mathbb{R}^{L \times J \times Q}$, captures motion dynamics, with L indicating the length of the motion sequence and $Q = 4$ reflecting the use of Quaternions for rotation representation. Additionally, the root joint is represented as $\mathrm{R} \in \mathbb{R}^{L \times (S+Q)}$, distinct from the J armatures. It comprises a sequence of global translations and rotations, crucial for maintaining the integrity and coordination of the entire skeletal motion.

This approach to data representation lays the groundwork for implementing *skeleton-aware* operations in $\mathcal{G}_T$. These operations, derived from the graph-based structural representation of the skeleton, consider key aspects such as bone hierarchy and joint adjacency. Such consideration is vital for ensuring the physical plausibility of generated motions. For a comprehensive understanding of these skeleton-aware operations and their implications in motion synthesis, we direct readers to the foundational work by Aberman et al. [292], which delves into the intricacies of skeletal convolution and skeletal pooling.

Training objectives. Let E, T, and D denote the encoder, transformer, and decoder components of $\mathcal{G}_T$, respectively. We optimize the following loss function:

$$\begin{aligned}\mathcal{L} = \mathbb{E}_{(\mathrm{S},\mathrm{Q}_i)\sim\mathcal{M}}\Big[\Big\|\Big(D(T(E(\mathrm{S},\hat{\mathrm{Q}}_i)),\mathrm{S}\Big)-\mathrm{Q}_i\Big\|_2\Big] \\ + \mathbb{E}_{(\mathrm{S},\mathrm{Q}_i)\sim\mathcal{M}}\Big[\Big\|\mathrm{FK}\Big(D(T(E(\mathrm{S},\hat{\mathrm{Q}}_i)),\mathrm{S}\Big)-\mathrm{P}_i\Big\|_2\Big],\end{aligned} \tag{17.1}$$

where $\mathcal{L}$ is a standard reconstruction loss over the joint rotations and joint positions. Each frame of motion $i \in \mathcal{M}$ is represented by $(\mathrm{S},\mathrm{Q}_i)$, the pair of skeleton offset and joint rotation. $\hat{\mathrm{Q}}_i$ is the SLERP rotation input. FK is a forward kinematic operator that, given skeleton offset and joint rotations, returns the joint positions. $\mathrm{P}_i = \mathrm{FK}(\mathrm{S},\mathrm{Q}_i)$ are the joint positions of the ground truth dance sequence.

Chapter 18: Implementation

Our system's design and the methodological approach we have adopted are versatile, allowing for implementation as either a standalone unit or within a larger online framework. In this chapter, we offer a reference implementation of our system as a web application specifically designed for a standalone kiosk shown in Figure 18.1.

18.1 Hardware design

Our kiosk setup is designed to deliver a realistic and immersive experience, featuring a 65-inch portrait monitor for displaying life-size dancing Bitmoji characters. This setup caters to a wide range of music sources, allowing users to play their choice of music from smartphones, tablets, MP3 players, or even online web players. An integral part of this setup is a QR code scanner [293], which lets users bring their personalized Bitmoji avatars into the dance animation system, adding a layer of personalization to the experience. We leverage The computational processing is handled by an Apple M2 Ultra Mac Studio, which synthesizes the 3D dance animations in realtime that are synchronized with the music.

18.2 Software design

Our system's versatility is enhanced by its implementation as a web application. The software topology diagram, as shown in Figure 18.3, provides a detailed overview of our system's architecture. Key components like `Music Analyzer` and `QR Parser` are implemented server-side, handling the backend processes. On the client side, we have developed the `Dance Synthesizer` engine as a JavaScript application, complemented by a WebGL presentation layer for visually rendering the animations.

The `Music Analyzer` processes the incoming music stream using a moving window

Figure 18.1: **DanceCraft kiosk:** The kiosk features an iPad for user to play music. The avatar dances in response to the music on a life-sized portrait monitor. Users can cast their personal 3D Bitmoji by presenting a QR code to the QR code scanner.

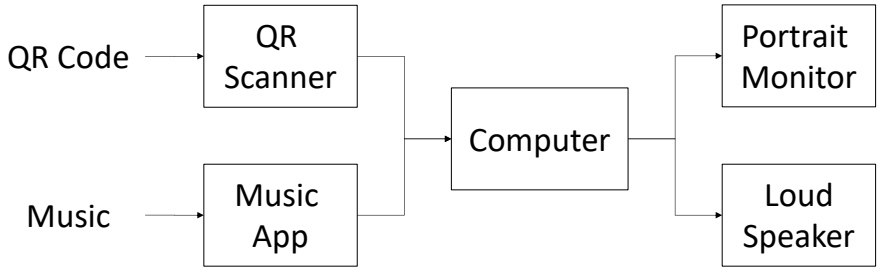


Figure 18.2: **Hardware setup topology diagram.**

approach. Based on empirical analysis, we have determined an appropriate window size of 3 seconds and a moving increment of 0.3 seconds. Within each window, the `Music Analyzer` extracts

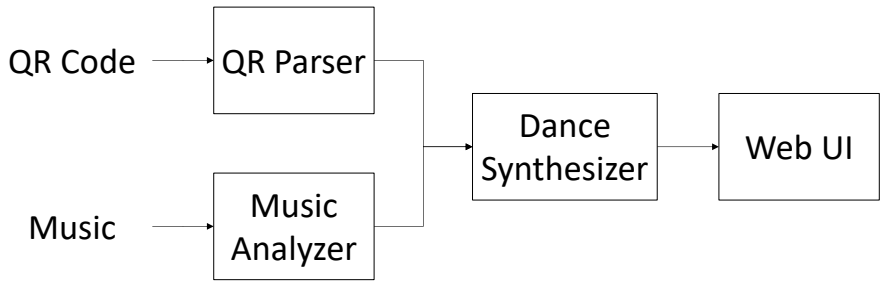


Figure 18.3: **High level software architecture.**

key musical characteristics, as detailed in Section 17.2. This process allows for the extraction of pertinent musical features such as genre, tempo, and energy. The extracted data is then relayed in real-time to the `Dance Synthesizer` engine.

For users with Bitmoji avatars, the integration is seamless. They can retrieve a QR code encoding the URL of their Bitmoji through Snapchat. Scanning this code with the kiosk's QR code scanner enables their personalized avatar to be projected into the animation system. It is important to highlight that all Bitmojis from different users share a common skeletal structure. This unified skeleton enables us to seamlessly switch between different Bitmojis in real-time according to the user's request. However, as stated in 17.3.3, Bitmojis can have different body shapes and sizes – from skinny to obese mesh. The metadata embedded in the retrieved Bitmoji model aids the dance synthesizer in switching to the appropriate set of dance animations to prevent interpenetration.

Chapter 19: Evaluation

This chapter presents the quantitative evaluations of our method, comparisons to baseline and prior works, and ablation studies. We also conduct user studies to assess the proposed system qualitatively.

19.1 Quantitative evaluations

19.1.1 Dance track coverage

A critical aspect of evaluating the performance of our proposed system is the assessment of how comprehensively individual dance tracks within our database are utilized. An ideal functioning of the system, in line with our dense motion graph approach described in Section 17.3, would involve traversing the full range of all dance tracks in our collection.

To quantitatively evaluate this aspect, we conducted a series of tests where the system was fed input songs across a spectrum of tempos, ranging from 80 BPM to 160 BPM, in increments of 5 BPM. For each tempo level, we initiated 1000 transitions and observed the extent of coverage for each dance track. The coverage results for four dance tracks, selected at random, are illustrated in Figure 19.1. These findings demonstrate that our system consistently covers the entire span of each selected dance track, showcasing its ability to effectively utilize the entire breadth of the dance database. This extensive coverage is indicative of the system's capacity to provide a diverse and comprehensive dance experience, adapting to a wide range of musical tempos and styles.

19.1.2 Motion in-betweening methods comparison

To assess the efficacy of our `Dance Synthesizer`, particularly $\mathcal{G}_T$, the real-time generation of 30-frame motion in-betweening, we conducted a comprehensive evaluation using the LAFAN1 dataset [242]. Our evaluation employed three metrics: i) L2 distances of global positions (L2P)[242], ii) L2 distances of global rotations (L2Q)[242], and iii) Normalized Power Spectrum Similarity

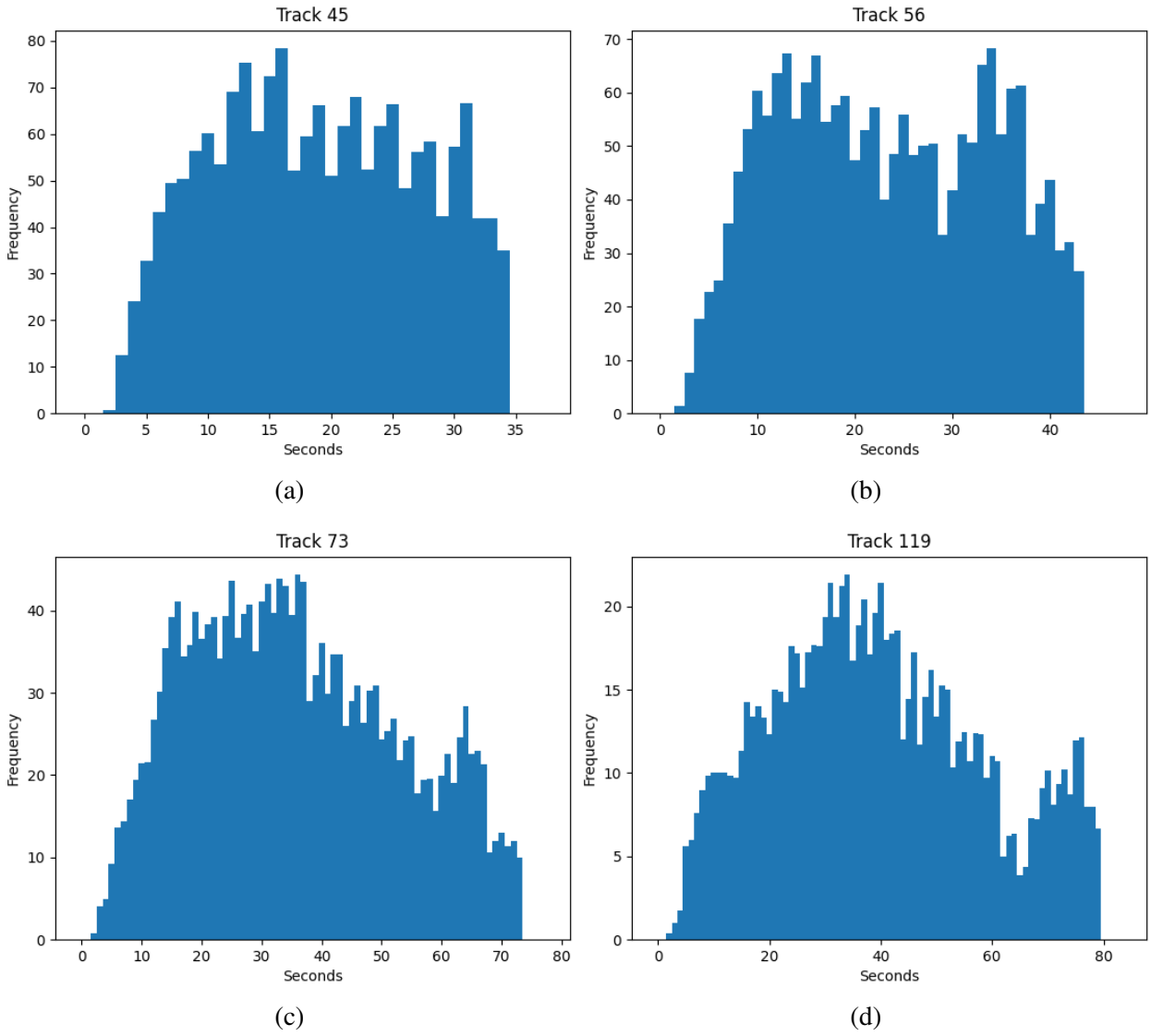


Figure 19.1: **Dance track coverage histogram.** This figure displays the coverage histograms for four dance tracks, randomly selected from our database. The histograms visually represent the extent to which each track is utilized by our proposed system. It is evident from the results that our system comprehensively covers the full length of every selected dance track, demonstrating its capability to exploit the entire range of the database for dance synthesis.

(NPSS) [294].

Our $\mathcal{G}_T$ was compared against various baselines and existing methods. The zero-velocity baseline approach involves replacing the intervening frames with the last frame of the source motion.

Table 19.1: **Comparison on LAFAN1 for motion in-between task (30 frames).** The best performance in each category is indicated in bold and the second-best is highlighted in cyan.

Length = 30	L2P ↓	L2Q ↓	NPSS ↓
Zero-Velocity	6.60	1.51	0.2318
Interpolation	2.32	0.98	0.2013
ERD-QV [242]	1.28	0.69	0.1328
SSMCT [241]	1.10	0.61	0.1222
CMIB [243]	1.19	0.59	0.1415
Δ-Interp [264]	**1.00**	0.57	0.1217
Ours	1.04	**0.54**	**0.1213**

The interpolation baseline utilizes linear interpolation (LERP on the global root position and SLERP on the rotation of each joint) between the final frame of the source motion and the initial frame of the target motion. Additionally, we compared our method with several prior techniques, including ERD-QV [242], SSMCT [241], CMIB [243], and Δ-Interp [264].

The results of this quantitative evaluation, presented in Table 19.1, demonstrate the robustness of our approach. Our method not only surpasses previous state-of-the-art (STOA) methods like ERD-QV, SSMCT, and CMIB in performance but also competes favorably with concurrent works, namely Δ-Interp. This indicates the effectiveness of our $\mathcal{G}_T$ in accurately and realistically generating intermediate dance frames in real-time, a crucial capability for the system's overall functionality.

19.1.3 Ablation study

To validate the impact of the transformer component in our $\mathcal{G}_T$, we conducted an ablation study. This involved removing the transformer component from our model for comparative analysis. The modified version of our model without the transformer is denoted as "Ours w/o Trans", and we juxtaposed its performance against our complete model, denoted as "Ours".

In the absence of the transformer, our model adopts a configuration akin to a denoising autoencoder, aligning with the structural outline provided in Section 17.3. While this simplified denoising autoencoder version demonstrates ease of training, as evidenced by a lower loss value

Table 19.2: **Ablation study.** We compared the performance of our proposed $\mathcal{G}_T$ with and without the transformer component. The best score under each metric is emphasized in bold.

Length = 30	L2P ↓	L2Q ↓	NPSS ↓
Ours w/o Trans	1.31	0.73	0.1337
Ours	**1.04**	**0.54**	**0.1213**

during the training phase, it falls short in effectively generalizing to infer dance motions. This limitation is evident when compared with the full model, incorporating the transformer. The comparative results, detailed in Table 19.2, underscore a significant performance discrepancy between the two versions. The findings from this ablation study highlight the critical role of the transformer component in enhancing the system's capability to accurately and dynamically generate dance motions, thereby validating its inclusion in our $\mathcal{G}_T$.

19.2 Qualitative user studies

Quantitative evaluations highlight our $\mathcal{G}_T$'s in-betweening abilities, but they only partly represent the goal of our full system, `Dance Synthesizer`, which is to generate real-time, music-reactive improvisational dances. GrooveNet [207], the most closely related work, also focuses on real-time dance generation but struggles in generalizing beyond its training data. Hence, for a more effective comparison, we included Bailando++ [277], an autoregressive method known for its qualitative strength in generating 3D dance movements based on music.

19.2.1 User studies

We conducted three sets of user studies to evaluate our system from various perspectives, including quality of dance, user experience, and overall preferences of the participants.

Participants. We recruited 32 participants, comprising 19 males and 13 females, aged between 18 and 37. None of the participants had previous experience with 3D dance systems.

User Study 1: Quality of dance.

(a) Bailando++ output (b) Retargeted Bitmoji

Figure 19.2: **Data preparation for user studies.** We retargeted the output from Bailando++, originally in SMPL skeleton format, to the Bitmoji format. We then associated a random Bitmoji character with the skeleton, ensuring a fair and consistent comparison between the outputs of Bailando++ and our proposed system.

Data preparation. Bailando++ is *not* a real-time method, as it conditions dance generation on the entirety of the input music. To ensure a fair comparison in our user studies, we prepared data as follows:

1. Since Bailando++ is trained on AIST++ dataset [240], we randomly selected 20 music clips from its test set to generate dance movements for each.

2. Bailando++ generates dances in SMPL format [295]. We retargeted the outputs to the Bitmoji skeleton format, and assigned random Bitmoji characters to the skeletons. The resulting dance sequences were then converted into videos, as shown in Figure 19.2a and Figure 19.2b.

3. Using the same 20 music clips, our system generated corresponding dance sequences in real-time. These sequences were recorded into videos for comparison.

Procedure and results. To compare dance quality, participants engaged in a blind test viewing 20 videos from either Bailando++ or our system. They rated each video on a scale from 1 to 7 across four

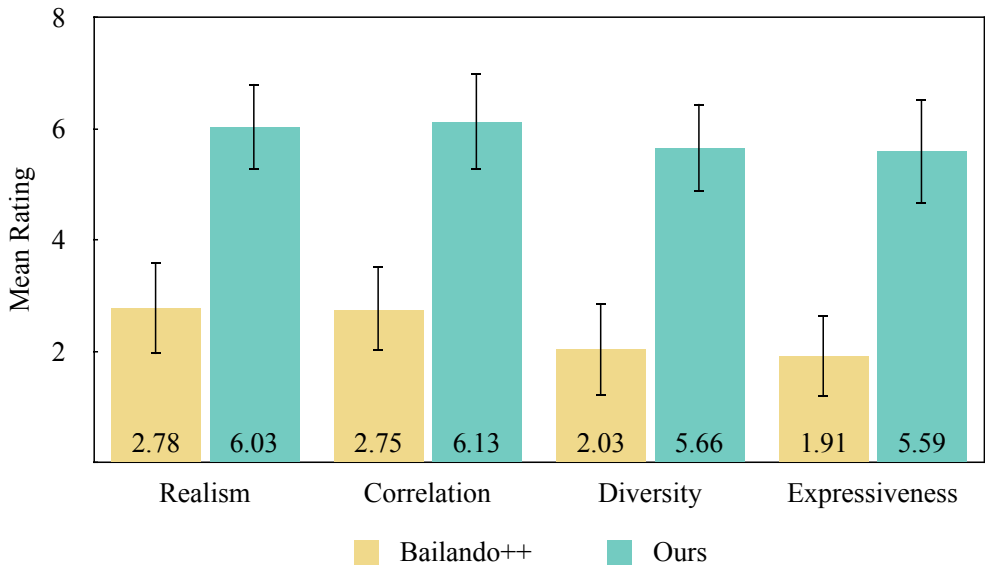


Figure 19.3: **Quality of dance comparison.** We conducted a blind comparison between dances generated by Bailando++ and our method. The results reveal a strong preference among participants for our system's dance quality.

dimensions: dance realism, music-dance correlation, dance diversity, and movement expressiveness. Participants then repeated the procedure with the other system. To counter order bias, the study sequence was counterbalanced. Figure 19.3 presents the average user ratings, indicating a marked preference for our proposed system.

User Study 2: User experience.

Procedure. Next, we focused on the participants' interaction with the two systems. For our proposed system, participants engaged with it as detailed in Chapter 16, enjoying the freedom to select their preferred music, pause, and switch tracks at their discretion. For Bailando++, the interaction was facilitated through the same iPad kiosk. Participants chose their music, then used an all-in-one script for processing. This script managed the music download, processing, running Bailando++, retargeting the SMPL skeleton to a Bitmoji skeleton, applying a random Bitmoji character, and finally exporting the dance video. The duration of this process varied, generally taking around 5 minutes or longer, depending on the music length.

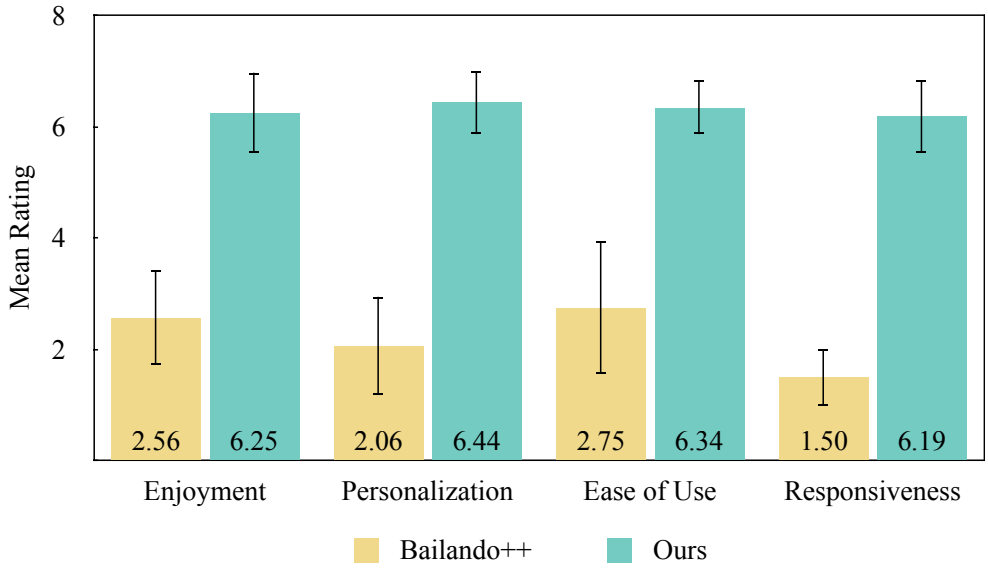


Figure 19.4: **User experience comparison.** Participants demonstrated a clear preference in this blind test comparing the user experience of Bailando++ against our method. The results overwhelmingly indicate that the experience provided by our system is favored, highlighting its effectiveness in engaging and satisfying users.

Participants were then asked to rate their experience on four aspects: enjoyment, perceived personal engagement, ease of use, and system responsiveness. They provided their ratings on a scale from 1 (extremely disagree) to 7 (extremely agree). This feedback offered valuable insights into not only the quality of the dance but also its synchrony with the selected music. Following this, participants repeated the process with the alternate system.

Result. In line with the previous study, the identity of the systems was not disclosed to the participants. A blind comparison was carried out in a counterbalanced order to minimize bias. The mean user ratings for both systems, as depicted in Figure 19.4, clearly demonstrate a strong preference for our proposed system. This preference highlights our system's superiority in terms of user experience, encompassing aspects of enjoyment, engagement, usability, and responsiveness.

User Study 3: Overall preference.

Procedure. Upon completing the blind interaction with both systems, participants were asked to

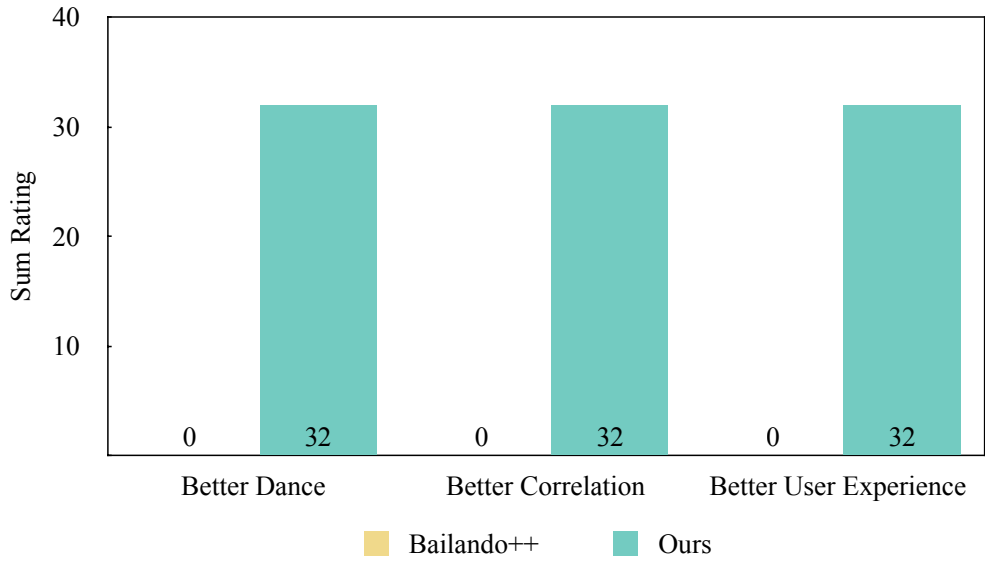


Figure 19.5: **Overall preference of participants.** In the concluding phase of our user studies, participants were asked to select their overall preferred system. The results show a decisive preference, with all of participants favoring the experience and capabilities of our system.

express their overall preferences through three targeted questions. These queries were designed to ascertain their favored system in terms of overall dance quality, music-dance correlation, and user experience. The categories were chosen to encapsulate the critical aspects of the systems that contribute to user satisfaction and engagement.

Result. The results of this final evaluative step, as shown in Figure 19.5, reveal the participants' collective preferences. Despite not being informed about the identities of the systems, the participants unanimously favored our proposed system across all the categories. This unanimous preference underscores the effectiveness of our system in delivering a superior user experience, characterized by a harmonious blend of dance quality and music-dance synchronization. It is a testament to the success of our system in not only meeting but exceeding user expectations in the realms of interactive dance synthesis and user engagement.

19.2.2 Positive feedback

The participants' response to our proposed system was overwhelmingly positive. A notable observation was the universal expression of enjoyment, as evidenced by the smiles on the faces of all participants while interacting with the system. A significant majority, 27 out of 32 participants, expressed astonishment and delight at seeing their Bitmoji avatars dance in sync with their chosen music on the large portrait TV.

The immersive experience of our system also inspired active participation, with 23 participants joining in by mirroring the dance movements of their virtual counterparts. This level of engagement reflects the system's capacity to captivate and motivate users to be part of the dance experience actively.

Additionally, 12 participants voiced their excitement at the prospect of integrating our system into VR dance parties, envisioning a new dimension of entertainment in virtual reality settings. Noteworthy individual reactions included Participant #8, who was so captivated by the experience that they returned with their daughter for a second interaction. Similarly, Participant #14 expressed fascination with her Bitmoji's improvisational dance abilities, pondering the system's potential to adapt to a full-scale Broadway musical.

These anecdotes, alongside the overall positive ratings, underscore the system's practicality, usability, and appeal across a diverse user base. The feedback underscores the system's potential for varied applications, from personal entertainment to larger social gatherings and virtual events.

19.2.3 User suggestions

Feedback from participants also included constructive suggestions for enhancing our system. One prominent suggestion, echoed by several participants, was the incorporation of a broader range of dance styles, particularly those reflecting diverse cultural traditions. Specific examples mentioned included the traditional Chinese ribbon dance, the Japanese Awa Odori, and Bharatanatyam from India. Such additions would undoubtedly enrich the system's repertoire and appeal to a more global audience.

Another frequent recommendation was to extend the system's compatibility to mobile devices, thereby increasing its accessibility and convenience. This modification would allow users to interact with the system through their smartphones, potentially broadening its user base.

Over half of the participants also saw potential in integrating our system with AR/VR headsets. They envisioned it as a valuable enhancement for immersive virtual environments, suggesting that it could significantly enrich the AR/VR experience by providing interactive, music-responsive dance content.

A particularly intriguing idea came from Participant #27, who envisioned transforming his 3D Bitmoji into a physical gadget. This gadget, inspired by our system, would accompany him and react to music in the real world, akin to the virtual experience provided by our system. This suggestion points towards innovative, tangible applications of our technology, extending its utility beyond the digital realm.

Chapter 20: Summary

20.1 Limitations and future work

Our emphasis on improvisational dances comes at the cost of choreographic accuracy and semantic nuances, which might be perceived as a drawback by trained dancers. Another limitation lies in our current dance routine retrieval system, which operates on a basic lookup table approach based on music genre, tempo, and energy. Future enhancements aim to evolve this into a more sophisticated two-tower embedding model [296] with predictive modeling, capable of not only retrieving semantically relevant dances but also ensuring synchronization with musical beats. Currently, another drawback of our system is that it is not computationally efficient on low-end computers. We would like optimize the method so that it can even run on mobile phones.

20.2 Conclusion

In this work, we have introduced a production-ready, real-time system adept at generating realistic, expressive, and captivating improvisational dances in response to music. Our approach overcomes the limitations of both data-driven graph-based and deep learning approaches by implementing a novel hybrid method. We also unveiled a comprehensive dance dataset encompassing a diverse array of musical genres, tempos, and energy levels. Our system emphasizes user interactivity and personalization, creating a deeply engaging experience.

Significantly, user study results show a marked preference for our system's generated dance motions among untrained users. Finally, our implementation of the system with web technologies enables versatile deployment options. We can set up the dance generator as a standalone kiosk or integrate it into larger virtual concert environments.

Conclusion

Chapter 21: Summary and Future Outlook

In this thesis, we explored two distinct yet interconnected types of audio signals: speech and music. Our research spanned three key areas — speech denoising, speech dereverberation, and music-dance generation — each informed by classical techniques such as spectral subtraction, Wiener deconvolution, and graph-based human motion synthesis. The methods we developed were specifically designed to produce high-quality results, whether in the form of denoised or dereverberated audio, or highly expressive 3D improvisational dances, ensuring their suitability for both human consumption and subsequent applications.

A unifying theme across all three methods is their emphasis on human-centric interactivity, which also significantly reduces the reliance on massive datasets. This interactivity is expressed in various forms: passively, by leveraging intrinsic structural elements of speech signals like silent intervals; actively, through a simple personalization step that enhances dereverberation; and proactively, by generating real-time dance movements that respond dynamically to music selection and user-inputted 3D avatars. By focusing on the human-centric features, our methods bypass the need for extensive data by effectively utilizing the inherent properties of the input signals. The effectiveness of these methods is highlighted by their state-of-the-art performance, validated through both quantitative evaluations and blind user studies.

In the introductory chapter of this thesis (Chapter 1), we posed several critical questions to investigate the potential of machine learning in the realm of audio signal processing: "Can we draw inspiration from classical methods, rather than solely pursuing a data-centric trend?" and "Can we leverage intrinsic properties of data to reduce the reliance on massive datasets, thereby adopting a more human-centric approach?" These questions, which encompass a broad spectrum of contemporary audio signal processing research, have been the guiding force behind the studies presented in this thesis. The findings discussed throughout the work affirmatively address these

inquiries, illustrating that the integration of classical inspirations with human-centric principles can indeed foster substantial advancements in the field of audio signal processing.

21.1 Future outlook

21.1.1 Extensions of speech enhancement

Exploiting multi-modal data. Incorporating multi-modal data into speech processing can greatly enhance noise suppression and speech clarity, particularly in complex environments. Visual cues such as lip movements and facial expressions provide additional context, improving the separation of speech from background noise by aligning the timing and emotional tone of the speech signal [297]. Text-based models offer further improvements by using contextual information to resolve ambiguities in degraded audio, enhancing intelligibility in noisy conditions [298].

For speech dereverberation, integrating beamforming techniques with traditional methods like Wiener deconvolution can focus on direct sound paths and reduce reflections, leading to better results in echo-prone environments [299]. Additionally, material recognition technologies help refine room impulse response (RIR) estimations, improving dereverberation by accounting for how different surfaces interact with sound [300]. LiDAR and Sonar technologies offer real-time environmental sensing, allowing speech processing algorithms to adapt dynamically to changes in room geometry and surface properties, resulting in more effective enhancement across various scenarios. By leveraging these additional modalities, the reliance on audio-only data is reduced, enabling a more robust and versatile approach to speech enhancement and dereverberation.

Speech enhancement as a prior. Utilizing enhanced speech as a foundational element in advanced speech processing tasks such as automatic speech recognition (ASR), speaker identification, and emotion recognition could significantly improve the accuracy and effectiveness of these systems. Clean, noise-free speech signals provide an optimal input for these downstream tasks, enhancing their overall performance and robustness in diverse acoustic environments [301]. Models like SEGAN [39] and Wavenet [71] have already shown their effectiveness in producing high-quality speaker embeddings, which can be crucial for tasks like speaker identification and separation.

Expanding on this, our proposed speech enhancement model [23] offers potential beyond just noise reduction. The embeddings generated by our model could be leveraged in speaker separation tasks, where distinguishing between different voices in a noisy environment is critical. Additionally, there is potential to explore its application in emerging areas like speech camouflaging [302], where the goal is to mask speech effectively in privacy-sensitive situations. This research direction not only broadens the applicability of our model but also opens up new possibilities for enhancing the privacy and security of speech-based interactions.

Speech enhancement in the wild. Current speech enhancement systems, while effective in controlled environments, often struggle to maintain their performance in more complex and unpredictable real-world settings, such as bustling streets, crowded public spaces, or highly reverberant environments like large auditoriums. Existing dereverberation techniques, including those developed in this thesis, typically demonstrate high efficacy in smaller, well-controlled environments where the reverberation time (T_{60}) is less than one second. However, they encounter significant limitations when applied to larger or more acoustically challenging spaces, such as concert halls, churches, or the immersive sound environment of the Sonic Sphere [205], where T_{60} can exceed five seconds. Therefore, enhancing the adaptability and robustness of speech enhancement systems to function effectively across a broader range of acoustic environments remains a critical area for future research.

21.1.2 Real-time processing

Real-time speech enhancement. The growing need for real-time speech processing applications, particularly in contexts such as online communication, presents both challenges and opportunities for innovation. Our proposed modification to the sliding window approach [24] successfully enables real-time operation within denoising systems that incorporate convolutional layers. However, the current method's dependency on specific model architectures limits its broader applicability. Future research should aim to develop more versatile real-time speech enhancement techniques that transcend architectural constraints, allowing for seamless integration across various types of models, including those that are not traditionally optimized for real-time performance.

Moreover, real-time speech processing faces a unique challenge with reverberation, which extends over a longer temporal span compared to typical additive noise. Addressing reverberation in real-time is particularly difficult due to the need to account for prolonged echo effects that can significantly distort speech clarity. Developing effective real-time speech dereverberation solutions would represent a major breakthrough, offering substantial improvements in communication clarity in reverberant environments, such as large halls or enclosed spaces with reflective surfaces.

For a more comprehensive discussion on these topics, please refer to Section 7.1 and Chapter 13.

Real-time 3D dance generation. In **Part III**, we introduced a method capable of real-time 3D dance generation by synthesizing transitions between dance movements. While this approach is effective for creating smooth and dynamic transitions, a significant challenge remains in achieving expressive 3D dance generation on a frame-by-frame basis in real time. Overcoming this challenge would allow for the incorporation of more nuanced musical properties, thereby enhancing the alignment of dance movements and styles with a wider range of musical genres and creating a more immersive experience.

A key limitation in achieving real-time frame-by-frame dance generation lies in the current state of real-time music information retrieval (MIR). Our method currently focuses on extracting and utilizing simpler musical features, such as tempo and energy levels, to guide dance generation. However, more intricate aspects of music, such as real-time onset detection, beat tracking, and bar detection, are still challenging to implement effectively in real-time scenarios. Advancing real-time deep MIR technology is a promising research direction that could significantly improve the synchrony between dance and music in live applications. By accurately capturing and responding to complex musical cues, future systems could generate dance sequences that are not only technically proficient but also emotionally resonant with the music.

Recent advancements in diffusion-based motion synthesis models [268, 269, 270, 271, 272] have demonstrated the capability to generate highly expressive and realistic motion sequences when trained on large datasets. While these models can produce impressive results, they often face challenges when tasked with generating extended sequences, such as hours of continuous

dance movements, which our method [26] handles more effectively. Additionally, diffusion-based models tend to be computationally intensive, requiring significant processing power, whereas our approach is optimized to run efficiently on lower-end devices, making it more accessible for broader applications.

In the realm of gaming and virtual environments, companies like EA [303] and Meta [304] are actively exploring real-time 3D character animation and control, particularly for dynamic scenarios involving spontaneous actions like walking, running, jumping, and dribbling [305, 306, 307]. These methods have shown remarkable results in maintaining fluidity and responsiveness in character movement. However, they have not yet been fully explored for their potential in generating complex dance sequences, particularly in minimizing common issues like foot sliding, which is critical in dance animations. Our method, which leverages pre-recorded dance sequences, addresses these challenges effectively, offering a stable foundation for real-time dance generation.

Incorporating these advancements and exploring their potential in dance generation presents an exciting avenue for future research. By combining the strengths of real-time MIR, graph-based and diffusion-based motion synthesis, and cutting-edge gaming animation techniques, we could develop systems that generate highly expressive, real-time dance sequences that are both technically robust and deeply engaging for users.

Lightweight models for scalable applications. Achieving real-time performance does not necessarily mean that a method is lightweight or optimized for devices with limited computational power [268, 269, 270, 271, 272]. For example, the entire 3D dance generation system discussed in Part III requires a high-performance M2 Max Mac Studio to operate smoothly, underscoring its current impracticality for lower-end devices such as mobile phones. The system's computational demands extend beyond just generating 3D dances; they include real-time music information retrieval, inter-model communication, and the rendering of dance outputs, all of which contribute to a substantial computational burden.

Addressing these challenges by developing lightweight models that can function efficiently on less powerful hardware is a critical direction for future research. This involves not only designing

efficient systems from the ground up but also applying advanced model compression techniques to existing models. Techniques such as quantization [308], which reduces the precision of model parameters to decrease computational load, pruning [309], which systematically removes redundant or non-essential elements from a model, and knowledge distillation [310], which transfers knowledge from larger, complex models to smaller, more efficient versions, are all promising strategies.

The key challenge in this endeavor is to maintain a balance where the reduction in model size and complexity does not come at the expense of performance. The goal is to achieve efficiency gains while preserving, or even enhancing, the model's effectiveness. This balance is particularly crucial in applications like real-time 3D dance generation and audio processing, where both speed and quality are imperative.

Exploring lightweight modeling techniques could greatly expand the reach of real-time audio processing and 3D dance generation systems, making them viable on a wider array of devices, including those with limited computational resources. This scalability is crucial for the broader adoption of these technologies, especially in mobile and embedded systems where computational power is often constrained. As demand for real-time, high-quality applications increases, developing efficient, scalable models will be key to enabling these technologies across diverse platforms.

21.1.3 Broader application in related fields

Virtual and augmented reality. Future advancements in virtual and augmented reality (VR/AR) could greatly benefit from the speech enhancement techniques developed in this thesis. Exploring ways to integrate these techniques could lead to significantly clearer and higher-quality audio communication in virtual environments. This integration has the potential to produce more realistic soundscapes in VR applications, enhancing the user's sense of immersion and making the virtual world more convincing and engaging.

Interactive gaming. The principles of real-time dance generation and audio processing developed in this thesis hold exciting potential for the future of interactive gaming, especially in rhythm and dance games. By incorporating these techniques, future gaming experiences could become more dynamic

and responsive, offering players richer and more nuanced audio feedback. This enhancement could elevate player engagement and enjoyment, opening up new possibilities in game design.

Automated customer service systems. In the realm of automated customer service, there is substantial scope for enhancing voice-activated assistants and chatbots using advanced speech processing techniques. Future developments could focus on integrating speech enhancement and dereverberation methods to improve voice recognition accuracy and reliability in diverse acoustic environments. Such advancements would lead to more efficient and user-friendly customer interactions, revolutionizing automated customer service systems.

In conclusion, the research presented in this thesis lays the groundwork for numerous exciting possibilities in the realm of audio processing and human-computer interaction. The future directions outlined in this chapter promise to extend the impact of our current work, fostering new developments in audio processing and interactive technologies, and opening up exciting opportunities for further exploration and innovation.

References

[1] *Amazon alexa*, https://alexa.amazon.com/, Accessed: 2024-01-01.

[2] *Google home assistant*, https://assistant.google.com/, Accessed: 2024-01-01.

[3] *Apple siri*, https://www.apple.com/siri/, Accessed: 2024-01-01.

[4] S. Boll, "Suppression of acoustic noise in speech using spectral subtraction", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 27, no. 2, pp. 113–120, 1979.

[5] M. R. Weiss, E. Aschkenasy, and T. W. Parsons, "Study and development of the intel technique for improving speech intelligibility", Nicolet Scientific Corporation, Technical Report NSC-FR/4023, 1974.

[6] P. Scalart and J. Filho, "Speech enhancement based on a priori signal to noise estimation", in *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 2, 1996, 629–632 vol. 2.

[7] Jae Lim and A. Oppenheim, "All-pole modeling of degraded speech", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 3, pp. 197–210, 1978.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[9] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks", in *Proceedings of the 31st International Conference on Machine Learning*, E. P. Xing and T. Jebara, Eds., ser. Proceedings of Machine Learning Research, vol. 32, Bejing, China: Pmlr, Jun. 2014, pp. 1764–1772.

[10] A. van den Oord *et al.*, "Wavenet: A generative model for raw audio", in *Proceedings of the 9th ISCA Workshop on Speech Synthesis Workshop (SSW 9)*, 2016, p. 125.

[11] D. Rethage, J. Pons, and X. Serra, "A wavenet for speech denoising", in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5069–5073.

[12] P.-S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Joint optimization of masks and deep recurrent neural networks for monaural source separation", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 12, pp. 2136–2147, Dec. 2015.

[13] G. Trigeorgis *et al.*, "Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network", in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 5200–5204.

[14] C. Plachouras, P. Alonso-Jiménez, and D. Bogdanov, *Mir_ref: A representation evaluation framework for music information retrieval tasks*, 2023. arXiv: 2312.05994 [cs.SD].

[15] O. Slizovskaia, G. Haro, and E. Gomez, "Conditioned source separation for musical instrument performances", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 2083–2095, 2021.

[16] A. Lamouret and M. van de Panne, "Motion synthesis by example", in *Computer Animation and Simulation '96*, R. Boulic and G. Hégron, Eds., Vienna: Springer Vienna, 1996, pp. 199–212, ISBN: 978-3-7091-7486-9.

[17] M. Lee, K. Lee, and J. Park, "Music similarity-based approach to generating dance motion sequence", *Multimedia Tools and Applications*, vol. 62, Feb. 2013.

[18] T. Shiratori, A. Nakazawa, and K. Ikeuchi, "Dancing-to-music character animation", *Computer Graphics Forum*, vol. 25, no. 3, pp. 449–458, 2006. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2006.00964.x.

[19] D. Holden, J. Saito, and T. Komura, "A deep learning framework for character motion synthesis and editing", *ACM Trans. Graph.*, vol. 35, no. 4, Jul. 2016.

[20] E. Aksan, M. Kaufmann, and O. Hilliges, *Structured prediction helps 3d human motion modelling*, 2019. arXiv: 1910.09070 [cs.CV].

[21] L. Crnkovic-Friis and L. Crnkovic-Friis, *Generative choreography using deep learning*, 2016. arXiv: 1605.06921 [cs.AI].

[22] H.-Y. Lee *et al.*, *Dancing to music*, 2019. arXiv: 1911.02001 [cs.CV].

[23] R. Xu, R. Wu, Y. Ishiwaka, C. Vondrick, and C. Zheng, "Listening to sounds of silence for speech denoising", in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. Nips'20, Vancouver, BC, Canada: Curran Associates Inc., 2020, ISBN: 9781713829546.

[24] J. Xiang, Y. Zhu, R. Wu, R. Xu, Y. Ishiwaka, and C. Zheng, "Dynamic sliding window for realtime denoising networks", in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 361–365.

[25] R. Xu, G. Krishnan, C. Zheng, and S. K. Nayar, "Personalized dereverberation of speech", in *Proceedings of INTERSPEECH 2023*, 2023, pp. 3859–3863.

[26] R. Xu, V. A. Tran, S. K. Nayar, and G. Krishnan, "Dancecraft: A music-reactive real-time dance improv system", in *Proceedings of the 9th International Conference on Movement and Computing*, ser. Moco '24, Utrecht, Netherlands: Association for Computing Machinery, 2024, ISBN: 9798400709944.

[27] A. J. E. Kell and J. H. McDermott, "Invariance to background noise as a signature of non-primary auditory cortex", *Nature Communications*, vol. 10, no. 1, p. 3958, Sep. 2019.

[28] M. Berouti, R. Schwartz, and J. Makhoul, "Enhancement of speech corrupted by acoustic noise", in *ICASSP '79. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, 1979, pp. 208–211.

[29] L.-p. Yang and Q.-J. Fu, "Spectral subtraction-based speech enhancement for cochlear implant patients in background noise.", *The Journal of the Acoustical Society of America*, vol. 117 3 Pt 1, pp. 1001–4, 2005.

[30] T. Sainburg, *Noise reduction in python using spectral gating*, `https://github.com/timsainb/noisereduce`, 2019.

[31] A. L. Maas, Q. V. Le, T. M. O'Neil, O. Vinyals, P. Nguyen, and A. Y. Ng, "Recurrent neural networks for noise reduction in robust asr", in *Interspeech*, 2012.

[32] C. Valentini-Botinhao, X. Wang, S. Takaki, and J. Yamagishi, "Investigating rnn-based speech enhancement methods for noise-robust text-to-speech", in *9th ISCA Speech Synthesis Workshop*, 2016, pp. 146–152.

[33] F. Weninger *et al.*, "Speech enhancement with lstm recurrent neural networks and its application to noise-robust asr", in *Proceedings of the 12th International Conference on Latent Variable Analysis and Signal Separation - Volume 9237*, ser. Lva/ica 2015, Liberec, Czech Republic: Springer-Verlag, 2015, pp. 91–99, ISBN: 9783319224817.

[34] L. Girin, J.-L. Schwartz, and G. Feng, "Audio-visual enhancement of speech in noise", *The Journal of the Acoustical Society of America*, vol. 109, no. 6, pp. 3007–3020, 2001. eprint: `https://doi.org/10.1121/1.1358887`.

[35] A. Gabbay, A. Shamir, and S. Peleg, *Visual speech enhancement*, 2017. arXiv: `1711.08789 [cs.CV]`.

[36] T. Afouras, J. S. Chung, and A. Zisserman, "The conversation: Deep audio-visual speech enhancement", in *Proceedings of Interspeech 2018*, 2018, pp. 3244–3248.

[37] J.-C. Hou, S.-S. Wang, Y.-H. Lai, Y. Tsao, H.-W. Chang, and H.-m. Wang, "Audio-visual speech enhancement using multimodal deep convolutional neural networks", *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, Mar. 2018.

[38] M. Gogate, A. Adeel, K. Dashtipour, P. Derleth, and A. Hussain, *Av speech enhancement challenge using a real noisy corpus*, 2019. arXiv: 1910.00424 [cs.SD].

[39] S. Pascual, A. Bonafonte, and J. Serrà, "Segan: Speech enhancement generative adversarial network", in *Proceedings of Interspeech 2017*, 2017, pp. 3642–3646.

[40] S. Pascual, J. Serrà, and A. Bonafonte, "Towards generalized speech enhancement with generative adversarial networks", in *Proceedings of Interspeech 2019*, 2019, pp. 1791–1795.

[41] A. Inc., *Adobe audition*, 2020.

[42] S. R. Rochester, "The significance of pauses in spontaneous speech", *Journal of Psycholinguistic Research*, vol. 2, no. 1, pp. 51–81, 1973.

[43] K. L. Fors, "Production and perception of pauses in speech", Ph.D. dissertation, Department of Philosophy, Linguistics, and Theory of Science, University of Gothenburg, 2015.

[44] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion", *ACM Trans. Graph.*, vol. 36, no. 4, Jul. 2017.

[45] P. C. Loizou, *Speech Enhancement: Theory and Practice*, 2nd. Usa: CRC Press, Inc., 2013, ISBN: 1466504218.

[46] P. Smaragdis, C. Févotte, G. J. Mysore, N. Mohammadiha, and M. Hoffman, "Static and dynamic source separation using nonnegative factorizations: A unified view", *IEEE Signal Processing Magazine*, vol. 31, no. 3, pp. 66–75, 2014.

[47] G. Doblinger, "Computationally efficient speech enhancement by spectral minima tracking in subbands", in *Proceedings of Eurospeech*, 1995, pp. 1513–1516.

[48] R. Martin, "Noise power spectral density estimation based on optimal smoothing and minimum statistics", *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 5, pp. 504–512, 2001.

[49] K. V. Sørensen and S. V. Andersen, "Speech enhancement with natural sounding residual noise based on connected time-frequency speech presence regions", *EURASIP J. Adv. Signal Process*, vol. 2005, pp. 2954–2964, Jan. 2005.

[50] Y. Ephraim and D. Malah, "Speech enhancement using a minimum mean-square error log-spectral amplitude estimator", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, no. 2, pp. 443–445, 1985.

[51] S. Rangachari, P. C. Loizou, and Yi Hu, "A noise estimation algorithm with rapid adaptation for highly nonstationary environments", in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, 2004, pp. I–305.

[52] I. Cohen, "Noise spectrum estimation in adverse environments: Improved minima controlled recursive averaging", *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 5, pp. 466–475, 2003.

[53] I. Cohen and B. Berdugo, "Noise estimation by minima controlled recursive averaging for robust speech enhancement", *IEEE Signal Processing Letters*, vol. 9, no. 1, pp. 12–15, 2002.

[54] Y. Ephraim, "Statistical-model-based speech enhancement systems", *Proceedings of the IEEE*, vol. 80, no. 10, pp. 1526–1555, 1992.

[55] H.-G. Hirsch and C. Ehrlicher, "Noise estimation techniques for robust speech recognition", *1995 International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, 153–156 vol.1, 1995.

[56] M. Dendrinos, S. Bakamidis, and G. Carayannis, "Speech enhancement from noise: A regenerative approach", *Speech Commun.*, vol. 10, no. 1, pp. 45–67, Feb. 1991.

[57] Y. Ephraim and H. L. Van Trees, "A signal subspace approach for speech enhancement", *IEEE Transactions on Speech and Audio Processing*, vol. 3, no. 4, pp. 251–266, 1995.

[58] S. Tamura and A. Waibel, "Noise reduction using connectionist models", in *ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing*, 1988, 553–556 vol.1.

[59] S. Parveen and P. Green, "Speech enhancement with missing data techniques using recurrent neural networks", in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, 2004, pp. I–733.

[60] Y. Xu, J. Du, L. Dai, and C. Lee, "An experimental study on speech enhancement based on deep neural networks", *IEEE Signal Processing Letters*, vol. 21, no. 1, pp. 65–68, 2014.

[61] Y. Xu, J. Du, Z. Huang, L.-R. Dai, and C.-H. Lee, "Multi-objective learning and mask-based post-processing for deep neural network based speech enhancement", in *Interspeech*, 2015.

[62] Y. Xu, J. Du, L. Dai, and C. Lee, "A regression approach to speech enhancement based on deep neural networks", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 1, pp. 7–19, 2015.

[63] A. Kumar and D. A. F. Florêncio, "Speech enhancement in multiple-noise conditions using deep neural networks", in *Interspeech*, 2016.

[64] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997.

[65] Z. C. Lipton, J. Berkowitz, and C. Elkan, *A critical review of recurrent neural networks for sequence learning*, 2015. arXiv: 1506.00019 [cs.LG].

[66] I. J. Goodfellow *et al.*, "Generative adversarial nets", in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. Nips'14, Montreal, Canada: MIT Press, 2014, pp. 2672–2680.

[67] S.-W. Fu, Y. Tsao, X. Lu, and H. Kawai, "Raw waveform-based speech enhancement by fully convolutional networks", *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, Dec. 2017.

[68] A. Pandey and D. Wang, "A new framework for supervised speech enhancement in the time domain", in *Proceedings of Interspeech 2018*, 2018, pp. 1136–1140.

[69] M. Michelashvili and L. Wolf, *Audio denoising with deep network priors*, 2019. arXiv: 1904.07612 [cs.SD].

[70] Y. Luo and N. Mesgarani, "Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation", *IEEE/ACM Trans. Audio, Speech and Language Processing*, vol. 27, no. 8, pp. 1256–1266, Aug. 2019.

[71] K. Qian, Y. Zhang, S. Chang, X. Yang, D. Florêncio, and M. Hasegawa-Johnson, "Speech enhancement using bayesian wavenet", in *Proceedings of Interspeech 2017*, 2017, pp. 2013–2017.

[72] F. G. Germain, Q. Chen, and V. Koltun, "Speech denoising with deep feature losses", in *Proceedings of Interspeech 2019*, 2019, pp. 2723–2727.

[73] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, "Speech enhancement based on deep denoising autoencoder", in *Interspeech*, 2013.

[74] Y. Wang and D. Wang, "Cocktail party processing via structured prediction", in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. Nips'12, Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 224–232.

[75] A. Narayanan and D. Wang, "Ideal ratio mask estimation using deep neural networks for robust speech recognition", in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 7092–7096.

[76] F. Weninger, J. R. Hershey, J. Le Roux, and B. Schuller, "Discriminatively trained recurrent neural networks for single-channel speech separation", in *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2014, pp. 577–581.

[77] A. Kumar and D. Florencio, "Speech enhancement in multiple-noise conditions using deep neural networks", *Interspeech 2016*, Sep. 2016.

[78] X. Zhang and D. Wang, "A deep ensemble learning method for monaural speech separation", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 5, pp. 967–977, 2016.

[79] J. Chen and D. Wang, "Long short-term memory for speaker generalization in supervised speech separation", *Acoustical Society of America Journal*, vol. 141, no. 6, pp. 4705–4714, Jun. 2017.

[80] D. Wang and Jae Lim, "The unimportance of phase in speech enhancement", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 30, no. 4, pp. 679–681, 1982.

[81] K. Paliwal, K. Wójcicki, and B. Shannon, "The importance of phase in speech enhancement", *Speech Commun.*, vol. 53, no. 4, pp. 465–494, Apr. 2011.

[82] J. Le Roux and E. Vincent, "Consistent wiener filtering for audio source separation", *IEEE Signal Processing Letters*, vol. 20, no. 3, pp. 217–220, 2013.

[83] F. G. Germain, G. J. Mysore, and T. Fujioka, "Equalization matching of speech recordings in real-world environments", in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 609–613.

[84] T. Gerkmann, M. Krawczyk-Becker, and J. Le Roux, "Phase processing for single-channel speech enhancement: History and recent advances", *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 55–66, 2015.

[85] Y. Wang and D. Wang, "A deep neural network for time-domain signal reconstruction", in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4390–4394.

[86] H. Erdogan, J. R. Hershey, S. Watanabe, and J. Le Roux, "Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks", in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 708–712.

[87] D. S. Williamson and D. Wang, "Time-frequency masking in the complex domain for speech dereverberation and denoising", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 7, pp. 1492–1501, 2017.

[88] J. A. Moorer, "A note on the implementation of audio processing by short-term fourier transform", in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2017, pp. 156–159.

[89] C. Busso and S. S. Narayanan, "Interrelation between speech and facial gestures in emotional utterances: A single subject study", *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 8, pp. 2331–2347, 2007.

[90] A. Adeel, M. Gogate, A. Hussain, and W. M. Whitmer, "Lip-reading driven deep learning approach for speech enhancement", *IEEE Transactions on Emerging Topics in Computational Intelligence*, pp. 1–10, 2019.

[91] A. Owens and A. A. Efros, "Audio-visual scene analysis with self-supervised multisensory features", *Lecture Notes in Computer Science*, pp. 639–658, 2018.

[92] A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba, "Ambient sound provides supervision for visual learning", in *European conference on computer vision*, Springer, 2016, pp. 801–816.

[93] Y. Aytar, C. Vondrick, and A. Torralba, "Soundnet: Learning sound representations from unlabeled video", in *Advances in neural information processing systems*, 2016, pp. 892–900.

[94] R. Arandjelovic and A. Zisserman, "Objects that sound", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 435–451.

[95] T. Le Cornu and B. Milner, "Generating intelligible audio speech from visual speech", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 9, pp. 1751–1761, 2017.

[96] A. Ephrat, T. Halperin, and S. Peleg, "Improved speech reconstruction from silent video", in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017, pp. 455–462.

[97] A. Owens, P. Isola, J. McDermott, A. Torralba, E. H. Adelson, and W. T. Freeman, "Visually indicated sounds", *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[98] T. Afouras, J. S. Chung, A. Senior, O. Vinyals, and A. Zisserman, "Deep audio-visual speech recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2018.

[99] K. Noda, Y. Yamaguchi, K. Nakadai, H. G. Okuno, and T. Ogata, "Audio-visual speech recognition using deep learning", *Applied Intelligence*, vol. 42, no. 4, pp. 722–737, Jun. 2015.

[100] A. Gabbay, A. Ephrat, T. Halperin, and S. Peleg, *Seeing through noise: Visually driven speaker separation and enhancement*, 2017. arXiv: 1708.06767 [cs.CV].

[101] A. Ephrat *et al.*, "Looking to listen at the cocktail party: A speaker-independent audio-visual model for speech separation", *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 1–11, Jul. 2018.

[102] H. Zhao, C. Gan, A. Rouditchenko, C. Vondrick, J. McDermott, and A. Torralba, "The sound of pixels", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 570–586.

[103] Z. Zhang *et al.*, "Deep audio priors emerge from harmonic convolutional networks", in *International Conference on Learning Representations*, 2020.

[104] L. Wyse, *Audio spectrogram representations for processing with convolutional neural networks*, 2017. arXiv: 1706.09559 [cs.SD].

[105] J. L. Flanagan, *Speech Analysis Synthesis and Perception*, 2nd. Springer-Verlag, 1972, ISBN: 9783662015629.

[106] E. Sejdic, I. Djurovic, and L. Stankovic, "Quantitative performance analysis of scalogram as instantaneous frequency estimator", *IEEE Transactions on Signal Processing*, vol. 56, no. 8, pp. 3837–3845, 2008.

[107] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks", *Signal Processing, IEEE Transactions on*, vol. 45, pp. 2673–2681, Dec. 1997.

[108] S. Mehri *et al.*, *Samplernn: An unconditional end-to-end neural audio generation model*, 2016. arXiv: 1612.07837 [cs.SD].

[109] N. Kalchbrenner *et al.*, *Efficient neural audio synthesis*, 2018. arXiv: 1802.08435 [cs.SD].

[110] H. Purwins, B. Li, T. Virtanen, J. Schluter, S.-Y. Chang, and T. Sainath, "Deep learning for audio signal processing", *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 206–219, May 2019.

[111] D. Wang and J. Chen, "Supervised speech separation based on deep learning: An overview", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 10, pp. 1702–1726, Oct. 2018.

[112] Y. Wang, A. Narayanan, and D. Wang, "On training targets for supervised speech separation", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 12, pp. 1849–1858, 2014.

[113] J. Thiemann, N. Ito, and E. Vincent, "The diverse environments multi-channel acoustic noise database (demand): A database of multichannel environmental noise recordings", in *21st International Congress on Acoustics*, The dataset itself is archived on Zenodo, with DOI 10.5281/zenodo.1227120, Acoustical Society of America, Montreal, Canada, Jun. 2013.

[114] J. F. Gemmeke *et al.*, "Audio set: An ontology and human-labeled dataset for audio events", in *Proceedings of IEEE ICASSP 2017*, New Orleans, LA, 2017.

[115] A. Rix, J. Beerends, M. Hollier, and A. Hekstra, "Perceptual evaluation of speech quality (pesq): A new method for speech quality assessment of telephone networks and codecs", in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, vol. 2, Feb. 2001, 749–752 vol.2, ISBN: 0-7803-7041-4.

[116] M. A. C. Schuyler R. Quackenbush Thomas P. Barnwell, *Objective Measures Of Speech Quality*. Englewood Cliffs, NJ: Prentice Hall, 1988, ISBN: 9780136290568.

[117] C. Taal, R. Hendriks, R. Heusdens, and J. Jensen, "A short-time objective intelligibility measure for time-frequency weighted noisy speech", in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, Apr. 2010, pp. 4214–4217.

[118] Y. Hu and P. Loizou, "Evaluation of objective quality measures for speech enhancement", *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, pp. 229–238, Feb. 2008.

[119] D. K. Freeman, G. Cosier, C. B. Southcott, and I. Boyd, "The voice activity detector for the pan-european digital cellular mobile telephone service", in *International Conference on Acoustics, Speech, and Signal Processing,*, 1989, 369–372 vol.1.

[120] R. Le Bouquin Jeannes and G. Faucon, "Proposal of a voice activity detector for noise reduction", *Electronics Letters*, vol. 30, no. 12, pp. 930–932, 1994.

[121] R. Le Bouquin Jeannes and G. Faucon, "Study of a voice activity detector and its influence on a noise reduction system", *Speech Communication*, vol. 16, no. 3, pp. 245–254, 1995.

[122] J. Wiseman, *Py-webrtcvad*, `https://github.com/wiseman/py-webrtcvad`, 2019.

[123] M. H. Soni, N. Shah, and H. A. Patil, "Time-frequency masking-based speech enhancement using generative adversarial network", in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5039–5043.

[124] S.-W. Fu, C.-F. Liao, Y. Tsao, and S.-D. Lin, *Metricgan: Generative adversarial networks based black-box metric scores optimization for speech enhancement*, 2019. arXiv: `1905.04874 [cs.SD]`.

[125] J. Kim, M. El-Kharmy, and J. Lee, *End-to-end multi-task denoising for joint sdr and pesq optimization*, 2019. arXiv: `1901.09146 [cs.SD]`.

[126] J. Kim, M. El-Khamy, and J. Lee, "T-gsa: Transformer with gaussian-weighted self-attention for speech enhancement", in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6649–6653.

[127] Y. Koizumi, K. Yatabe, M. Delcroix, Y. Masuyama, and D. Takeuchi, *Speech enhancement using self-adaptation and multi-head self-attention*, 2020. arXiv: 2002.05873 [eess.AS].

[128] M. Nikzad, A. Nicolson, Y. Gao, J. Zhou, K. K. Paliwal, and F. Shang, *Deep residual-dense lattice network for speech enhancement*, 2020. arXiv: 2002.12794 [eess.AS].

[129] W. Wei and E. Huerta, "Gravitational wave denoising of binary black hole mergers with deep learning", *Physics Letters B*, vol. 800, p. 135 081, 2020.

[130] A. Defossez, G. Synnaeve, and Y. Adi, *Real time speech enhancement in the waveform domain*, 2020.

[131] X. Hao, X. Su, R. Horaud, and X. Li, "Fullsubnet: A full-band and sub-band fusion model for real-time single-channel speech enhancement", in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Ieee, 2021, pp. 6633–6637.

[132] T. Vuong, Y. Xia, and R. M. Stern, *A modulation-domain loss for neural-network-based real-time speech enhancement*, 2021. arXiv: 2102.07330 [eess.AS].

[133] Q. Zhang, A. Nicolson, M. Wang, K. K. Paliwal, and C. Wang, "Deepmmse: A deep learning approach to mmse-based noise power spectral density estimation", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1404–1415, 2020.

[134] H.-S. Choi, S. Park, J. H. Lee, H. Heo, D. Jeon, and K. Lee, "Real-time denoising and dereverberation with tiny recurrent u-net", *arXiv preprint arXiv:2102.03207*, 2021.

[135] Y. Xia, S. Braun, C. K. Reddy, H. Dubey, R. Cutler, and I. Tashev, "Weighted speech distortion losses for neural-network-based real-time speech enhancement", in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Ieee, 2020, pp. 871–875.

[136] L. Cheng, J. Li, and Y. Yan, "Fscnet: Feature-specific convolution neural network for real-time speech enhancement", *IEEE Signal Processing Letters*, 2021.

[137] K. Tan and D. Wang, "A convolutional recurrent neural network for real-time speech enhancement", in *Interspeech*, 2018, pp. 3229–3233.

[138] P. Howell, "Effect of speaking environment on speech production and perception", *Journal of the human-environment system : JHES*, vol. 11, pp. 51–57, Nov. 2008.

[139] R. W. S. Alan V. Oppenheim, *Digital Signal Processing*. Englewood Cliffs, NJ: Prentice Hall, 1975, ISBN: 9780132146357.

[140] S. Neely and J. Allen, "Invertibility of a room impulse response", *The Journal of the Acoustical Society of America*, vol. 66, pp. 165–169, Jul. 1979.

[141] M. Miyoshi and Y. Kaneda, "Inverse filtering of room acoustics", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 2, pp. 145–152, 1988.

[142] N. Wiener, *Extrapolation Interpolation and Smoothing of Stationary Time Series*. The MIT Press, 1964, ISBN: 0262730057.

[143] P. Naylor and N. Gaubitch, *Speech Dereverberation*. Springer London, Jan. 2011, vol. 59, ISBN: 978-1-84996-055-7.

[144] J. B. Allen, D. A. Berkley, and J. Blauert, "Multimicrophone signal-processing technique to remove room reverberation from speech signals", *Journal of the Acoustical Society of America*, vol. 62, pp. 912–915, 1977.

[145] J. Moorer, "About this reverberation business", in Jstor, Jan. 1985, vol. 3, pp. 605–639, ISBN: 978-0262680516.

[146] D. Cole, M. Moody, and S. Sridharan, "Position-independent enhancement of reverberant speech", *J. Audio Eng. Soc*, vol. 45, no. 3, pp. 142–147, 1997.

[147] J. Mourjopoulos and J. Hammond, "Modelling and enhancement of reverberant speech using an envelope convolution method", in *ICASSP '83. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 8, 1983, pp. 1144–1147.

[148] T. Langhans and H. Strube, "Speech enhancement by nonlinear multiband envelope filtering", in *ICASSP '82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 7, 1982, pp. 156–159.

[149] P. M. Clarkson and S. F. Bahgat, "Envelope expansion methods for speech enhancement.", *The Journal of the Acoustical Society of America*, vol. 89 3, pp. 1378–82, 1991.

[150] B. Gillespie, H. Malvar, and D. Florencio, "Speech dereverberation via maximum-kurtosis subband adaptive filtering", in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, vol. 6, Feb. 2001, 3701–3704 vol.6, ISBN: 0-7803-7041-4.

[151] K. Furuya and Y. Kaneda, "Two-channel blind deconvolution for non-minimum phase impulse responses", in *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, 1997, 1315–1318 vol.2.

[152] B. Yegnanarayana, P. Satyanarayana Murthy, C. Avendano, and H. Hermansky, "Enhancement of reverberant speech using lp residual", in *Proceedings of the 1998 IEEE International*

Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181), vol. 1, 1998, 405–408 vol.1.

[153] M. Brandstein and S. Griebel, "Explicit speech modeling for microphone array applications", in *Microphone Arrays: Signal Processing Techniques and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, Jan. 2001, pp. 133–153, ISBN: 978-3-642-07547-6.

[154] M. Tonelli, N. Mitianoudis, and M. Davies, "A maximum-likelihood approach to blind audio de-reverberation", *Proceedings of the 7th Int. Conference on Digital Audio Effects (DAFX-04), Naples, Italy, October 5-8, 200*, pp. 2004–1, Nov. 2008.

[155] B. Yegnanarayana and P. Murthy, "Enhancement of reverberant speech using lp residual signal", *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 3, pp. 267–281, 2000.

[156] M. Wu and D. Wang, "A two-stage algorithm for one-microphone reverberant speech enhancement", *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 3, pp. 774–784, 2006.

[157] N. Kilis and N. Mitianoudis, "A novel scheme for single-channel speech dereverberation", *Acoustics*, vol. 1, no. 3, pp. 711–725, 2019.

[158] K. Lebart, J.-M. Boucher, and P. Denbigh, "A new method based on spectral subtraction for speech dereverberation", *Acta Acustica united with Acustica*, vol. 87, pp. 359–366, May 2001.

[159] E. Habets, "Multi-channel speech dereverberation based on a statistical model of late reverberation", in *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, vol. 4, 2005, iv/173–iv/176 Vol. 4.

[160] E. A. P. Habets, "Towards multi-microphone speech dereverberation using spectral enhancement and statistical reverberation models", in *2008 42nd Asilomar Conference on Signals, Systems and Computers*, 2008, pp. 806–810.

[161] E. Habets, "Single- and multi-microphone speech dereverberation using spectral enhancement", Ph.D. dissertation, Eindhoven University of Technology, Jan. 2007.

[162] T. Nakatani, T. Yoshioka, K. Kinoshita, M. Miyoshi, and B.-H. Juang, "Blind speech dereverberation with multi-channel linear prediction based on short time fourier transform representation", in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008, pp. 85–88.

[163] T. Nakatani, T. Yoshioka, K. Kinoshita, M. Miyoshi, and B.-H. Juang, "Speech dereverberation based on variance-normalized delayed linear prediction", *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 7, pp. 1717–1731, 2010.

[164] M. Parchami, W.-P. Zhu, and B. Champagne, "Speech dereverberation using linear prediction with estimation of early speech spectral variance", in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 504–508.

[165] M. Parchami, W.-P. Zhu, and B. Champagne, "Speech dereverberation using weighted prediction error with correlated inter-frame speech components", *Speech Commun.*, vol. 87, no. C, pp. 49–57, Mar. 2017.

[166] T. Yoshioka and T. Nakatani, "Generalization of multi-channel linear prediction methods for blind mimo impulse response shortening", *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 10, pp. 2707–2720, 2012.

[167] J. Caroselli, I. Shafran, A. Narayanan, and R. Rose, "Adaptive multichannel dereverberation for automatic speech recognition", in *Proceedings of Interspeech 2017*, Aug. 2017, pp. 3877–3881.

[168] H. L. V. Trees, *Detection, Estimation, and Modulation Theory, Optimum Array Processing*. Wiley-Interscience, 2002, ISBN: 9780471093909.

[169] G. W. Elko, "Microphone array systems for hands-free telecommunication", *Speech Communication*, vol. 20, no. 3, pp. 229–240, 1996, Acoustic Echo Control and Speech Enhancement Techniques.

[170] S. Fischer and K. U. Simmer, "Beamforming microphone arrays for speech acquisition in noisy environments", *Speech Commun.*, vol. 20, pp. 215–227, 1996.

[171] A. v. d. Oord *et al.*, *Wavenet: A generative model for raw audio*, 2016.

[172] D. Rethage, J. Pons, and X. Serra, *A wavenet for speech denoising*, 2017.

[173] J. Su, A. Finkelstein, and Z. Jin, "Perceptually-motivated environment-specific speech enhancement", in *Icassp 2019*, May 2019.

[174] D. Stoller, S. Ewert, and S. Dixon, "Wave-u-net: A multi-scale neural network for end-to-end audio source separation", *CoRR*, 2018.

[175] C. Macartney and T. Weyde, *Improved speech enhancement with the wave-u-net*, 2018.

[176] O. Ernst, S. E. Chazan, S. Gannot, and J. Goldberger, *Speech dereverberation using fully convolutional networks*, 2018.

[177] D. León and F. Tobar, *Late reverberation suppression using u-nets*, 2021.

[178] X. Xiao *et al.*, "Speech dereverberation for enhancement and recognition using dynamic features constrained deep neural networks and feature adaptation", *EURASIP Journal on Advances in Signal Processing*, vol. 2016, no. 1, p. 4, Jan. 13, 2016.

[179] V. Kothapally, W. Xia, S. Ghorbani, J. H. Hansen, W. Xue, and J. Huang, "SkipConvNet: Skip convolutional neural network for speech dereverberation using optimally smoothed spectral mapping", in *Interspeech 2020*, Isca, Oct. 2020.

[180] J. Su, Z. Jin, and A. Finkelstein, *Hifi-gan: High-fidelity denoising and dereverberation based on speech deep features in adversarial networks*, 2020.

[181] S. Pascual, A. Bonafonte, and J. Serrà, *Segan: Speech enhancement generative adversarial network*, 2017.

[182] S. Pascual, J. Serrà, and A. Bonafonte, *Towards generalized speech enhancement with generative adversarial networks*, 2019.

[183] S.-W. Fu, C.-F. Liao, and Y. Tsao, "Learning with learned loss function: Speech enhancement with quality-net to improve perceptual evaluation of speech quality", *IEEE Signal Processing Letters*, vol. 27, pp. 26–30, 2020.

[184] A. Ratnarajah, Z. Tang, and D. Manocha, *Ir-gan: Room impulse response generator for far-field speech recognition*, 2020.

[185] A. Ratnarajah, Z. Tang, and D. Manocha, *Ts-rir: Translated synthetic room impulse responses for speech augmentation*, 2021.

[186] J. Allen and D. Berkley, "Image method for efficiently simulating small-room acoustics", *The Journal of the Acoustical Society of America*, vol. 65, pp. 943–950, Apr. 1979.

[187] Z. Tang, L. Chen, B. Wu, D. Yu, and D. Manocha, "Improving reverberant speech training using diffuse acoustic simulation", in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Ieee, May 2020.

[188] A. F. Genovese, H. Gamper, V. Pulkki, N. Raghuvanshi, and I. J. Tashev, "Blind room volume estimation from single-channel noisy speech", in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 231–235.

[189] F. Xiong, S. Goetze, and B. T. Meyer, *Joint estimation of reverberation time and direct-to-reverberation ratio from speech using auditory-inspired features*, 2015.

[190] H. Gamper and I. J. Tashev, "Blind reverberation time estimation using a convolutional neural network", in *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, 2018, pp. 136–140.

[191] N. J. Bryan, *Impulse response data augmentation and deep neural networks for blind room acoustic parameter estimation*, 2019.

[192] A. Farina, "Simultaneous measurement of impulse response and distortion with a swept-sine technique", *Journal of the Audio Engineering Society*, Feb. 2000.

[193] P. Guidorzi, L. Barbaresi, D. D'Orazio, and M. Garai, "Impulse responses measured with mls or swept-sine signals applied to architectural acoustics: An in-depth analysis of the two methods and some case studies of measurements inside theaters", *Energy Procedia*, vol. 78, pp. 1611–1616, 2015, 6th International Building Physics Conference, IBPC 2015.

[194] P. Samarasinghe, T. D. Abhayapala, and W. Kellermann, "Acoustic reciprocity: An extension to spherical harmonics domain", *The Journal of the Acoustical Society of America*, vol. 142, no. 4, El337–el343, 2017. eprint: https://doi.org/10.1121/1.5002078.

[195] K. Wapenaar, "Unified matrix–vector wave equation, reciprocity and representations", *Geophysical Journal International*, vol. 216, no. 1, pp. 560–583, Oct. 2018. eprint: https://academic.oup.com/gji/article-pdf/216/1/560/26663159/ggy451.pdf.

[196] J. D. Achenbach, "Reciprocity in acoustics", in *Reciprocity in Elastodynamics* (Cambridge Monographs on Mechanics), Cambridge Monographs on Mechanics. Cambridge University Press, 2004, pp. 55–69.

[197] J. Dong, S. Roth, and B. Schiele, "Deep wiener deconvolution: Wiener meets deep learning for image deblurring", *Advances in Neural Information Processing Systems*, vol. 33, pp. 1048–1059, 2020.

[198] K. Kumar *et al.*, "Melgan: Generative adversarial networks for conditional waveform synthesis", in *Advances in Neural Information Processing Systems*, 2019.

[199] J. Kong, J. Kim, and J. Bae, "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis", in *Advances in Neural Information Processing Systems*, 2020.

[200] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An asr corpus based on public domain audio books", in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015, pp. 5206–5210.

[201] M. Jeub, M. Schafer, and P. Vary, "A binaural room impulse response database for the evaluation of dereverberation algorithms", in *2009 16th International Conference on Digital Signal Processing*, 2009, pp. 1–5.

[202] I. Szöke, M. Skácel, L. Mošner, J. Paliesek, and J. Černocký, "Building and evaluation of a real room impulse response dataset", *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 4, pp. 863–876, 2019.

[203] L. Drude, J. Heymann, C. Boeddeker, and R. Haeb-Umbach, "NARA-WPE: A python package for weighted prediction error dereverberation in Numpy and Tensorflow for online and offline processing", in *13. ITG Fachtagung Sprachkommunikation (ITG 2018)*, Oct. 2018.

[204] T. Falk, C. Zheng, and W.-Y. Chan, "A non-intrusive quality and intelligibility measure of reverberant and dereverberated speech", *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 18, pp. 1766–1774, Oct. 2010.

[205] *Sonic sphere*, `https://www.theshed.org/program/304-sonic-sphere`, Accessed: 2024-01-01.

[206] T. Tang, J. Jia, and H. Mao, "Dance with melody: An lstm-autoencoder approach to music-oriented dance synthesis", in *Proceedings of the 26th ACM International Conference on Multimedia*, ser. MM '18, Seoul, Republic of Korea: Association for Computing Machinery, 2018, pp. 1598–1606, ISBN: 9781450356657.

[207] O. Alemi and P. Pasquier, "Groovenet : Real-time music-driven dance movement generation using artificial neural networks", 2017.

[208] G. Sun, Y. Wong, Z. Cheng, M. S. Kankanhalli, W. Geng, and X. Li, "Deepdance: Music-to-dance motion choreography with adversarial learning", *IEEE Transactions on Multimedia*, vol. 23, pp. 497–509, 2021.

[209] R. Huang, H. Hu, W. Wu, K. Sawada, M. Zhang, and D. Jiang, *Dance revolution: Long-term dance generation with music via curriculum learning*, 2023. arXiv: `2006.06119 [cs.CV]`.

[210] B. Li, Y. Zhao, S. Zhelun, and L. Sheng, "Danceformer: Music conditioned 3d dance generation with parametric motion transformer", in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 1272–1279.

[211] J. P. M. Ferreira *et al.*, "Learning to dance: A graph convolutional adversarial network to generate realistic dance motions from audio", *CoRR*, vol. abs/2011.12999, 2020. arXiv: `2011.12999`.

[212] Z. Ye *et al.*, "ChoreoNet: Towards music to dance synthesis with choreographic action unit", in *Proceedings of the 28th ACM International Conference on Multimedia*, Acm, Oct. 2020.

[213] C. Kang *et al.*, "Choreomaster: Choreography-oriented music-driven dance synthesis", *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, 2021.

[214] K. Sasaki, *Bigaku Jiten = Dictionary of Aesthetics*. Tokyo Daigaku Shuppankai, 1995.

[215] M. m. Ribeiro and A. Fonseca, "The empathy and the structuring sharing modes of movement sequences in the improvisation of contemporary dance", *Research in Dance Education*, vol. 12, no. 2, pp. 71–85, 2011.

[216] S. Paxton, "Contact improvisation", *The Drama Review: TDR*, vol. 19, no. 1, pp. 40–42, 1975.

[217] Y. Nakano and T. Okada, "Process of improvisational contemporary dance", in *34th Annual Meeting of the Cognitive Science Society*, 2012.

[218] K. Carey, A. Moran, and B. Rooney, "Learning choreography: An investigation of motor imagery, attentional effort, and expertise in modern dance", *Frontiers in Psychology*, vol. 10, Mar. 2019.

[219] S. Inc., *Bitmoji*, `https://www.bitmoji.com/`, Accessed: 2024-01-01.

[220] S. Inc., *Happy 15th birthday, bitmoji*, `https://newsroom.snap.com/happy-birthday-bitmoji/`, Accessed: 2024-01-01.

[221] L. Tanco and A. Hilton, "Realistic synthesis of novel human movements from a database of motion capture examples", in *Proceedings Workshop on Human Motion*, 2000, pp. 137–142.

[222] R. Bowden, *Learning statistical models of human motion*, 2000.

[223] M. Brand and A. Hertzmann, "Style machines", in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, ser. Siggraph '00, Usa: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 183–192, ISBN: 1581132085.

[224] A. M. Lehrmann, P. V. Gehler, and S. Nowozin, "Efficient nonlinear markov models for human motion", in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1314–1321.

[225] V. Pavlovic, J. M. Rehg, and J. MacCormick, "Learning switching linear models of human motion", in *Proceedings of the 13th International Conference on Neural Information Processing Systems*, ser. Nips'00, Denver, CO: MIT Press, 2000, pp. 942–948.

[226] J. Chai and J. K. Hodgins, "Constraint-based motion optimization using a statistical dynamic model", *ACM Trans. Graph.*, vol. 26, no. 3, 8–es, Jul. 2007.

[227] M. Lau, Z. Bar-Joseph, and J. Kuffner, "Modeling spatial and temporal variation in motion data", *ACM Trans. Graph.*, vol. 28, no. 5, pp. 1–10, Dec. 2009.

[228] D. Holden, J. Saito, T. Komura, and T. Joyce, "Learning motion manifolds with convolutional autoencoders", in *SIGGRAPH Asia 2015 Technical Briefs*, ser. Sa '15, Kobe, Japan: Association for Computing Machinery, 2015, ISBN: 9781450339308.

[229] L.-Y. Gui, Y.-X. Wang, X. Liang, and J. M. F. Moura, "Adversarial geometry-aware human motion prediction", in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., Cham: Springer International Publishing, 2018, pp. 823–842, ISBN: 978-3-030-01225-0.

[230] A. H. Ruiz, J. Gall, and F. Moreno-Noguer, *Human motion prediction via spatio-temporal inpainting*, 2019. arXiv: 1812.05478 [cs.CV].

[231] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik, *Recurrent network models for human dynamics*, 2015. arXiv: 1508.00271 [cs.CV].

[232] P. Ghosh, J. Song, E. Aksan, and O. Hilliges, *Learning human motion models for long-term predictions*, 2017. arXiv: 1704.02827 [cs.CV].

[233] J. Martinez, M. J. Black, and J. Romero, *On human motion prediction using recurrent neural networks*, 2017. arXiv: 1705.02445 [cs.CV].

[234] D. Pavllo, D. Grangier, and M. Auli, *Quaternet: A quaternion-based recurrent model for human motion*, 2018. arXiv: 1805.06485 [cs.CV].

[235] J. Bütepage, M. Black, D. Kragic, and H. Kjellström, *Deep representation learning for human motion prediction and classification*, 2017. arXiv: 1702.07486 [cs.CV].

[236] H.-k. Chiu, E. Adeli, B. Wang, D.-A. Huang, and J. C. Niebles, *Action-agnostic human pose forecasting*, 2018. arXiv: 1810.09676 [cs.CV].

[237] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, *Structural-rnn: Deep learning on spatio-temporal graphs*, 2016. arXiv: 1511.05298 [cs.CV].

[238] B. Wang, E. Adeli, H.-k. Chiu, D.-A. Huang, and J. C. Niebles, *Imitation learning for human pose prediction*, 2019. arXiv: 1909.03449 [cs.CV].

[239] E. Aksan, M. Kaufmann, P. Cao, and O. Hilliges, *A spatio-temporal transformer for 3d human motion prediction*, 2021. arXiv: 2004.08692 [cs.CV].

[240] R. Li, S. Yang, D. A. Ross, and A. Kanazawa, *Ai choreographer: Music conditioned 3d dance generation with aist++*, 2021.

[241] Y. Duan *et al.*, *Single-shot motion completion with transformer*, 2021. arXiv: 2103.00776 [cs.CV].

[242] F. G. Harvey, M. Yurick, D. Nowrouzezahrai, and C. Pal, "Robust motion in-betweening", *ACM Trans. Graph.*, vol. 39, no. 4, Aug. 2020.

[243] J. Kim, T. Byun, S. Shin, J. Won, and S. Choi, "Conditional motion in-betweening", *Pattern Recognition*, p. 108 894, 2022.

[244] C. Guo, Y. Mu, M. G. Javed, S. Wang, and L. Cheng, "Momask: Generative masked modeling of 3d human motions", 2023. arXiv: 2312.00063 [cs.CV].

[245] J. Lee and S. Y. Shin, "A hierarchical approach to interactive motion editing for human-like figures", in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, ser. Siggraph '99, Usa: ACM Press/Addison-Wesley Publishing Co., 1999, pp. 39–48, ISBN: 0201485605.

[246] L. Kovar, M. Gleicher, and F. H. Pighin, "Motion graphs", in *International Conference on Computer Graphics and Interactive Techniques*, 2002.

[247] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs", *ACM SIGGRAPH 2008 classes*, 2008.

[248] O. Arikan and D. A. Forsyth, "Interactive motion generation from examples", *ACM Trans. Graph.*, vol. 21, no. 3, pp. 483–490, Jul. 2002.

[249] J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard, "Interactive control of avatars animated with human motion data", vol. 21, no. 3, pp. 491–500, Jul. 2002.

[250] A. Safonova and J. K. Hodgins, "Construction and optimal search of interpolated motion graphs", *ACM Trans. Graph.*, vol. 26, no. 3, 106–es, Jul. 2007.

[251] J. Min and J. Chai, "Motion graphs++: A compact generative model for semantic motion analysis and synthesis", *ACM Trans. Graph.*, vol. 31, no. 6, Nov. 2012.

[252] T.-h. Kim, S. I. Park, and S. Y. Shin, "Rhythmic-motion synthesis based on motion-beat analysis", *ACM Trans. Graph.*, vol. 22, no. 3, pp. 392–401, Jul. 2003.

[253] J. W. Kim, H. Fouad, and J. K. Hahn, "Making them dance", in *AAAI Fall Symposium: Aurally Informed Performance*, 2006.

[254] A. Berman and V. James, "Kinetic imaginations: Exploring the possibilities of combining ai and dance", in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. Ijcai'15, Buenos Aires, Argentina: AAAI Press, 2015, pp. 2431–2437, ISBN: 9781577357384.

[255] W. Li, X. Chen, P. Li, O. Sorkine-Hornung, and B. Chen, "Example-based motion synthesis via generative motion matching", *ACM Transactions on Graphics (TOG)*, vol. 42, no. 4, 2023.

[256] C. Rose, M. Cohen, and B. Bodenheimer, "Verbs and adverbs: Multidimensional motion interpolation", *IEEE Computer Graphics and Applications*, vol. 18, no. 5, pp. 32–40, 1998.

[257] T. Mukai and S. Kuriyama, "Geostatistical motion interpolation", *ACM Trans. Graph.*, vol. 24, no. 3, pp. 1062–1070, Jul. 2005.

[258] R. Heck and M. Gleicher, "Parametric motion graphs", in *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, ser. I3d '07, Seattle, Washington: Association for Computing Machinery, 2007, pp. 129–136, ISBN: 9781595936288.

[259] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 283–298, 2008.

[260] F. G. Harvey and C. Pal, *Recurrent transition networks for character locomotion*, 2021. arXiv: 1810.02363 [cs.GR].

[261] X. Tang *et al.*, "Real-time controllable motion transition for characters", *ACM Transactions on Graphics*, vol. 41, no. 4, pp. 1–10, Jul. 2022.

[262] M. Kaufmann, E. Aksan, J. Song, F. Pece, R. Ziegler, and O. Hilliges, "Convolutional autoencoders for human motion infilling", in *2020 International Conference on 3D Vision (3DV)*, Ieee, Nov. 2020.

[263] Y. Zhou, J. Lu, C. Barnes, J. Yang, S. Xiang, and H. li, *Generative tweening: Long-term inbetweening of 3d human motions*, 2020. arXiv: 2005.08891 [cs.CV].

[264] B. N. Oreshkin, A. Valkanas, F. G. Harvey, L.-S. Ménard, F. Bocquelet, and M. J. Coates, *Motion inbetweening via deep Δ-interpolator*, 2022. arXiv: 2201.06701 [cs.LG].

[265] J. Qin, Y. Zheng, and K. Zhou, "Motion in-betweening via two-stage transformers", *ACM Trans. Graph.*, vol. 41, no. 6, Nov. 2022.

[266] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models", *arXiv preprint arxiv:2006.11239*, 2020.

[267] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, *Deep unsupervised learning using nonequilibrium thermodynamics*, 2015. arXiv: 1503.03585 [cs.LG].

[268] G. Tevet, S. Raab, B. Gordon, Y. Shafir, D. Cohen-or, and A. H. Bermano, "Human motion diffusion model", in *The Eleventh International Conference on Learning Representations*, 2023.

[269] M. Zhang *et al.*, "Motiondiffuse: Text-driven human motion generation with diffusion model", *arXiv preprint arXiv:2208.15001*, 2022.

[270] J. Kim, J. Kim, and S. Choi, "Flame: Free-form language-based motion synthesis & editing", *arXiv preprint arXiv:2209.00349*, 2022.

[271] J. Tseng, R. Castellon, and C. K. Liu, "Edge: Editable dance generation from music", *arXiv preprint arXiv:2211.10658*, 2022.

[272] S. Raab, I. Leibovitch, G. Tevet, M. Arar, A. H. Bermano, and D. Cohen-Or, *Single motion diffusion*, 2023. arXiv: 2302.05905 [cs.CV].

[273] H.-K. Kao and L. Su, "Temporally guided music-to-body-movement generation", in *Proceedings of the 28th ACM International Conference on Multimedia*, ser. MM '20, Acm, Oct. 2020.

[274] N. Yalta, S. Watanabe, K. Nakadai, and T. Ogata, *Weakly supervised deep recurrent neural networks for basic dance step generation*, 2019. arXiv: 1807.01126 [cs.LG].

[275] J. Li *et al.*, *Learning to generate diverse dance motions with transformer*, 2020. arXiv: 2008.08171 [cs.CV].

[276] L. Siyao *et al.*, "Bailando: 3d dance generation via actor-critic gpt with choreographic memory", in *Cvpr*, 2022.

[277] L. Siyao *et al.*, "Bailando++: 3d dance gpt with choreographic memory", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 12, pp. 14 192–14 207, 2023.

[278] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black, "AMASS: Archive of motion capture as surface shapes", in *International Conference on Computer Vision*, Oct. 2019, pp. 5442–5451.

[279] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1325–1339, 2014.

[280] C.-M. University, *Carnegie-mellon motion capture database*, http://mocap.cs.cmu.edu, Accessed: 2024-01-01, 2010.

[281] S. F. University and N. U. of Singapore, *Sfu motion capture database*, https://mocap.cs.sfu.ca, Accessed: 2024-01-01, 2017.

[282] A. S. Inc., *Mixamo*, https://www.mixamo.com, Accessed: 2024-01-01, 2018.

[283] J. Lee, S. Kim, and K. Lee, *Listen to dance: Music-driven choreography generation using autoregressive encoder-decoder network*, 2018. arXiv: 1811.00818 [cs.MM].

[284] S. Tsuchida, S. Fukayama, M. Hamasaki, and M. Goto, "Aist dance video database: Multi-genre, multi-dancer, and multi-camera database for dance information processing", in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019*, Delft, Netherlands, Nov. 2019.

[285] W. Zhuang, C. Wang, J. Chai, Y. Wang, M. Shao, and S. Xia, "Music2dance: Dancenet for music-driven dance generation", vol. 18, no. 2, Feb. 2022.

[286] G. Stevens, *Psychophysics: Introduction to its perceptual, neural, and social prospects*. John Wiley & Sons, 1975.

[287] M. Heydari, F. Cwitkowitz, and Z. Duan, "Beatnet: Crnn and particle filtering for online joint beat downbeat and meter tracking", in *22th International Society for Music Information Retrieval Conference, ISMIR*, 2021.

[288] M. Plakal and D. Ellis, *Yamnet*, `https://github.com/tensorflow/models/tree/master/research/audioset/yamnet`, 2020.

[289] J. Punks, *Jingle punks*, `https://www.jinglepunks.com/`, Accessed: 2024-01-01.

[290] Rokoko, *Capture your body's motion in real-time with smartsuit pro ii*, 2023.

[291] J. Zhang *et al.*, "Skinned motion retargeting with residual perception of motion semantics & geometry", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 13 864–13 872.

[292] K. Aberman, P. Li, D. Lischinski, O. Sorkine-Hornung, D. Cohen-Or, and B. Chen, "Skeleton-aware networks for deep motion retargeting", *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, p. 62, 2020.

[293] N. EMEA, *Newland fm3080 hind*, `https://www.newland-id.com/en/products/stationary-scanners/fm3080-hind/`, Accessed: 2024-01-01.

[294] A. Gopalakrishnan, A. Mali, D. Kifer, C. L. Giles, and A. G. Ororbia, *A neural temporal model for human motion prediction*, 2019. arXiv: `1809.03036 [cs.CV]`.

[295] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "SMPL: A skinned multi-person linear model", *ACM Trans. Graphics (Proceedings of SIGGRAPH Asia)*, vol. 34, no. 6, 248:1–248:16, Oct. 2015.

[296] X. Yi *et al.*, Eds., *Sampling-Bias-Corrected Neural Modeling for Large Corpus Item Recommendations*, 2019.

[297] D. Michelsanti, *Audio-visual speech enhancement based on deep learning*, PhD supervisor: Professor Zheng-Hua Tan, Aalborg University, Denmark Assistant PhD supervisor: Professor Jesper Jensen, Aalborg University, Denmark & Oticon A/S, Denmark, 2021.

[298] D. Liu, Q. Mao, L. Gao, Q. Ren, Z. Chen, and M. Dong, "Te-kws: Text-informed speech enhancement for noise-robust keyword spotting", in *Proceedings of the 31st ACM International Conference on Multimedia*, ser. MM '23, Ottawa ON, Canada: Association for Computing Machinery, 2023, pp. 601–610, ISBN: 9798400701085.

[299] W. Zhang, X. Chang, C. Boeddeker, T. Nakatani, S. Watanabe, and Y. Qian, "End-to-end dereverberation, beamforming, and speech recognition in a cocktail party", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 3173–3188, 2022.

[300] C. Rodriguez-Pardo, H. Dominguez-Elvira, D. Pascual-Hernandez, and E. Garces, *Umat: Uncertainty-aware single image high resolution material capture*, 2023. arXiv: 2305.16312 [cs.CV].

[301] Y. Hu, C. Chen, R. Li, Q. Zhu, and E. S. Chng, *Gradient remedy for multi-task learning in end-to-end noise-robust speech recognition*, 2023. arXiv: 2302.11362 [eess.AS].

[302] M. Chiquier, C. Mao, and C. Vondrick, *Real-time neural voice camouflage*, 2022. arXiv: 2112.07076 [cs.SD].

[303] E. A. Inc., *Ea*, https://www.ea.com, Accessed: 2024-01-01.

[304] I. Meta Platforms, *Meta*, https://about.meta.com, Accessed: 2024-01-01.

[305] S. Starke, I. Mason, and T. Komura, "Deepphase: Periodic autoencoders for learning motion phase manifolds", *ACM Trans. Graph.*, vol. 41, no. 4, Jul. 2022.

[306] I. Mason, S. Starke, and T. Komura, "Real-time style modelling of human locomotion via feature-wise transformations and local motion phases", *Proc. ACM Comput. Graph. Interact. Tech.*, vol. 5, no. 1, May 2022.

[307] S. Starke, P. Starke, N. He, T. Komura, and Y. Ye, "Categorical codebook matching for embodied character controllers", *ACM Trans. Graph.*, vol. 43, no. 4, Jul. 2024.

[308] B. Jacob *et al.*, *Quantization and training of neural networks for efficient integer-arithmetic-only inference*, 2017. arXiv: 1712.05877 [cs.LG].

[309] M. A. Carreira-Perpinan and Y. Idelbayev, ""learning-compression" algorithms for neural net pruning", in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8532–8541.

[310] A. Mishra and D. Marr, *Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy*, 2017. arXiv: 1711.05852 [cs.LG].

Appendix A: Metrics for Silent Interval Detection

Here we provide the details of the metrics used for evaluating our silent interval detection in Part I (i.e., results in Table 5.1 from Section 5.3). Detecting silent intervals is a binary classification task, one that classifies every time segment as being silent (i.e., a positive condition) or not (i.e., a negative condition). Recall that the confusion matrix in a binary classification task is as follows:

Table A.1: **Confusion matrix.**

		Actual	
		Positive	Negative
Predicted	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

In our case, we have the following conditions:

- A true positive (TP) sample is a correctly predicted silent segment.
- A true negative (TN) sample is a correctly predicted non-silent segment.
- A false positive (FP) sample is a non-silent segment predicted as silent.
- A false negative (FN) sample is a silent segment predicted as non-silent.

The four metrics used in Table 5.1 follow the standard definitions in statistics, which we review here:

$$\begin{aligned}
\text{precision} &= \frac{N_{\mathsf{TP}}}{N_{\mathsf{TP}} + N_{\mathsf{FP}}}, \\
\text{recall} &= \frac{N_{\mathsf{TP}}}{N_{\mathsf{TP}} + N_{\mathsf{FN}}}, \\
\text{F1} &= 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}, \text{ and} \\
\text{accuracy} &= \frac{N_{\mathsf{TP}} + N_{\mathsf{TN}}}{N_{\mathsf{TP}} + N_{\mathsf{TN}} + N_{\mathsf{FP}} + N_{\mathsf{FN}}},
\end{aligned} \tag{A.1}$$

where N_{TP}, N_{TN}, N_{FP}, and N_{FN} indicate the numbers of true positive, true negative, false positive, and false negative predictions among all tests. Intuitively, *recall* indicates the ability of correctly finding all true silent intervals, *precision* measures how much proportion of the labeled silent intervals are truly silent. *F1* score takes both precision and recall into account, and produces their harmonic mean. And *accuracy* is the ratio of correct predictions among all predictions.

Appendix B: Derivations for Equations from Real-time Speech Denoising

Here we provide detailed derivation for equations (7.1) and (7.2) in Part I, i.e., the accumulated delay for the fixed-size and dynamic sliding window approaches detailed in Section 7.2.1. Audio denoising can be divided into three steps: audio recording, denoised network processing and denoised audio playback. We show the fixed-size and dynamic sliding window examples in Figure B.1 and Figure B.2 respectively. In the figures, our timeline is from left to right.

We use block to indicate a certain time span. Each rectangle in the figures is regarded as a block. The blank area represents the waiting time between two neighbor blocks. We use R_i, N_i and P_i to indicate the recording, processing and playback block, use $s(block)$ and $e(block)$ to indicate the start time and end time of each block. In particular, we define t_i as the network processing time for a processing block, i.e. $t_i = e(N_i) - s(N_i)$, and d_i as the total delay of a certain audio block i. d_i is the time from the moment of receiving to the moment of outputting , i.e. $d_i = s(P_i) - s(R_i)$.

We first claim some facts for sliding window approach:

1. There is no blank area for all recording block, i.e. $s(R_i) = e(R_{i-1})$ for any $i > 1$.

2. The network will process current audio immediately when there exists unprocessed audio and the previous audio has been processed, i.e. $s(N_i) = \max\left(e(R_i),\ e(N_{i-1})\right)$, for any $i > 1$.

3. The player will play current audio immediately when there exists unplayed audio and the previous audio has been played, i.e. $s(P_i) = \max\left(e(N_i),\ e(P_{i-1})\right)$, for any $i > 1$.

4. The recording block and the playback contains the audio of the same length, i.e. $e(R_i) - s(R_i) = e(P_i) - s(P_i)$ for any $i > 0$. In particular, $e(R_i) - s(R_i) = L$ for any i in fixed-size sliding window approach. Here L is the fixed window size.

5. Over time, the total delay will be accumulated, i.e. $d_i \leq d_{i-1}$ for any $i > 1$.

The following analysis shows the theoretical delay d_i, which in practices is smaller than the actual measured audio playback delay (D_A) as introduced in Chapter 7.

B.1 Fixed-size sliding window

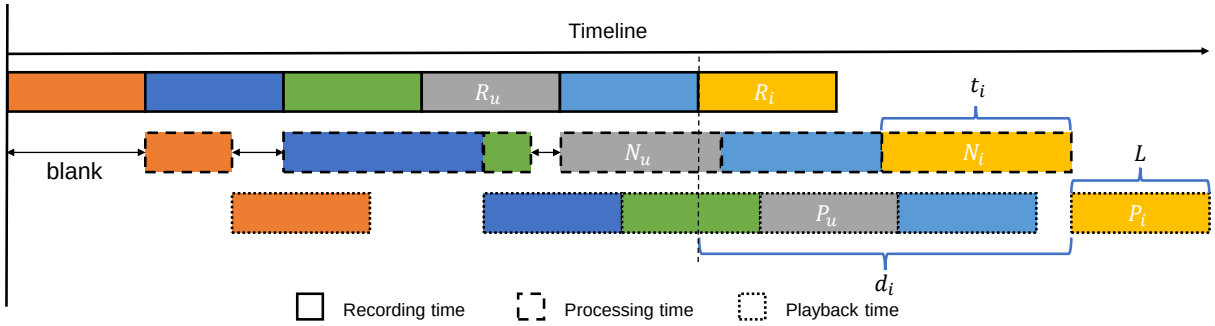


Figure B.1: **Fixed-size sliding window.**

To calculate d_i in fixed-size sliding window approach, we first find the previous processing blank area, and the processing block N_u right after that blank. Thus, by fact 2, we have $e(R_u) = s(N_u)$. There are $i - u$ blocks with length L between R_i and R_u, thus we have $s(R_i) = S(R_u) + (i - u)L$ and $e(P_{i-1}) = e(P_{u-1}) + (i - u)L$. By fact 3, we can derive,

$$
\begin{aligned}
d_i &= s(P_i) - s(R_i) \\
&= \max\left(e(N_i),\ e(P_{i-1})\right) - s(R_i) \\
&= \max\left(s(N_u) + \sum_{k=u}^{i} t_k,\ e(P_{u-1}) + (i-u)L\right) - s(R_i) \\
&= \max\left(e(R_u) + \sum_{k=u}^{i} t_k,\ e(P_{u-1}) + (i-u)L\right) - \left(s(R_u) + (i-u)L\right) \\
&= \max\left(s(R_u) + L + \sum_{k=u}^{i} t_k - s(R_u) - (i-u)L,\ e(P_{u-1}) - s(R_u)\right)
\end{aligned}
$$

$$= \max\left(2L + \sum_{k=u}^{i}(t_k - L),\ e(P_{u-1}) - e(R_{u-1})\right)$$
$$= \max\left(2L + \sum_{k=u}^{i}(t_k - L),\ d_u\right).$$

Through recursion, we can get,

$$d_i = 2L + \max_{1 \le p \le q \le i} \sum_{k=p}^{q}(t_k - L). \tag{B.1}$$

B.2 Dynamic sliding window

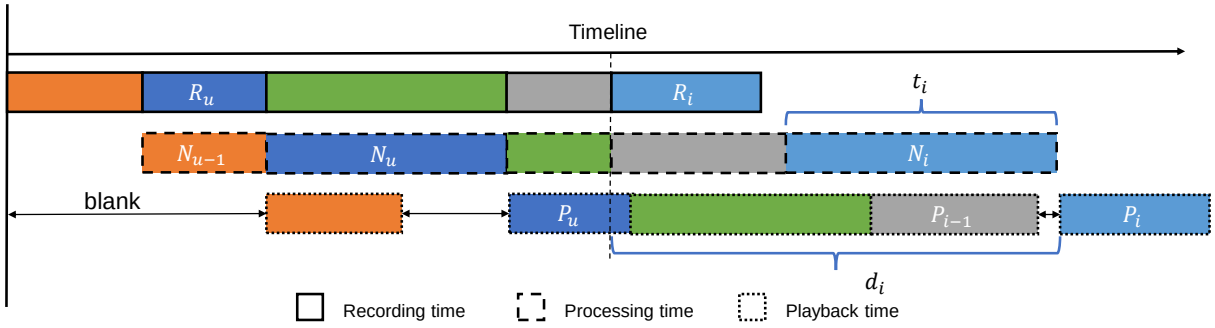


Figure B.2: **Dynamic sliding window.**

For the dynamic sliding window approach, the recording time is always equal to the previous processing time, i.e. $N_{i-1} = R_i = P_i$ for any $i > 1$. We have three cases that need to be analyzed.

Case 1: If there is no blank area between P_i and P_{i-1}, for any $i > 1$,

$$d_i = s(P_i) - s(R_i)$$
$$= e(P_{i-1}) - e(R_{i-1})$$
$$= d_{i-1}$$

Case 2: If there is a blank area before P_i, and $i > 1$, then we can always find the previous playback blank area and the playback block P_u right after the blank area. By fact 3, we can derive,

$$
\begin{aligned}
d_i &= s(P_i) - s(R_i) \\
&= \max\left(e(N_i),\ e(P_{i-1})\right) - s(R_i) \\
&= \max\left(s(N_u) + \sum_{k=u}^{i} t_k,\ s(P_u) + \sum_{k=u}^{i-1}(e(P_k) - s(P_k))\right) \\
&\quad - \left(s(R_{u+1}) + \sum_{k=u+1}^{i-1}(e(P_k) - s(P_k))\right) \\
&= \max\left(s(N_u) + \sum_{k=u}^{i} t_k,\ s(P_u) + \sum_{k=u}^{i-1}(e(N_{k-1}) - s(N_{k-1}))\right) \\
&\quad - \left(s(R_{u+1}) + \sum_{k=u+1}^{i-1}(e(N_k - 1) - s(N_k - 1))\right) \\
&= \max\left(s(N_u) + \sum_{k=u}^{i} t_k,\ s(P_u) + \sum_{k=u}^{i-1} t_{k-1}\right) - \left(s(R_{u+1}) + \sum_{k=u+1}^{i-1} t_{k-1}\right) \\
&= \max\left(\sum_{k=u}^{i} t_k,\ t_u + \sum_{k=u}^{i-1} t_{k-1}\right) - \sum_{k=u+1}^{i-1} t_{k-1}\right) \\
&= \max\left(t_{i-1} + t_i,\ t_{u-1} + t_u\right).
\end{aligned}
\tag{B.2}
$$

Through recursion, we can get,

$$
d_i = \max_{k \le i}\left(t_{k-1} + t_k\right), \quad i \ge 2. \tag{B.3}
$$

Case 3: When $i = 1$, $d_1 = L_0 + t_1$, where L_0 is an initial window size to start the denoising process at the beginning. In summary, we have,

$$
d_i = \begin{cases} L_0 + t_1, & i = 1 \\ \max\limits_{k \le i}\left(t_{k-1} + t_k\right), & i \ge 2 \end{cases} \tag{B.4}
$$