Interactive Methods for Computational Sound Design, Control, and Calibration

（コンピュータによるサウンドデザイン,

コントロール, キャリブレーションのための対話的手法）

by

Kazuhiko Yamamoto

山本和彦

A Doctor Thesis

博士論文

Submitted to
the Graduate School of the University of Tokyo
on June 9, 2017
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Information Science and Technology
in Computer Science

Thesis Supervisor: Takeo Igarashi 五十嵐健夫
Professor of Computer Science

Abstract

Computational sound plays an important role in digital computer entertainment. Traditionally, fixed set of sound data is prepared beforehand and the system simply plays these sounds at runtime. However, as the demand for high quality virtual experience increases, there is an increase need for synthesizing appropriate sound interactively responding to user control at runtime. Unfortunately, technology development in sound effects is lagging behind compared to those for visual effects and there remain various difficulties to bring interactivity in sound processing. For this reason, according to the increases of its interactivity, it arises a problem that the efforts of the artist for designing a lot of sounds one by one would be increased. On the other hand, the user would be dissatisfied with a control device which has only a limited degree of freedom due to the difficulty for manipulating the sound. Furthermore, although it becomes important to render the sounds with appropriately calibrating to the individual specifications or tastes of each user, there is a problem that it is difficult for the designer to gather such the information beforehand. To address these problems, this thesis presents methods to bring higher interactivity to computational sound by reducing computational cost and user's operation cost.

First, we discuss interactivity in the design of physically based sound. Although a sound designer needs to design appropriate material properties for physically based sound, exploration of the design space of physical parameters directly is unintuitive and difficult. This thesis addresses this problem by an example-based user interface that optimizes inversely the material from a few example inputs of sound clips by the user. Second, we discuss interactivity in the control of singing voice synthesizer. A user needs to continuously input lyric and melody to use singing voice synthesizer in improvisational performance. However, inputting them both simultaneously is difficult with standard input devices. This thesis addresses this problem by predicting latent lyrics desired by the user in realtime from the input of a standard control device. Finally, we discuss interactively in calibration of 3d spatial sound. Although it is necessary to calibrate a spatial audio system to adopt them for a specific user, traditional calibration procedure is expensive and time consuming. This thesis addresses this problem by adaptation of the system for a specific user using an user interface that requires simple pairwise comparison tasks.

To achieve these goals, this thesis builds a computational model behind each task in precomputation, and exploits the model to reduce computational cost or user's operation cost at runtime. Specifically, in the design of physically based sound, we present dramatically fast vibrational analysis using precomputed mesh simplification algorithm using machine learning and hierarchical component mode synthesis, which allows material optimization at interactive rate. In the control of singing voice synthesizer, we present a method to estimate latent lyrics as higher DoF parameters from the input of lower DoF control device using machine learning of lyrics dataset at precomputation phase. Finally, in the calibration of 3d spatial audio, we propose a machine learning model that allows adaptation of the system to a specific user using individual and non-individual factors of dataset which are extracted at precomputation. This thesis describes the details of each method and presents the results of numerical evaluations and empirical studies with end users. These methods, in which the ideas are based on the improvements the interactivity of runtime applications by user interfaces using precomputation, have generality and can be widely applied to similar problems in other domains.

論文要旨

インタラクティブなデジタルコンテンツにおいて音の利用は重要な役割を担う. 従来は, あらかじめ用意された限られた数のサウンドファイルを実行時にユーザのインタラクションに対して鳴らす, というのが一般的な音の利用方法であった. しかし, デジタルコンテンツへのインタラクティブ性への要求が高まるにつれて, ユーザのあらゆるインタラクションに対して適切な音で反応することが求められてきているが, それを実現するための音の利用技術は映像技術と比較して非常に遅れている. 現状, コンテンツのインタラクティブ性が高まるほど, アーティストにとっては多くの音をデザインする労力が増し, 一方ユーザにとっては, 限られた自由度の操作手段で多くの要素を制御することが困難であること, に対する不満が高まっていく. また, 個人差のあるユーザそれぞれの知覚や好みに合わせてキャリブレーションした上で音をレンダリングすることの重要性が高まるが, そうした情報をデザイナーがあらかじめ収集しておくことは困難である, という問題もある. これらを解決するために, 本論文では, コンピュータの計算コストとユーザの操作コスト両方を大きく軽減することによって, デジタルコンテンツにおける音の利用のインタラクティブ性を高める手法を提案する.

第一の手法では, 物理ベースのサウンドデザインについて論じる. 物理ベースのサウンドをデザインするためには, デザイナーは適切な材質という直感的ではない物理的なパラメータを設定する必要があり, これは非常に困難であるという問題がある. そこで本論文では, 例示ベースのユーザインタフェースでデザイナーの所望する音を入力することで逆に材質パラメータを最適化し, この問題を解決する. 第二の手法では, 歌声合成のリアルタイムコントロールを例に挙げる. ユーザは歌詞とメロディ両方を同時に入力する必要があるが, これは従来の入力デバイスでは非常に困難である. 本論文では, 従来通りの入力デバイスからの入力から, ユーザの想定した歌詞を, リアルタイム予測をすることで, この問題を解決する. 第三の手法では, 3 次元音響のユーザへのキャリブレーションについて挙げる. 3 次元音響を再生するためには, ユーザに対して適切なキャリブレーションをおこなう必要がある が, 従来これは特別な機材や膨大な時間が必要であり困難であった. そこで本論文では簡単な比較タスクのみによるユーザインタフェースでシステムの個人適応をおこない, この問題を解決する.

これらを実現するために, 本論文では, 事前計算によってこれらのタスクを表現する計算モデルを構築することで, 実行時のコンピュータの計算コストやユーザの操作コストを大幅に軽減するアルゴリズムを提案する. 具体的には, 物理ベースサウンドのデザインをするために, 事前の機械学習によるメッシュ簡略化と階層的モーダル分解を利用した高速な振動解析を提案し, インタラクティブな速度での材質最適化を可能にする. また, 歌声合成のリアルタイム演奏を可能にするために, 歌詞データを事前に機械学習しておくことによって, 低自由度の入力からの高次情報のリアルタイム予測をおこなう. さらに3次元音響のユーザへのキャリブレーションをおこなうために, 事前にデータセットから個人性と非個人性を分離して個人適応に利用することのできる機械学習モデルを提案する. 本論文では, それぞれの手法について詳しく述べるとともに, 数値実験やユーザスタディによってその有効性を実証する. 本論文で提案しているコンピュータによる事前計算を利用したユーザインタフェースによって, アプリケーション実行時のイン

タラクティブ性を高める手法は, 広く他の分野にも応用することが可能である.

Acknowledgement

I would like to thank all those who helped me complete this work. First and foremost, I would like to express my deepest appreciation to my supervisor, Takeo Igarashi, who has supervised me through my doctoral research career. It has been my great honor to work under his supervision. I would like to thank Daisuke Sakamoto for his advice on my research and on life in general and for discussions of nascent research ideas. I am also very grateful to my thesis committee members, Issei Sato, Reiji Suda, Toshiya Hachisuka, and Kazuyoshi Yoshii for providing useful suggestions for improving this thesis. I thank past and present members of Igarashi laboratory, especially, Yuki Koyama, Kazuyo Mizuno, Seung-Tak Noh, Hikaru Ibayashi, Kazutaka Nakajima, Haoran Xie, Hironori Yoshida, Nobuki Yoda, Hiroaki Mikami, Yosuke Takami, Kai Shidachi, Ryohei Suzuki, Takahito Hamanaka, Makoto Nakajima, Naoki Sasaki, Koumei Fukahori, and Ko Mizoguchi, who spent a long time with me discussing interesting topics, as well as giving me a comprehensive introduction to their own favorite research topics. A half year of my doctoral studies were supported by ACT-I Program of Japan Science and Technology Agency. I would like to thank my advisor, Yutaka Matsuo, and the research supervisor, Masataka Goto for his advice on perceptual based 3D audio calibration project. I would like to thank my father and mother, Kazuo and Michiyo Yamamoto. for their steadfast support of my decisions in life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computational sound plays an important role in digital computer entertainment. Traditionally, fixed set of sound data is prepared beforehand and the system simply plays these sounds at runtime. However, as the demand for high quality virtual experience, there is an increase need for synthesizing appropriate sound interactively responding to user control at runtime. Unfortunately, technology development in sound effects is lagging behind compared to those for visual effects and there remain various difficulties to bring interactivity in sound processing. In this thesis, to improve the interactivity of computational sound and allow to use computational sounds techniques in real scene, we discuss methods for addressing these problems from three aspects of computational sound production: *Design*, *Control*, and *Calibration* (Figure 1.1).

1.1 Background

In this section, we describe the background of the problems addressed in this thesis. Specifically, this thesis treats three significant problems in computational sound: physically based sound design, High DoFs parameter control of singing voice synthesizer, and calibration of 3D audio spatialization for a specific user.

First, for designing computational sound, we need to prepare appropriate sounds for each user's interaction. For example, in VR game, a player interacts with various object in various way. The player would expect different sounds by different interactions. However, to achieve such interactions, traditional sound design procedure requires a large efforts of the artist for designing a lot of sounds one by one. This is impractical in actual scene. As a result, this limitation makes the virtual experience of sounds to be quite limited. As a solution for this problem, physically-based sound synthesis techniques known as sound rendering [104] have been proposed by the graphics community in the last decade. These methods reduce the effort of manipulating a large number of audio samples for sound designers. However, although input parameter for these techniques is the material distribution inside the model,

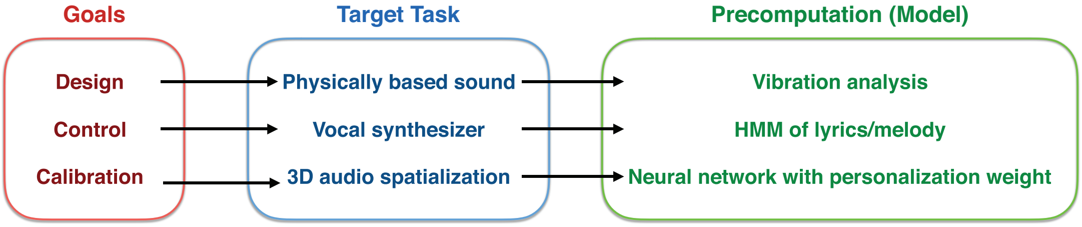


Figure 1.1: The goal of this thesis is three target tasks in computational sound to be capable to design, control, and calibration by proposing novel user interfaces using precomputation method.

designing the internal material distribution properly so that the system produces a specific sound as a result of physical simulation is difficult even for professional designers.

Second, for controlling computational sound by a user, we need to manipulate a large set of parameters in realtime. However, the user would be dissatisfied with a control device which has only a limited degree of freedoms (DoFs) due to the difficulty for manipulating the sound. As an example of computational sound that requires to manipulate such many parameters in realtime, we focus on a singing voice synthesizer [67] in this thesis. To control a singing voice synthesizer in realtime, we need to input lyrics and melodies simultaneously. Generally, this is quite difficult because a song lyric has large DoFs compared to standard input devices (e.g., piano keyboard, QWERTY keyboard).

Finally, we need to calibrate the sound rendering system for each user. Although it becomes important to render the sounds with appropriately calibrating to the individual specifications or tastes of each user, there is a problem that it is difficult for the designer to gather such the information beforehand. Specifically, calibrating 3D audio spatialization for each user is an important issue. The human auditory system perceives the directions of incoming sounds using both ears. According to the direction from which a sound arrives to the head, an arrival time difference to the left and right ears can be determined. In addition, the sound is intricately diffracted by the shape of the person's head and ears. This diffraction effect depends on the frequency and incoming direction of the sound. Therefore, the spectrums of the sounds that arrive at each ear are modified. We can recognize the localization of the sound by these sound modifications. These two-channel transforms of the spectrums can be represented as finite impulse response filters and are called human-related transfer functions (HRTFs). Because of this, three-dimensional (3D) spatialization of sounds in virtual environments (e.g., VR and games) requires HRTFs to reproduce incoming sounds from various directions using a two-channel headphones. However, HRTFs are highly specific to individuals because they depend considerably on the shape of the user's ears and head. We know that inappropriate HRTFs can lead to improper localization of the sound source accompanied by an unexpected equalization of the timbre. Because of this, we must essentially measure the specific HRTF for each user. The measurement procedure requires special equipment, including an anechoic chamber, as well as time-consuming and tedious efforts of the user. Thus, using specific HRTFs for each end user has been impractical.

1.2 Contributions

To address the issues described above, this thesis presents methods to bring higher interactivity to computational sound by reducing computational cost and user's operation cost. First, we discuss interactivity in the design of physically based sound. Although a sound designer needs to design appropriate material properties for physically based sound, the exploration of the design space of physical parameters directly is unintuitive and difficult. This thesis addresses this problem by inversely optimizing the material from a few example inputs of sound clips by the user. Second, we discuss interactivity in the control of a singing voice synthesizer. A user needs to continuously input lyric and melody to use a singing voice synthesizer in improvisational performance. However, inputting both parameters simultaneously is difficult with standard low DoFs input devices. This thesis addresses this problem by predicting latent lyrics desired by the user in realtime from the input of a standard control device. Finally, we discuss interactively in calibration of 3d spatial sound. Although it is necessary to calibrate a spatial audio system for a specific user, traditional calibration procedure is expensive and time consuming. This thesis addresses this problem by adaptation of the system for a specific user using simple pairwise comparison tasks.

To achieve these goals, this thesis builds a computational model behind each task in precomputation, and exploits the model to reduce computational cost or user's operation cost at runtime. Specifically, in the design of physically based sound, we present a material optimization method for example-based framework, and a fast vibrational analysis using precomputed mesh simplification algorithm using machine learning and hierarchical component mode synthesis. This allows practical workflow of physically-based sound design. In the control of singing voice synthesizer, we present a method to estimate latent lyrics as higher DoF parameters from the input of lower DoF control device using machine learning of lyrics dataset at precomputation phase. Finally, in the calibration of 3d spatial audio, we propose a machine learning model that allows adaptation of the system to a specific user using individual and non-individual factors of dataset which are extracted at precomputation. These methods, that improves the interactivity of applications at runtime with precomputation, have generality and can be widely applied to similar problems in other domains.

1.3 Outline

In this dissertation, we introduce three user interfaces using precomputation to reduce the computational cost or user's operation cost for achieving interactive sound design, control, and calibration as follows.

First, we propose an example based framework for designing physically based sound of a 3D model. Our method enables the user to design the sound of an object, which can respond various user's interaction, without directly specifying the physical parameters. The user first provides a 3D surface model to the system as input. Next, the user selects a few sample positions on the model surface, and assigns corresponding sound clips that define the target sounds to be rendered when the positions are struck. The system then optimizes the material distribution inside the model so that physically-based sound simulations yield the expected sounds. This is similar to the traditional sound design procedure and the designer does not need to learn new skills for designing physically based sound. However, modal analysis that is required for obtaining the vibrational property of the object at an iteration in our optimization is a prohibitively expensive. To address this, we introduce a dramatically fast vibrational analysis method by using precomputed mesh simplification algorithm using machine learning and hierarchical component mode synthesis (HCMS). The mesh simplification algorithm learns accuracy preserving fine to coarse mapping of finite element mesh at precomputation phase, and uses it as online mesh simplify function at runtime. On the other hand, the HCMS decomposes the finite element mesh into many computationally-light subdomains based on precomputed mesh segmentation, and merges them. In these methods, we reduces the user's design cost for improving the interactivity of computational sound (Figure 1.1: Top).

Second, to control a large set of parameters of a singing voice synthesizer in realtime, we present a method to estimate latent lyrics from the input of lower DoFs control device using machine learning of lyrics dataset at precomputation phase. This allows the user to control higher DoFs parameters (in singing voice synthesizer, lyrics and melodies of a song) with lower DoFs input devices (e.g., piano keyboard). At the performance step, the user simultaneously inputs vowel sequences using a vowel keyboard and melodies using a standard musical keyboard. The system estimates the plausible lyrics from the vowel sequences and synthesizes singing voice sounds. Note that the system does not use a melody sequence for estimation. Specifically, our system automatically finds a portion of the predefined lyrics whose vowel sequence matches well with the vowel sequence being input by the player. We use a Hidden Markov model (HMM) for this vowel sequences to lyrics alignment. This enables realtime control of a singing voice synthesizer using a standard piano keyboard. Our system also allows the user to modify the melodies of a song freely and to pick an arbitrary portion of predefined lyrics during a live performance. In this method, we reduce the user's operation cost for improving the interactivity of sound control (Figure 1.1: Middle).

Finally, to calibrate 3D audio spatialization in a virtual environment for a specific user, we present a machine learning model that allows adaptation of the system to a specific user using individual and non-individual factors of dataset which are extracted at precomputation. Our algorithm requires neither special equipment nor tedious measurement procedures. The user only needs to provide several feedbacks rating A or B pairwise comparisons of test signals provided by the system based on his or her individual perceptions during calibration. Our algorithm uses a novel adaptive variational AutoEncoder[71, 112] trained with a publicly available HRTFs data set at precomputation phase. During training, it decomposes HRTFs in the data set into factors based on individual users and the rest. During calibration, our adaptive variational AutoEncoder generates individualized HRTFs for a new user by blending several individualities with personalization weight in nonlinear space. An advantage of this two steps algorithm is that it does not require optimizations for all the spherical directions around the head because the personalization weight is shared within all the directions, which has been not addressed in previous studies. This dramatically reduces the user's efforts at obtaining specific HRTFs. In this method, we reduce the measurement cost for rendering appropriate spatial sounds which is essential in highly interactive digital content (Figure 1.1: Bottom).

1.4 Thesis Overview

The remainder of this dissertation is organized as follows:

1. In Chapter 2, we review the existing techniques for improving the runtime interactivity by a user interface using precomputation, and clarify the differences from our approaches. We focus on significant two motivations for using precomputation methods for a user interface: accelerating runtime computation, and making inferences at runtime.

2. In Chapter 3, we present an example based method for designing physically based sound. We also describe a method for achieving responsive speed for vibrational analysis of a 3D model, which enables material optimization of the model at an interactive rate. We validate the effectiveness of the algorithm via several benchmarks and actual sound design.

3. In Chapter 4, we discuss a method for controlling a singing voice synthesizer in realtime with a standard piano keyboard. We describe how high DoFs parameters can be controlled using low DoFs input device. We also demonstrate the feasibility of our algorithm to control higher DoFs parameters via several user studies.

4. In Chapter 5, In Chapter 5, we present a fully perceptual based method to calibrate 3D audio spatialization for a specific user. We describe two steps machine learning technique to analyze the HRTF dataset that extracts individual factors at precomputation phase, and how efficiently they are optimized at runtime. We validate the effectiveness of the algorithm via cross validations, simulations and a user study.

5. In Chapter 6, we conclude the thesis. We briefly summarize how the interactivity of computational sound were improved by presented user interfaces using precomputation. Finally, we describe possible directions for future research and applications.

1.5 Publication

The following is a list of publications from which this thesis was derived:

1. Our example-based system for designing physically-based sound was published as "Interactive Physically-Based Sound Design of 3D Model using Material Optimization" in ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA) 2016 in Zurich, Switzerland, in collaboration with Takeo Igarashi from the University of Tokyo.

2. Our realtime interface system for controlling singing voice synthesizer was published as "LiVo: Sing a Song with a Vowel Keyboard" in ACM New Interfaces for Musical Expression (NIME) 2015 in Baton Rouge, USA, in collaboration with Takeo Igarashi from the University of Tokyo.

3. Our calibration system for optimizing 3d audio spatialization for a specific user was published as "Fully Perceptual-Based 3D Spatial Sound Individualization with an Adaptive Variational AutoEncoder" in ACM Transaction on Graphics (SIGGRAPH Asia) 2017 in Bangkok, Thailand, in collaboration with Takeo Igarashi from the University of Tokyo.

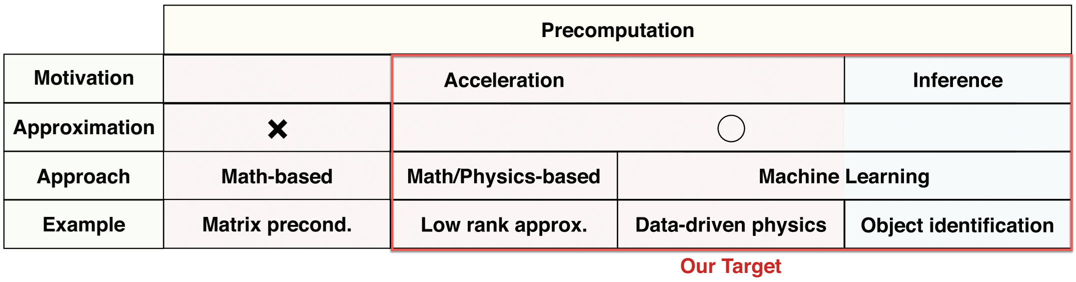
Chapter 2

Related Work

The main technical contribution of this thesis is to achieve interactive computational sound applications by presenting three novel user interfaces using precomputation algorithms. The precomputation splits the overall computations into two parts, the precomputation phase and runtime phase, so that performance is improved at runtime. Precomputation itself is not a new approach. To provide a context for our study, this chapter reviews previous researches related to precomputation and how these methods are used for interactive applications including computer human interface.

There are two types of significant motivation for using precomputation (Figure 2.1); (i) accelerating runtime computation, (ii) making inferences at runtime. In addition, the methods used for the former motivation can be categorized by three approaches: (a) mathematically based approach, (b) mathematically or physically based approximation approach, and (c) machine learning based-approximation approach. The first approach mathematically transforms each original target problem. Thus, we can obtain exactly same solution as the original problem with faster computation. On the other hand, we can not obtain exact solution by the other two approaches, because these approaches use approximations for accelerating the algorithms, but they are usually faster than the first approach and also can be applied to the problems that is difficult to analytically transform for precomputation. In another perspective, we call the former two approaches as model-based approach in contrast to the last approach which can be also called as data driven approach. Note that the first approach is beyond the scope of this thesis. The precomputation method used for our first method in this thesis accelerates the runtime computation using a combination of physically-based and machine learning-based approximations.

On the other hand, the methods for the second motivation mostly use several kinds of machine learning techniques. Such machine learning techniques perform training the model by datasets at precomputation phase and make inferences for new data using the trained model at runtime. Machine learning is essentially equivalent to a function approximation. Once we can design a model to train, the model can be optimized via providing a dataset. The



	Precomputation			
Motivation		Acceleration		Inference
Approximation	✗	○		
Approach	Math-based	Math/Physics-based	Machine Learning	
Example	Matrix precond.	Low rank approx.	Data-driven physics	Object identification

Our Target

Figure 2.1: A classification of precomputation methods. Our targets in this thesis are accelerating runtime computation by a combination of physics based and machine learning based approximations (the first method), and making an inference at runtime by machine learning (the second and third methods).

dataset can be both real observation and synthetic (simulation) data. Thus, it is useful when it is difficult to determine the model parameters because it is expensive to obtain due to some reasons (e.g., high computational cost, blackbox function). The precomputation methods used for our second and third methods in this thesis belong to this group of motivation and use hidden Markov model and deep neural network model respectively. In the following sections, we focus on two significant motivations for using precomputation as described above.

2.1 Accelerating Runtime Computation

A significant motivation for using precomputation is to accelerate an application at runtime. Various studies compute the expensive operations for each target at precomputation phase, and use the results for interactive applications at run- time. There are roughly two kinds of such precomputation approach; model-based approach and data driven approach.

2.1.1 Model-Based Approach

Model-based approaches precompute the computationally expensive informations for accelerating runtime application based on mathematical or physics theory. Additionally, this approach can be categorized by whether it uses approximation or not.

Precomputation for Linear Systems

Linear system solves are required for many interactive applications (e.g., Poisson equation for incompressible fluid simulation [12], and the ordinary differential equations for finite element deformable body simulation [116]). For accelerating linear systems solves $\mathbf{A}x = b$, where $\mathbf{A}$ is a matrix and x, b are vectors, there are many precomputation methods based on mathematical theories. For example, LU decomposition separates the matrix $\mathbf{A}$ into lower triangular matrix $\mathbf{L}$ and upper triangular matrix $\mathbf{U}$ at precomputation phase. This decomposition allows two step solves the original problem by $\mathbf{L}y = b$ for $y = \mathbf{U}x$ and $\mathbf{U}x = y$ for x. These two linear systems can be solved by direct solver such as forward and backward substitution at runtime. This allows much easier solves than the original problem. This is non-approximate solution, thus we can obtain exactly equivalent x to that of the original problem. An another non-approximate approach is matrix preconditioning. Preconditioning finds a preconditioner $\mathbf{P}$ of a matrix $\mathbf{A}$ so that $\mathbf{P}^{-1}\mathbf{A}$ has a smaller condition number than $\mathbf{A}$ at precomputation phase. At runtime, we can solve a linear system $\mathbf{A}\mathbf{P}^{-1}y = b$ for y, where $y = \mathbf{P}x$, instead of solving $\mathbf{A}x = b$ for x using iterative methods. Finally, we can obtain the solution by solving $x = \mathbf{P}^{-1}b$ for x. This trick accelerates the iterative methods because the convergence rate of interactive methods (e.g., conjugate gradient method, GMRES method) usually decreases as the condition number of a matrix. This matrix preconditioning is not an approximation, and it provides the exactly same solution as the original system. There are many methods for computing preconditioner. A popular method to find preconditioner is incomplete Cholesky decomposition for sparse matrix that uses only non-zero entries unlike exact Cholesky decomposition.

On the other hand, many methods use approximation for accelerating the linear system solves. For example, there are many methods called Low-rank approximation. These methods minimizes the cost function measures the fit between a matrix $\mathbf{A}$ and an approximating matrix $\mathbf{R}$ ($||\mathbf{A} - \mathbf{R}||$), subject to a constraint that the approximating matrix has reduced rank rank(**R**) « rank(**A**), so that the dimensionality of $\mathbf{R}$ is reduced. Singular Value Decomposition (SVD) is a major approach used for such reduction. SVD first decomposes the matrix $\mathbf{A}$ into $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$, where the columns of $\mathbf{U}$ and $\mathbf{V}$ consist of the left and right singular vectors, respectively, and $\mathbf{D}$ is a diagonal matrix whose diagonal entries are the singular values of $\mathbf{A}$. Using these decomposed matrices, we can reduce $\mathbf{A}$ by $\mathbf{A}^* = \mathbf{U}^*\mathbf{D}^*\mathbf{V}^{*T}$, where $\mathbf{U} = [\mathbf{U}^*\mathbf{U}^{**}]$, $\mathbf{D} = diag(\mathbf{D}^*, \mathbf{D}^{**})$, and $\mathbf{V} = [\mathbf{V}^*, \mathbf{V}^{**}]$. At runtime, we can decompose the original problem into two easer problems solving $\mathbf{D}^*y = \mathbf{U}^{*T}b$ for y and $y = \mathbf{V}^{*T}x$ for x. Eigen decomposition can be used for the reduction of diagonalizable matrix, on which we focus in the first method in this thesis. It solves the problem $\mathbf{A}\mathbf{U} = \Omega\mathbf{U}$, where Ω is a vector in which each element represents an eigenvalue, and $\mathbf{U}$ is a matrix in which each column vector is an eigenvector. The reduced eigenvector matrix $\mathbf{U}^*$ can be obtained by remaining only the eigenvectors corresponding to larger eigenvalues. At runtime, the original problem can be transformed into $\mathbf{U}^{*T}\mathbf{A}y = \mathbf{U}^{*T}b$, where $y = \mathbf{U}^*x$, and we first solve this equation for y. Using y, we can obtain the approximated solution by $x^* = \mathbf{U}^{*T}y$. This transformed equation can be solved fewer computational cost than the original problem.

Precomputation for Interactive Visual Rendering

Model-based precomputation methods can be used for accelerating visual rendering applications at runtime. For realistic visual rendering, simulating accurate global illumination (GI) effects such as self-shadowing, ambient occlusion, caustics, and inter-reflections are required. However, computing these effects is very expensive. To address this problem, Precomputed Radiance Transfer (PRT) [120] approximates the light distribution around an object as a linear combination of spherical harmonics basis. To approximate global illumination effects and use it for interactive rendering, PRT uses precomputation of the linear response of a single 3D object exposed to (low-frequency) environmental lighting basis vectors. At runtime, using both the precomputed data and lighting informations, PRT approximates global illumination effects at interactive rate.

A critical limitation of PRT is that it can be applied to only static objects. To address this, Zhong et al. [109] introduce spherical harmonic exponentiation that approximates an object as a collection of various radius particles at precomputation phase. James and Fatahalian [58] integrate physically based deformable body simulation and global illumination by precomputation. They prepare deformation palettes that describe the basis of possible deformations of the object, and precompute the global illumination for each basis by similar approach to PRT. Then, the light transfer informations are associated to finite deformation states. At runtime, they simulate the dynamic scene of deformable body and the global illumination simultaneously by a linear combination of the precomputed palettes at an interactive rate.

Precomputation for Interactive Animation

For accelerating 3D physically based animation at runtime, there are many reduction methods. The basic idea of reduction is to project the high-dimensional equation of motion to a carefully chosen low-dimensional subspace to construct a reduced model. It usually precomputes the basis of the deformation of elastic object, and drives the basis at runtime for fast deformation simulation. Elastic deformation of an object can be represented by the ordinary differential equations of motion $\mathbf{M}\ddot{u} + \mathbf{D}\dot{u} + f_{int}(u) = f_{ext}$, where u are mesh vertex displacements, $\dot{u}$ are velocities, $\ddot{u}$ are accelerations, $f_{int}(u)$ and $f_{ext}(u)$ describe internal and external forces, and $\mathbf{M}$ and $\mathbf{D}$ are the mass and damping matrices respectively.

A major approach for reduce this system is modal analysis [41]. Modal analysis solves generalized eigenproblem $\mathbf{KU} = \mathbf{\Omega MU}$ of finite element stiffness $\mathbf{K}$ and mass matrices where $\mathbf{\Omega}$ and $\mathbf{U}$ denote eigenvalues and eigenvectors at precomputation phase. These remained eigenvectors are called linear modes. At runtime, we can simulate the deformation of an object by approximating it as a linear combination of harmonically vibrated linear modes instead of solving the differential equations. This allows much cheaper deformation simulation than the original problem because it requires only summing up the precomputed mode vectors.

There are two critical limitations of linear mode approximation. First, it can not handle the object's local deformation behaviors well unless a large number of basis vectors are used, which in turn would cancel out the benefit of acceleration. Multi-domain subspace techniques [148] provide a good solution to this problem by partitioning the deformable object into multiple domains and constructing reduced models for each domain independently. Second, nonlinearity features of the deformation are lost. This causes significant artifacts when the object deforms largely. Modal warping [18] partially addresses this problem by introucing geometrically nonlinearity (specifically, rotation). Rotation-strain (RS) coordinates [80] also addresses this problem that is more robust. In addition, to adding nonlinearity to subspace reduction, Barbic and James [6] introduced a method of computing modal derivatives to expand the subspace. While their method is based on linear modal analysis, Yang et al. [147] use Krylov-type modes that can also be extended to capture non-linear deformations.

Precomputation for Interactive Sound Applications

Modal analysis described in the previous subsection also can be used for vibrational analysis of a structure because high frequency deformations is equivalent to the vibration. This vibration is radiated into air as sound. Thus, many physically based sound simulation uses modal analysis for precomputation, which is one of the targets in this thesis. As a direct approach of physical sound simulation, [104] attempted to simulate a vibrating object at audio rate (44.1kHz). However, it was impractical for interactive applications because of its expensive computation. To address this, modal sound synthesis has been widely adopted to simulate quasi-rigid body sounds [1, 110, 132, 155]. This uses linearized vibrational properties (modes) obtained by modal analysis at precomputation phase, and at runtime, generates the sounds by a linear combination of them. This approach releases the physically-based sound simulation from the audio rate simulation, and allows to use it for interactive applications.

For an another example of sound application, a computation of a sound radiation from a vibrating object is also an expensive problem. To simulate sound radiation problem, Boundary Element Method (BEM) is widely used. However, it is difficult to execute the overall computation of BEM at interactive rate. To address this, Doug et al. [57, 79] approximate the sound radiation property around an object by a collection of point sources (monopole and dipole sound sources) by the precomputation. At runtime, they simulate the radiation as a simply summing up the point sources at a receiver (microphone) position. This allows interactive rate simulation of sound radiation at runtime.

The motivation of these precomputation methods for sound applications are accelerate the runtime computations based on physics theoretically motivated approximations.

The precomputation method used for our first method described in this thesis is also accelerating physically based sound simulation for an interactive sound design tool. A significant problem of using modal analysis is their expensive computational cost. Our methods allows to analyze the vibrational property (modal analysis) of an object extremely faster. Note that a major difference of our method from previous approaches is that we accelerate precomputation by precomputation. As described above, modal analysis is based on physically theoretical approximation, and used for precomputation because it is expensive. We accelerate this precomputation method using a combination of different types of precomputation approaches including machine learning technique, and make such expensive precomputation method to be executable at runtime.

2.1.2 Data Driven Approach

Many machine learning algorithms are also adopted as precomputation methods for accelerating computationally expensive applications. An advantage to use machine learning for precomputation is to allows to replace each original expensive problem with computationally cheaper regression function. For example, recent deep neural network studies [46, 101] replace expensive computation of global illumination with computationally cheaper functions, and allows realistic visual rendering at interactive rate. In addition, many studies uses machine learning for interactive physical simulation. NeuroAnimator [38] also uses neural network to approximate expensive physical simulation for interactive computer animation. Kim et al. [68] simulate expensive cloth simulation using motion graph that is constructed by real measurement dataset. Data driven fluid [75] accelerates expensive large fluid simulations by replacing expensive physical simulation with regression forests trained by synthetic dataset gathered at precomputation phase. Um et al. [129] also accelerate large expensive smoothed hydrodynamics particles liquid simulation using deep neural network.

An interesting application of accelerating physical simulation by machine learning technique is an interactive design tool. Umetani et al. [130] present an interactive tool for designing freeform and actually flying paper glider. Although aero dynamic simulation is expensive, they achieve it by using machine learning regression method based on actual measured dataset instead of aero dynamics simulation. Nakamura et al. [99] is a similar approach for designing freeform and actually flying bamboo-copter. They use radial basis functions interpolation trained by actually captured flight dataset of bamboo-copters instead of expensive aero dynamic simulation.

2.2 Making Inferences at Runtime

Precomputation methods are deeply linked to recent exponential development of machine learning techniques. A powerful ability of machine learning is inference. Machine learning algorithm extracts feature parameters of dataset at precomputation phase, and at runtime, it uses the extracted features for classification or regression for given new input data. The motivations of the precomputations used for our second and third methods are also to use such high inference abilities for interactive applications. In the second method, we predict the user's next intent by the action sequences for realtime control user interface. Note that this target problem requires seriously faster interaction response than those of existing studies. In this thesis, we describe how our machine learning model is designed for achieving such a quick response. The third method described in this thesis employs two steps optimization for calibrating 3D audio spatialization for a user; feature parameter extraction at precomputation phase and interactive optimization of the extracted features with the user for generating new data at runtime. This two steps machine learning approach can extract the essential factors of a dataset and allows optimization for the user efficiently in the reduced design space. This allows to represent the feature of black box system such as human perception and taste by reduced parameters.

In the following subsections, we describe the precomputation methods for motivating inferences at runtime.

2.2.1 Inferences for Interactive Visual Application

There are many machine learning algorithms used for interactive visual application. For example, support vector machine (SVM) [43] is used for image recognition. SVM belong to the class of maximum margin classifiers. It performs pattern recognition between two classes by finding a decision surface that has maximum distance to the closest points in the training set which are termed support vectors.

Many recent deep neural network (DNN) studies successfully identify visual objects [74, 61] in an image or a video [103]. Various DNN studies can also generate and translate high quality images [3, 56, 81, 82]. Neural network consists of multiple layers in which each layer can be represented as a combination of linear and non-linear functions. We can use arbitrary differentiable functions for these functions. DNN can be stochastically trained by back propagation at precomputation phase. At runtime, DNN can be used for new input data to make inferences with high generalization capability.

For designing 3D character motion, Holden et al. [44, 45] uses DNN to generate a new motion. In their system, the user can edit the character's motion by simply interpolating latent variables which represents the features of trained motion dataset. This editing procedure is easier than conventional motion manipulation (e.g., editing key frames). As a similar approach, Yumer et al.

[153] presented 3D modeling by an AutoEncoder trained at precomputation phase. In this system, the user can explore the intermediate shapes of trained models by interpolating the latent variables. In addition, as a cross domain translation, Karras et al. [65] control the facial expressions of a 3D model from audio speech signals that allows goal oriented design of facial motion.

Such inference ability of machine learning is also useful for user interfaces for an interactive design tool. Ribeiro and Igarashi [113] develop a sketch user interface that progressively learns visual models of objects from user sketches, and uses the models at the next interactive with the user. In this system, the user can create sketch with machine by mutual cooperation. In addition, there are various useful design tools for sketching and image retouching using DNN; filling the holes in an image [54], sketch beautification from rough sketch [118], colorize monochrome photo [53], and painting colors naturally on monochrome sketch [107]. Although these algorithms largely depend on the specifications of trained dataset, such design tools have a possibility to augment the user's creativity.

2.2.2 Inferences for Interactive Sound Application

Handling time series data is an important problem for sound application. For example, an audio signal sample at a time is usually correlated with the previous time samples. Hidden Markov model (HMM) [87] is widely used for such time series data. HMM models a system by a Markov process with unobserved (hidden) states that assumes the current state is determined only by the previous state, and the output observation at time t is dependent only on the current state. Given a set of training dataset, we can estimate the model parameters (initial, transition, and observation probabilities) in HMM by two standard approaches. If the training dataset contain both the inputs and outputs of a process, we can perform supervised training by equating inputs to observations, and outputs to states. If only the inputs are provided in the dataset, we use unsupervised training to guess a model that may have produced those observations. For supervised training of HMM, maximum likelihood estimation is widely used. On the other hand, Baum-Welch algorithm [108] is used for unsupervised training.

Many speech recognition and synthesis algorithms use HMM for modeling human speech [126, 150]. It also can be used for beat tracking [26] and chord detection [128] of music. HMM has an advantage that it can be trained by fewer data than DNN. Thus, it can be used for applications in which it is difficult to gather large training dataset. Specifically, modeling musical sessions between human musicians and computer [86] is an example that is suited for HMM because we can not take rehearsals so many times for making training dataset.

Neural network also can be used for time series data. Such type of neural network is called recurrent neural network (RNN) [114]. RNN takes as their input not just the current input example they see, but also what they perceived one step back in time. It is often said that RNN has memory. A layer in RNN is represented as

$$h_t = \psi(\mathbf{W}x_t + \mathbf{U}h_t + b), \tag{2.1}$$

where x_t and h_t denote the input and hidden vectors at time t respectively, $\mathbf{W}$ and $\mathbf{H}$ are weight matrices, b is the bias vector, and $\psi()$ is an arbitrary non-linear function. RNN can be trained back propagation through time (BPTT) [98]. BPTT unfolds the RNN through time to be able to back propagation. Long short term memory (LSTM) [35], which is a variant of RNN, successfully classify, process and predict long time series that have time lags of unknown size and bound between important events. There are many recent studies of speech synthesis [30], singing synthesis [11], and sound event detection [123, 124] use LSTM instead of HMM.

Dilated convolution [133] is an another approach to treat audio signal by neural network instead of RNN. Standard convolutional layers need either large filters to capture a sufficient range of input sequence. However, the computational cost increases extremely to reach a certain size of the receptive field for the actual output. On the other hand, dilated convolution just refers to the fact that a certain number input values is skipped when applying the filter of a convolutional layer. This allows to capture the correlations between long sequence data with a reasonable computational cost. This type of neural network can be used for many sound applications including text to speech synthesis [135] and music generation [29].

Chapter 3

Interactive Physically-Based Sound Design of 3D Model using Material Optimization

3.1 Introduction

Realistic sound effects that respond to visual events in a scene significantly enhance the user experience in virtual environments. Traditionally, sound effect designers prepared pre-recorded and pre-edited audio samples, and these samples were synchronized to visual events by manual tweaking (e.g., for feature films) or using scripts (e.g., for VR and games). However, it is laborious to prepare appropriate audio samples for a large variety of visual events, limiting expressiveness and variation of sound effects.

As a solution for this problem, physically-based sound synthesis techniques known as sound rendering [104] have been proposed by the graphics community in the last decade. Modal sound synthesis [1, 155] is widely used for sound simulation of quasi-rigid bodies; it can efficiently produce physically-plausible sounds responding to a large variation of visual events (e.g., collision, bounce, and scratch). This method reduces the effort of manipulating a large number of audio samples for sound designers because it does not use any pre-recorded audio sample. All sounds are automatically triggered and rendered by physical simulation responding to visual events. However, although input parameter for these techniques is the material distribution inside the model, designing the internal material distribution properly so that the system produces a specific sound as a result of physical simulation is difficult even for professional designers.

To address this problem, we propose an example-based interactive design framework for rendering the physically-based sound of a 3D model using material optimization (Figure 3.1). Our approach enables a user to control the timbre of modal sound synthesis easily without directly specifying the internal material distributions. The user first provides a 3D surface model to

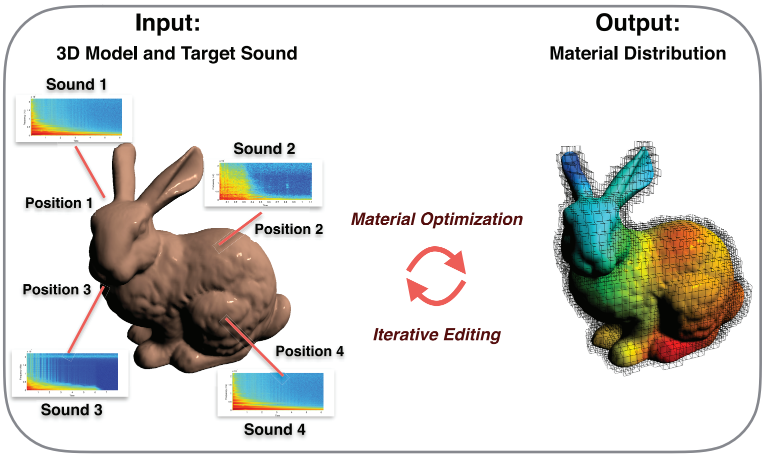


Figure 3.1: The user assigns several target sounds to sample points on given 3D model as examples. The system then optimizes the material distribution inside the model so that physically-based sound simulation produces the expected sounds. The user can check the simulated sound during the optimization interactively and re-assign additional target sounds to design the desired sounding object. Finally, the system outputs embedded FEM mesh with eigenpairs which can be used for standard physically-based sound rendering pipeline.

the system as input. Next, the user selects a few sample positions on the model surface, and assigns corresponding sound clips that define the target sounds to be rendered when the positions are struck. The system then optimizes the material distribution inside the model so that physically-based sound simulations yield the expected sounds.

However, modal analysis that is required for obtaining the vibrational property of the object at an iteration in our optimization is a prohibitively expensive. To execute the optimization at an interactive rate, we present a novel fast approximate modal analysis method that achieves three orders of magnitude acceleration compared to the standard modal analysis (Figure 3.2). Our technique consists of data-driven finite element coarsening of the mesh and hierarchical component mode synthesis with efficient error correction. Our data-driven online coarsening extends Chen et al.'s method [15] to handle a large range of continuous material settings by reducing the material parameter space, and can be evaluated with a constant cost for a large amount of datasets using regression forests. Additionally, our highly parallelized hierarchical component mode synthesis extends conventional methods [5] to efficiently compute approximate solutions of modal analysis, and our error correction algorithm efficiently improves its accuracy.

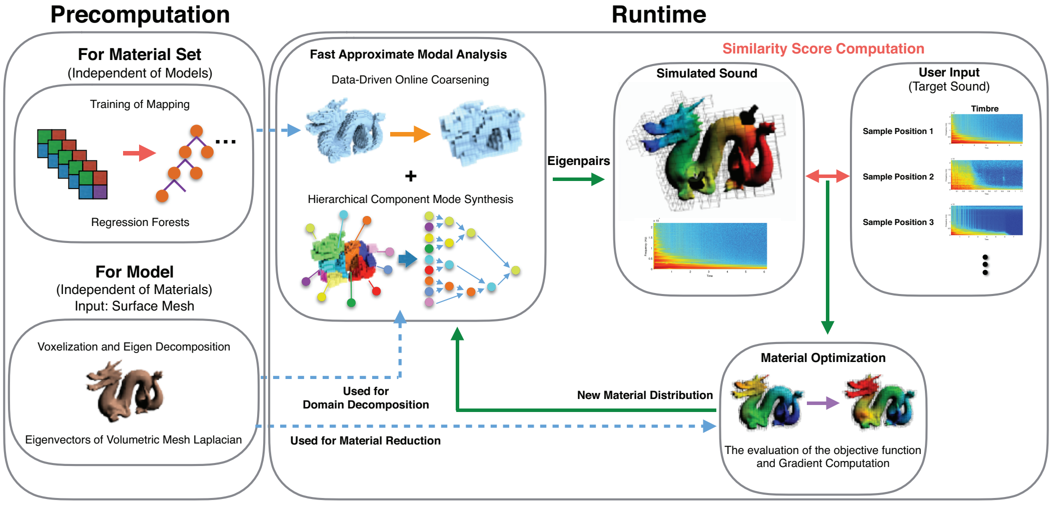

Precomputation
Runtime
For Material Set
(Independent of Models)
Training of Mapping
Regression Forests
For Model
(Independent of Materials)
Input: Surface Mesh
Voxelization and Eigen Decomposition
Eigenvectors of Volumetric Mesh Laplacian
Fast Approximate Modal Analysis
Data-Driven Online Coarsening
+
Hierarchical Component Mode Synthesis
Eigenpairs
Similarity Score Computation
Simulated Sound
User Input
(Target Sound)
Timbre
Sample Position 1
Sample Position 2
Sample Position 3
Used for
Domain Decomposition
New Material Distribution
Used for Material Reduction
Material Optimization
The evaluation of the objective function
and Gradient Computation

Figure 3.2: Algorithm Overview. Our optimization algorithm consists of the precomputation and runtime. An iteration of our optimization procedure at runtime consists of 3 steps. First, the system computes modal analysis to obtain the vibrational property of the object. Second, it computes the similarity score between the simulated sounds of the object and user specified target sounds. According to this similarity score, the system updates the material distribution inside the object to minimize the cost.

3.2 Related Work

3.2.1 Parameter Acquisition for Modal Sound Synthesis

To determine the material parameters used in modal sound synthesis, Pai et al. [106] and Corbett et al. [23] acquired the parameters from actually measured impact sound data, and interpolated them in auditory space. A robotic actuated device is used to apply impulses on a real object at a large number of sample points, and map the recorded impact sounds to virtual objects. However, the measurement procedure of such a huge number of samples for an object and manipulating them are prohibitively expensive. FoleyAutomatic [132] also employed similar approach, but interpolated them in modal space for achieving rich sound interactions. However, they also require sufficient amount of samples to estimate the modal function on the surface. Same example sound can be reused at different locations, but it causes a lack of the sound variations when the object interacts with other objects at various locations. This problem becomes profound when the model to be designed has a larger scale.

To avoid measuring such a huge number of parameters for one object from many audio clips, Lloyd et al. [83] proposed a data-driven approach to assign the sound of an object from only one audio clip. They estimated the modal parameters from the audio clip, and at runtime, they randomized the mixture gains of all the tracked modes to generate imaginary varied sounds when hitting different locations on the object. However, this method produces

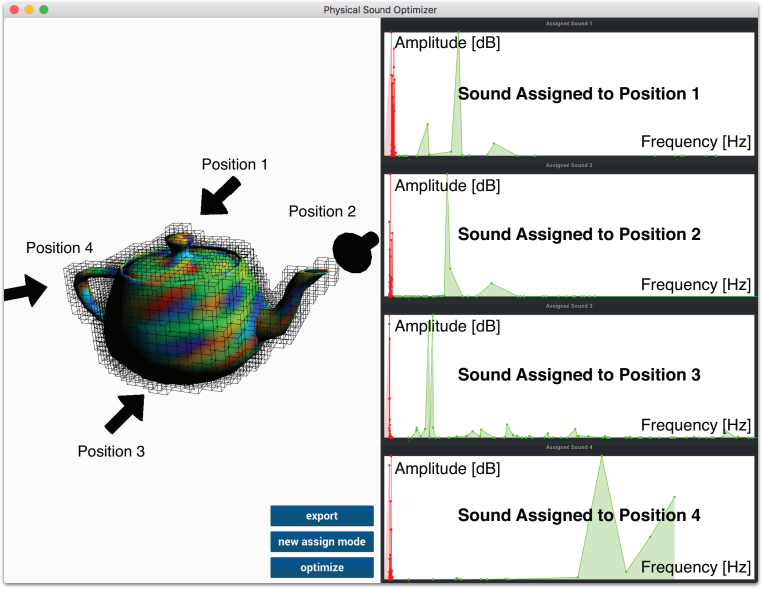


Figure 3.3: The user interface view. The left pane allows the user to assign the target sounds for the model and preview the contact sound while the right views represent the power spectrums of assigned sounds (green) and the sounds when the positions the user selected are struck (red). The black arrows on the left pane represent the positions the user assigned targe sounds.

unnatural artifacts because the sounds are not consistent with hit points.

As another approach, Ren et al. [111] proposed a method to estimate the material specific parameter (Rayleigh damping parameters) directly instead of modal parameters from a audio clip under the assumption of uniform material distribution inside the object. The advantage of their approach is that it enables the estimated material parameters to be transferred to different shapes. However, their approach requires that the real object have exactly the same shape as the virtual model to be estimated and should be easy to prepare. These requirements are impractical to implement in actual scenes, which is considered in this thesis.

On the other hand, the vibration of an object is radiated into air, and it propagates as a sound in the space. This propagate specification which is affected by the room shape and materials is also an important factor to design for perceptual sound, specifically, determining material parameters of the objects and walls in the room is the most important design factor that is similar to vibrating objects. There are several kinds of material parameters of an object used for sound propagation problems. A major parameter is bidirectional reflection distribution functions that describe the reflectance of sound with respect to the incoming and outgoing directions [36]. However, these functions are difficult to measure in real scene. Alternatively, several algorithms have been proposed to inversely optimize the acoustic parameters from sounds [97] for a room that is built by primitive shapes or estimate the acoustic properties of real world scenes for inverse sound rendering [102, 115]. Christensen et al. [19] uses genetic algorithm to estimate the material properties of a room so that it matches measurements. Their motivations are similar to our approaches. However, these methods are computationally expensive and can not be used for interactive applications.

3.2.2 Vibrational Property Optimization

To obtain the desired vibrational property of an object, Yamasaki et al. [146] optimized the shape and topology of an industrial structure using levelset optimization, and controlled the several lowest eigenfrequencies. Yua et al. [151, 152] optimized the topology of a violin's body as specific thin shell structure to control the mode frequencies and amplitudes (mode vectors) that are expected to be largely contributed to the timbre. Bharaj et al. [9] optimized the shape of a common elastic structure to control both a few mode frequencies as well as their amplitudes for fabricating metal percussion instruments. Our formulation is similar to theirs, but there are four differences. 1: We control a much larger number of modes for dramatically changing the sound's timbre and sacrificing the fabrication possibility. 2: We optimize the material distribution while maintaining the shape whereas they optimize the shape. 3: Our optimization runs at an interactive rate that is enabled by an expansion of data-driven finite elements method (FEM) [15] and highly parallelized hierarchical component mode synthesis. 4: Our objective function considers the perceptual differences of two sounds whereas they use square distances of frequencies and amplitudes.

3.2.3 Modal Analysis

Modal analysis is a well-studied technique in both computer graphics and engineering. It solves the generalized eigenproblem of the finite element stiffness and mass matrices to obtain the vibrational frequencies and the corresponding deformations [41]. Because modal analysis is a time-consuming operation, it is usually used for only the precomputation phase. As some exceptions, Umetani et al. [131] introduced 2D modal analysis into an interactive design tool for percussion instrument by limiting the fundamental mode computation. Maxwell and Bindel [90] computed quasi-3D modal analysis of thin shell structure percussion instruments including the several overtones at a quasi-interactive rate. We introduced 3D modal analysis of a more complex structure into an interactive application.

Many studies focused on the improvement of the computational efficiency of modal analysis. A powerful solution is the domain decomposition approach called the component mode synthesis method (CMS) [51]. CMS decomposes a large problem into many small problems of subdomains and merges them. There are several variations of CMS according to how the boundaries between subdomains are treated [16, 149]. The major approach is the Craig-Bamptom method [5] that treats the interfaces of subdomains as fixed. However, finding an optimal division of a mesh in subdomains is non-trivial, and it should be often undertaken manually for improving the accuracy. It requires additional expertise and manual efforts by the user. Our approach does not distinguish between subdomains and boundaries, and automatically decomposes it as a hierarchical structure and merges them in parallel. In addition, we improve the accuracy using the fast error correction algorithm, which consists of a combination of the subspace iteration method [7] and sparse mass-Gram-Schmidt process [147].

3.3 User Workflow

This section describes the user workflow of our interactive physically-based sound design tool. Please see the supplemental video for an interactive demonstration. As seen in the screen capture shown in Figure 3.3, the user first provides a 3D surface model as an input. The system automatically voxelizes it and converts it into a uniform hexahedral finite element mesh, and executes precomputations as described in the next section. Next, the user selects a vertex position on the surface of the mesh using the mouse, and assigns a sound clip to the position by a drag-and-drop operation. The sound clip defines the sound to be rendered when the model is stuck at the position. The assigned sound clip can be either a pre-recorded real sound (exists in the real world) or an artificial sound (e.g., sound generated by sound synthesizer), but it needs to be an attenuated contact-like sound (free vibrational sound caused by single impulse. An impulse response is ideal). The system allows the user to select multiple positions for each corresponding sound clip. After assigning sounds, the user presses the “optimize" button, and the system optimizes the material distribution inside the model to obtain the desired sound properties. Finally, the system exports the optimized embedded finite element mesh for the surface model with the eigenpairs, and the user can use it for modal sound synthesis.

The optimization gradually progresses at an interactive rate. The system visualizes the current material distribution inside the model by colors and the resulting sounds when the sample positions are struck by power spectrums. The user also can check the sound by clicking the mouse on the mesh surface during optimization at any time. The user can stop the optimization procedure at an arbitrary timing, reassign another sound to a new sample point, and restart the optimization iteratively. In this way, the user can interactively design the physically-based sound for a 3D object as if it were a sound synthesizer.

3.4 Algorithm Overview

Figure 3.2 shows an overview of our optimization algorithm. Our algorithm consists of two stages: the precomputation stage and the runtime. The precomputation stage consists of two parts. One is precomputation for each material set (independent of models), and it constructs regression forests for data-driven FEM. The regression forests are used for online mesh coarsening using data-driven FEM (§3.6.1). The other is precomputation for each input model (independent of materials), and it involves voxelizing the model into a hexahedral FEM mesh and computation of the eigenvectors of the volumetric Laplacian of the mesh following [142]. The eigenvectors of the volumetric Laplacian are used for material reduction (§3.5), and mesh segmentation (§3.6.2).

At runtime, the system minimizes the perceptual difference between the user-specified input sound and simulated sound by iterative optimization of material distribution (§3.5). We consider vibrational property (mode frequencies and amplitudes) to measure perceptual difference (§3.4). We optimized Young's modulus at each element of FEM, and we kept the densities and Poisson's ratios constant for simplicity. At each iteration, it is necessary to execute a modal analysis of the model to compute the resulting sound. Conventional modal analysis solves the generalized eigenproblem of large stiffness and mass matrices, but it is prohibitively expensive and impractical to use during iterative optimization. To address this, we propose a fast approximate modal analysis based on a combination of data-driven FEM using regression forests (§3.6.1) and hierarchical component mode synthesis method including error correction (§3.6.2).

3.5 Problem Formulation

When the user assigns a sound clip onto a sample position, the system extracts the parameters of the sound's timbre from it. An attenuated contact sound can be parameterized by modal parameters (frequencies, amplitudes, and dampings). For the details of the modal parameters, please see Appendix A. We employ Ren et al.'s technique [111] to extract these parameters from a sound clip. We also extract the residual parameters following them. After T assignments, the system has N sorted mode frequencies of assigned sounds $(F_1, ..., F_N)$, corresponding dampings $(D_1, ..., D_N)$, corresponding residuals $(R_1, ..., R_N)$, and corresponding amplitudes at T sample positions $(A_1^1, ..., A_N^1), ..., (A_1^T, ..., A_N^T)$. We call these extracted parameters as target parameters.

For a given finite element mesh, we compute the first N mode frequencies $(f_1, ..., f_N)$ and corresponding amplitudes at T sample positions $(a_1^1, ..., a_N^1), ...,$ $(a_1^T, ..., a_N^T)$ using modal analysis. The modal analysis computes a generalized eigenproblem: $\boldsymbol{KU} = \boldsymbol{\Lambda MU}$, where $\boldsymbol{K}$ and $\boldsymbol{M}$ denote the stiffness and mass matrix respectively and $\boldsymbol{\Lambda}$ and $\boldsymbol{U}$ denote the eigenvalues and the corresponding eigenvectors. To compute the mode amplitudes, we assume each sample position p_i $(i = 1, ..., N_p)$ is struck by a unit force impulse $\boldsymbol{f}_n^{p_i}$ which has the inverse direction of the surface normal n at the position. Then, the k-th mode amplitude at the position p_i is represented as $a_k^{p_i} = u_k^T f_n^{p_i}$, where $\boldsymbol{u}_k$ is the k-th eigenvector.

Using the target frequencies F, amplitudes A and simulated parameters, our objective function for minimizing the perceptual difference of the mode frequencies is represented as

$$\boldsymbol{E}_f = \frac{1}{2}\sum_{i=2}^{N}(Bark(s_f f_i) - Bark(F_i))^2, \tag{3.1}$$

where $Bark(f)$ is a function to transform the frequency to critical band rate $[bark]$ [158], and $s_f = F_1'/f_1$ is the scaling factor. The objective function for amplitudes is also obtained using the balances with other mode amplitudes at the position:

$$\boldsymbol{E}_a = \frac{1}{2}\sum_{j=1}^{T}\sum_{i=2}^{N}\left(\frac{a_i^j}{a_{max}^j} - \frac{A_i^j}{A_{max}^j}\right)^2, \tag{3.2}$$

where a_{max}^j and A_{max}^j denote the largest amplitude at the position j of the simulated and target's modes respectively. These formulations are similar to [9]; however, we use the perceptual metrics whereas they use square distances of frequencies and amplitudes. We minimize these functions by optimizing the Young's modulus $Y_e \in \mathbb{R}^M$ at each finite element e, where M denotes the number of the elements. Finally, our design problem is formulated as

$$\arg\min_{Y_e} : \ w_f \boldsymbol{E}_f + w_a \boldsymbol{E}_a, \ \ subject\ to : Y_e > 0, \tag{3.3}$$

where w_f and w_a denote the positive weights.

Note that we do not optimize damping parameters. We instead reuse the estimated damping from the assigned sound clips as mode-dependent damping. This means that our damping is not spatially constant. This setting is physically incorrect, but it makes the problem simpler.

3.6 Material Optimization

The optimization of element-wise material parameters is impractical. To reduce the design space of material parameters, we introduce the reduction technique of [142]. The technique expresses the Young's modulus as $Y = \boldsymbol{\Phi} z$ using the eigenvectors of the volumetric mesh Laplacian $\boldsymbol{\Phi} \in \mathbb{R}^{M \times m}$, and uses the generalized material parameters $z \in \mathbb{R}^m$, $(m << M)$ for the optimization.

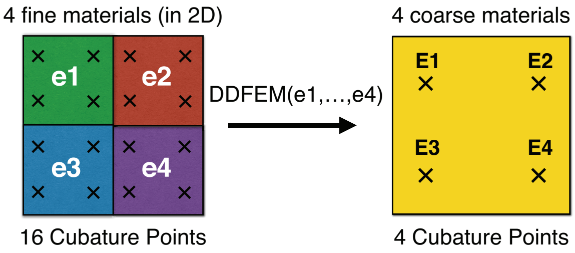


Figure 3.4: Data-Driven Coarsening [Chen et al. 2015] (in 2D illustration). The function $DDFEM()$ takes four material parameters $(e1, e2, e3, e4)$ of fine four elements (left) and returns corresponding four coarse material parameters $(E1, E2, E3, E4)$ at the quadrature points (right) to minimize the error.

Then, our design problem can be rewritten in the reduced space as

$$\arg\min_{z} : \; w_f \boldsymbol{E}_f + w_a \boldsymbol{E}_a + w_r \boldsymbol{R}, \;\; \boldsymbol{R} = \frac{1}{2} z^T \boldsymbol{Q} z, \tag{3.4}$$

where w_r is a weight, $\boldsymbol{R}$ is the regularization term, and $\boldsymbol{Q}$ is the reduced Laplacian matrix which is diagonal and its entries consist of the eigenvalues of the volumetric mesh Laplacian (please see [142] for the details). This material reduction also has a merit to reduce the over-fitting problem.

We solve our design problem Eq. (3.4) by decomposing it into two problems $min : \boldsymbol{E}_f$ and $min : \boldsymbol{E}_a$, and minimizing them alternately. We employ a hybrid optimization scheme [17] of evolutional strategies (we used CMA-ES [40]) and gradient descent approach (we employed the Quasi Newton method). For the details of the gradient computation and this hybrid scheme, please see Appendix B and C.

3.7 Fast Approximate Modal Analysis

At each iteration during our optimization, a modal analysis is required for the evaluation of the objective function and its gradient. However, standard modal analysis (solving a generalized eigenproblem of large stiffness and mass matrices) is prohibitively expensive and impossible to execute at an interactive rate. To address this, we present a method that combines extended data-driven online coarsening of finite elements (§3.6.1) and highly parallelized hierarchical component mode synthesis (§3.6.2).

3.7.1 Data-Driven FEM using Regression Forests

In this section, we explain the data-driven online coarsening of the FEM mesh. It takes the detailed voxel mesh ($2 \times 2 \times 2 \times$ cube elements) as input and generates a coarse approximated mesh (a cube element) as output using the the material parameter mapping learned from training data in the precomputation

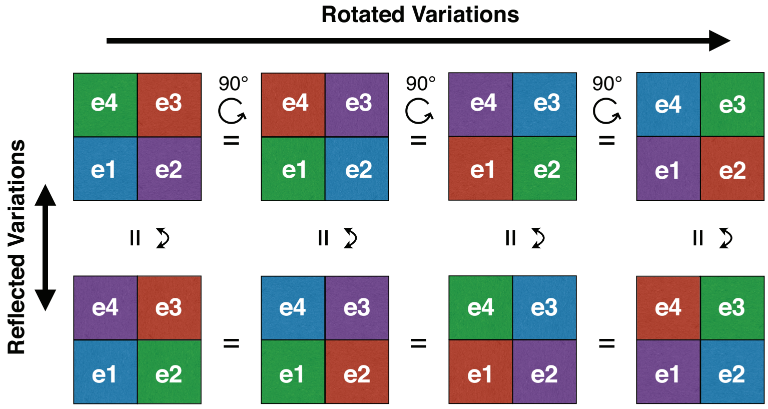


Figure 3.5: Eight equivalent cell variations (in 2D illustration). The top row represents four rotated variations and the bottom row represents four reflected variations.

step (Figure 3.4). The concept of our data driven FEM coarsening is based on [15]. The goal of their data-driven FEM is obtaining

$$(E_1, ..., E_8) = DDFEM(e_1, .., e_8), \tag{3.5}$$

where $DDFEM()$ is a function that takes eight material parameters $(e_1, ..., e_8)$ of a detailed mesh and returns the corresponding eight coarse material parameters $(E_1, ..., E_8)$ at the cubature points to minimize the error. Their system computes this function for all possible input values in precomputation and stores the result in the main memory. The system then evaluates this function referring the memory at runtime. It aggressively accelerates FEM while maintaining the accuracy by reducing the Dofs $(24/81)$ and the number of the cubature points $(8/64)$ although the total number of the material parameters remains unchanged between the detailed and coarse mesh. However, in their approach, given N discrete materials, the number of material combinations becomes N^8. Although they also proposed a compression algorithm by retaining only the small number of representative material combinations, it still cannot be used for our material optimization that requires a large range of continuous material settings. Additionally, it is non-trivial to obtain an actual value from such representative materials. To address this, we present three techniques: 1: Overlapping Free Cell Ordering, 2: Scaling Factor Separation, 3: Regression Forests. The former two techniques reduce the parameter space of the feature vector e (the detailed eight material parameters) for efficient machine learning, and the last technique enables handling of a large amount of dataset with a constant evaluation cost.

Overlapping Free Cell Ordering

As shown in Figure 3.5, the rotated and reflected variations of a material setting are basically equivalent. To enumerate such patterns increases the

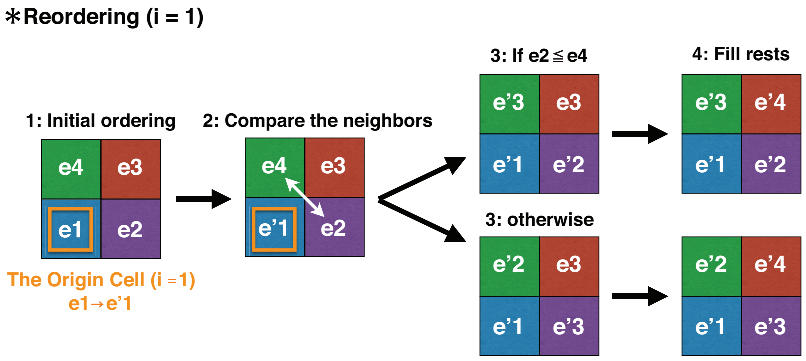


Figure 3.6: Overlapping Free Cell Ordering (in 2D illustration). 1: At the i-th cell evaluation (in this example, we assume $i = 2$), e_i becomes the origin cell e'_1. 2: we compare the material values of the adjacent cells. 3: the smaller cell becomes e'_2 and the other becomes e'_3. 4: The left cell becomes e'_4.

parameter space of the feature vector unnecessarily and it should be reduced for efficiency. To address this, we define Overlapping Free Cell Ordering algorithm which makes explicit consideration of rotated and reflected patterns unnecessary.

First, we redefine the data-driven function $DDFEM()$ Eq.3.5 as

$$E_i = DDFEM_i(e_1, .., e_8), \ \ i = 1, 2, ..., 8. \tag{3.6}$$

Our data-driven FEM function returns a scalar while Eq.3.5 outputs a $\mathbb{R}^8$ vector. It means we repeat this $DDFEM()$ evaluation eight times to convert a detailed $2 \times 2 \times 2$ element into a coarse element. Next, we reorder the numbering of the eight cells by each $DDFEM()$ evaluation. We show this operation as a 2D example in Figure 3.6. The indices of the cells are defined in a local $\mathbb{R}^3$ space coordinate. At the i-th evaluation within the eight evaluations, we define the i-th cell as the origin e'_1. Then, we compare the value of the Young's modulus of the three adjacent cells of the origin cell (in 2D, two cells), and define the index the cell who has the smallest value as e'_2, the cell who has the secondary smallest value as e'_3, and the other cell as e'_4. Finally, we decide the ordering of the rest four cells by the following rule: The cell that is adjacent to e'_2 and e'_3 becomes e'_5. The cell that is adjacent to e'_3 and e'_4 becomes e'_6. The cell that is adjacent to e'_2 and e'_4 becomes e'_7. The last one becomes e'_8.

Then, using these reordered parameters, our $DDFEM()$ function is redefined again as

$$E_i = DDFEM_i(e'_1, e'_2, ..., e'_8), \ \ e'_1 = e_i. \tag{3.7}$$

By using this representation, we can avoid explicit enumeration of the eight rotated and eight reflected patterns of a material pattern, and reduce the input parameter space in 3D at both training and runtime. For dataset generation at the training, we first determine the value at the origin cell, and seed the values at the three cells e'_2, e'_3, e'_4 to be $e'_2 \leq e'_3 \leq e'_4$, and the rest of the values are

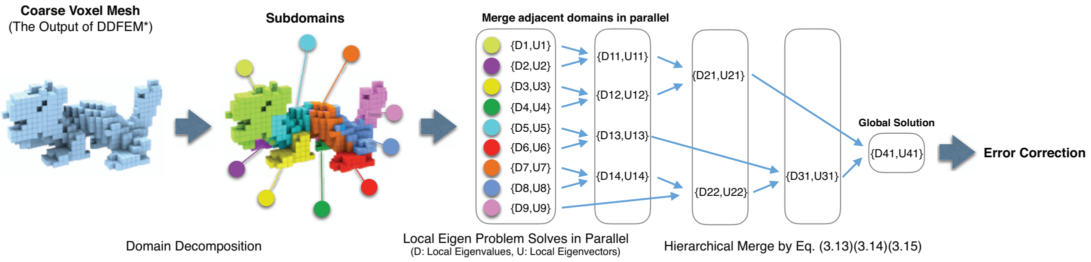


Figure 3.7: Hierarchical Component Mode Synthesis. After coarsening the mesh, we decompose it into many subdomains and hierarchically merges them with reducing their DoFs in parallel. Finally, we improve the accuracy using an error correction algorithm.

randomly seeded.

Scaling Factor Separation

Young's modulus has a large range of the value 10^{-2} (Rubber) $\sim 10^{3}$ (Diamond) GPa while Poisson's ratio has a small range $(-1/2,\ 1/2)$. It is difficult to treat a practical amount of data for such a large range during training. To avoid this, we dramatically reduce the training size by separating the scale factor.

Based on [15], our $DDFEM()$ is constructed to minimize the square difference of the integral of the strain energy density functions between the detailed and coarse meshes

$$\arg\min_{E_i} \sum_{f \in F} \left(\sum_{i=1}^{8} w_i v_i^c(f, E_i) - \sum_{j=1}^{8} \sum_{i=1}^{8} w_i v_{ji}^d(f, e_j') \right)^2, \tag{3.8}$$

where w denotes the cubature weights, F denotes a set of randomly sampled external forces, and v^c and v^d represent the strain energy density function of the coarse and detailed mesh respectively. Here, the strain energy density function in linear elastic is represented as $v(f, e) = \boldsymbol{K}(e)u(e)^2 = \boldsymbol{K}(e)(\boldsymbol{K}^{-1}(e)f)^2$, where $\boldsymbol{K}(e)$ and f are the stiffness matrix and the external forces respectively. In addition, multiplying e by a scalar s, $v(f, s \cdot e) = \boldsymbol{K}(s \cdot e)u(s \cdot e)^2 = s\boldsymbol{K}(e)((s\boldsymbol{K}(e))^{-1}f)^2 = v(f, e)/s$ because $\boldsymbol{K}()$ is the linear function of e. Then, the minimization problem

$$\arg\min_{E_i} \sum_{f \in F} \left(\sum_{i=1}^{8} w_i v_i^c(f, E_i) - \sum_{j=1}^{8} \sum_{i=1}^{8} w_i v_{ji}^d(f, s \cdot e_j') \right)^2, \tag{3.9}$$

is equivalent to

$$\arg\min_{E_i' = E_i/s} \sum_{f \in F} \left(\sum_{i=1}^{8} w_i v_i^c(f, E_i') - \sum_{j=1}^{8} \sum_{i=1}^{8} w_i v_{ji}^d(f, e_j') \right)^2. \tag{3.10}$$

This means that we can separate the input parameter space of our $DDFEM()$ problem by the multiplication of the value of the origin cell as a scale factor and their quotients. Finally, we can obtain our $DDFEM()$ function as

$$E_i = e_i \cdot DDFEM_i\left(\frac{e'_2}{e_i}, ..., \frac{e'_8}{e_i}\right). \tag{3.11}$$

An advantage of this representation is that it reduces not only the range of dataset but also the dimensions of the feature vector from $\mathbb{R}^8$ to $\mathbb{R}^7$. Note that we assume our model as linear elastics although the original $DDFEM()$ treats nonlinearity because the vibrational analysis discussed in this paper is a linear analysis. Introducing the nonlinearity for large deformation is a future work.

Regression Forests

In contract with Chen et al.'s method [15], we do not construct the database of data-driven materials because of two reasons. First, their database approach cannot handle the inputs that are not included in the training dataset because it has no generalization ability. Second, the evaluation cost at runtime is increased at a rate proportional to the amount of the dataset although the amount of the dataset should be increased for handling more material patterns. To address these problems, we train our $DDFEM()$ function using two regression forests. Our regression forests are similar to [75] which construct each tree through two steps training: tree structure construction with a subset of learning data and least-square solve for the regression coefficients at each leaf node with all the dataset. The regression forest has an advantage of constant cost evaluation even if the amount of the dataset is increased. Finally, our $DDFEM()$ becomes

$$\begin{cases} E_i = \bar{e} \ \cdot Reg_1(\frac{e'_2}{\bar{e}}, ..., \frac{e'_8}{\bar{e}}) & (e_i = 0) \\ E_i = e_i \cdot Reg_2(\frac{e'_2}{e_i}, ..., \frac{e'_8}{e_i}) & (e_i > 0), \end{cases} \tag{3.12}$$

where $Reg()$ represents the regression function, and $\bar{e}$ is the average of the Young's modulus in the target eight cells.

3.7.2 Hierarchical Component Mode Synthesis

After coarsening the mesh, we compute modal analysis using a novel hierarchical component mode synthesis method (HCMS) including an efficient error correction algorithm (Figure 3.7). It takes the coarse voxel mesh as input and solves a generalized eigenproblem via hierarchical merging. It first decomposes the mesh into small components and solves a generalized eigenproblem for each component. It then hierarchically merges adjacent components and solves generalized eigenproblems for the merged component. Conventional CMS [5] computes the eigenmodes of a structure by combining several small local subdomains after decomposing it into several small subdomains. Our

HCMS decomposes a structure into finer subdomains compared to conventional CMS to increase the computational efficiency while sacrificing accuracy. To compensate for the loss of accuracy, we apply a subspace iterative error correction using the result of HCMS as an initial solution.

To simplify the explanation for our HCMS, we first begin with assuming that a model can be decomposed into two non-overlapping domains $S1$ and $S2$ as in conventional CMS, and the eigenpairs of each domain are already known. Under this assumption, the entire stiffness matrix $\boldsymbol{K}_{total}$ and the entire mass matrix $\boldsymbol{M}_{total}$ can be represented as

$$\boldsymbol{K}_{total} = \begin{bmatrix} \boldsymbol{K}_{11} & \boldsymbol{K}_{12} \\ \boldsymbol{K}_{12} & \boldsymbol{K}_{22} \end{bmatrix}, \ \boldsymbol{M}_{total} = \begin{bmatrix} \boldsymbol{M}_{1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{M}_{2} \end{bmatrix}, \tag{3.13}$$

where $\boldsymbol{K}_{11}$, $\boldsymbol{K}_{22}$ and $\boldsymbol{M}_1$, $\boldsymbol{M}_2$ denote the local stiffness and mass matrices of each sub-domain respectively. $\boldsymbol{K}_{12}$ and $\boldsymbol{K}_{21}$ are the interface matrices that connect the domains $S1$ and $S2$. If the eigenvectors of each domain $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$ are already known, we can rewrite the Eq. (3.13) using the reduced matrices of each domain with remaining the lower frequency modes as

$$\boldsymbol{K'}_{total} = \begin{bmatrix} \boldsymbol{D}_1 & \boldsymbol{U}1^T\boldsymbol{K}_{12}\boldsymbol{U}2 \\ \boldsymbol{U}2^T\boldsymbol{K}_{12}^T\boldsymbol{U}1 & \boldsymbol{D}_2 \end{bmatrix}, \tag{3.14}$$

where $\boldsymbol{D}_1 = \boldsymbol{U}1^T\boldsymbol{K}_{11}\boldsymbol{U}1$ and $\boldsymbol{D}_2 = \boldsymbol{U}2^T\boldsymbol{K}_{22}\boldsymbol{U}2$ are diagonal matrices in which each diagonal entry is the eigenvalue of the respective subdomain. Note that the entire mass matrix also takes the same form for, $\boldsymbol{U}_1^T\boldsymbol{M}_1\boldsymbol{U}_1 = \boldsymbol{I}$, and $\boldsymbol{U}_2^T\boldsymbol{M}_2\boldsymbol{U}_2 = \boldsymbol{I}$, meaning that the entire mass matrix becomes an identity matrix. Athough conventional CMS distinguishes the interface of adjacent subdomains and subdomains, and assumes the interface as fixed [5] or considers the boundary modes [148], our approach neither distinguish them nor fix the interface, and does not treat the interface explicitly. We can obtain a reduced eigenproblem of the entire structure as $\boldsymbol{K'}_{total}\boldsymbol{U'}_{total} = \boldsymbol{\Lambda}_{total}\boldsymbol{U'}_{total}$, where $\boldsymbol{\Lambda}_{total}$ is a diagonal matrix in which each diagonal entry is the eigenvalues of the entire domain. We solve this reduced eigenproblem, and finally recover the global eigenvectors by

$$\boldsymbol{U}_{total} = \begin{bmatrix} \boldsymbol{U}_1 \\ \boldsymbol{U}_2 \end{bmatrix} \boldsymbol{U'}_{total}. \tag{3.15}$$

We apply the pair wise merger explained above in a hierarchical manner. We divide a large structure into many small subdomains and merge them in a hierarchical manner (Figure 3.7). The system first decomposes the volumetric mesh after coarsening into many small subdomains $S_1, S_2, ..., S_N$ by a domain decomposition. To decompose a mesh, we use [134] by expanding it into volumetric mesh, which decomposes the mesh by K-Means++ clustering [4] of the eigenvectors of the volumetric mesh Laplacian. It requires no additional precomputation costs since the volumetric mesh Laplacian has been already obtained at the precomputation stage as described in §3.5.

We compute the local generalized eigenproblem of N sub-domains in parallel and reduce the DoFs using the eigenvectors at each subdomain. Next, we iteratively merge two adjacent subdomains by Eq. (3.14), and solve the reduced eigenproblem, and Eq. (3.15) to obtain the eigenvectors of the merged subdomain. This procedure also can be executed in parallel until all the subdomains are merged. Finally, we merge all the subdomains and obtain the approximate eigenvector of the entire structure. The order of merging subdomains is irrelevant in our algorithm because the error caused by suboptimal order will be fixed later in our error correction (§7.2.1). We note that this hierarchical merging procedure is new. We implemented local eigenprobrem solves of each subdomain by a combination of incomplete Lanczos matrix triangulation and QR method.

Error Correction

HCMS is just an approximation method and sacrifices the accuracy for computational efficiency. To correct this error, we introduce the subspace iteration method [7] using reduced mass Gram-Shmidzt process [147]. We set approximated eigenvectors of HCMS as the starting iteration vectors $\boldsymbol{X}_0$ and execute the following iteration $k = 1, 2, 3...$ until it converges.

$$\begin{aligned} Solve\ PCG:\ \boldsymbol{KU} &= \boldsymbol{MX}_{k-1}, && (3.16)\\ \boldsymbol{U} &\leftarrow ReducedMGS(\boldsymbol{U}), && (3.17)\\ \boldsymbol{K}' = \boldsymbol{U}^T\boldsymbol{KU}, &\quad\ \ \boldsymbol{M}' = \boldsymbol{U}^T\boldsymbol{MU}, && (3.18)\\ Solve\ QR:\ \boldsymbol{K}'\boldsymbol{Q} &= \Lambda\boldsymbol{M}'\boldsymbol{Q}, && (3.19)\\ \boldsymbol{X}_k &= \boldsymbol{U} + \Sigma(\boldsymbol{UQ} - \boldsymbol{U}), && (3.20)\\ \boldsymbol{X}_k &\leftarrow ReducedMGS(\boldsymbol{X}_k), && (3.21) \end{aligned}$$

where $ReducedMGS()$ is the reduced mass Modified Gram-Schmidt process to orthogonalize the eigenvectors [147], $\boldsymbol{\Sigma}$ denotes a diagonal matrix in which each diagonal corresponds to the overrelaxation weight of the i-th eigenvalue to accelerate the convergence [8]. We solve the first line Eq. (3.16) by incomplete cholesky factorized pre-conditioned conjugate gradient method with respect to each column vector in parallel, and implement the QR method Eq. (3.19) on GPU.

3.8 Results

3.8.1 Validation of Modal Analysis

In this subsection, we verify the accuracy and computational efficiency of our fast approximate modal analysis. As the ground truth, we used the

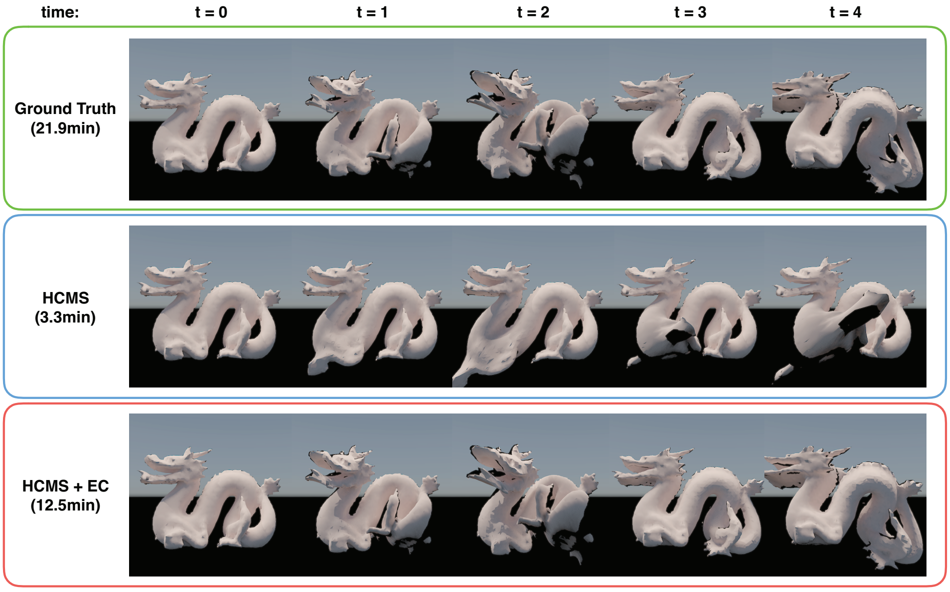
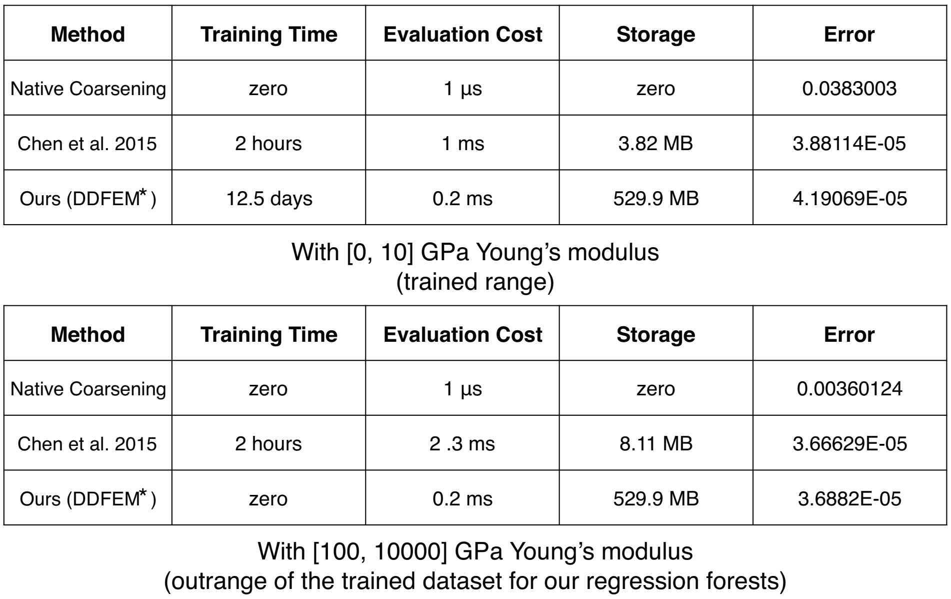


Figure 3.8: Comparison of the deformation of the 7-th modes between HCSM with/without EC and the ground truth. Our error correction algorithm efficiently improve the accuracy within an additional few minutes.

result of the full-DoF standard modal analysis using ARPack (with sufficiently fine-resolution uniform hexahedral mesh). We used CPU: Intel Core i7 2.6 GHz, RAM: 16GB, GPU: NVIDIA GeForce GT 750M as the equipments in §8 excluding the DDFEM trainings. We set the Poisson's ratio as 0.25 and the density as 1.0 kg/m^3 for all experiments.

Data-Driven FEM: We call our data-driven FEM as extended data-driven FEM (DDFEM*) for distinguishing from Chen et al.'s method [15] (DDFEM). We used two regression forests, and three regression trees for each forest, and set the maximum depth of all the trees as 20. We trained each regression forest for DDFEM* by 1 billion entries of the dataset for constructing the tree structures and 10 billion entries of the dataset for training each leaf node (regression function construction). For the dataset generation of data-driven FEM (a sample includes 8 material parameters of detailed $2 \times 2 \times 2$ blocks and the corresponding 8 coarse material parameters), we used 1,000 force directions and sample 5 sample magnitudes in each direction, resulting in 5,000 force samples for each material combination. We seeded the material combinations randomly with [0, 10] GPa of range of Young's modulus using hypercube sampling. The training time took about ten days for data generation, two days for the tree structure training using clusters of 12 computers, and a half day for training the leaves. Finally, our regression forests required 529.9 MB for storage.

We verified the accuracy and the evaluation cost of our data-driven FEM by comparing it with a native coarsening approach (simply averaged material setting of the detailed elements) and Chen et al's method [15]. We prepared



Method	Training Time	Evaluation Cost	Storage	Error
Native Coarsening	zero	1 µs	zero	0.0383003
Chen et al. 2015	2 hours	1 ms	3.82 MB	3.88114E-05
Ours (DDFEM*)	12.5 days	0.2 ms	529.9 MB	4.19069E-05

With [0, 10] GPa Young's modulus
(trained range)

Method	Training Time	Evaluation Cost	Storage	Error
Native Coarsening	zero	1 µs	zero	0.00360124
Chen et al. 2015	2 hours	2 .3 ms	8.11 MB	3.66629E-05
Ours (DDFEM*)	zero	0.2 ms	529.9 MB	3.6882E-05

With [100, 10000] GPa Young's modulus
(outrange of the trained dataset for our regression forests)

Figure 3.9: Comparison of the accuracy of data-driven FEM. Top: The results with [0, 10] GPa range of Young's modulus (trained range of our regression forests). Bottom: The results with [100, 10000] GPa range of Young's modulus (untrained range of our regression forests).

10,000 detailed $2 \times 2 \times 2$ FEM cube blocks with randomized material distribution and applied them to 2,000 random external force samples. We define the error of coarsening as the average of the square distances of the displacements between detailed simulation (ground truth) and coarse simulation over the samples. Because the detailed elements and the coarse element have different numbers of vertices, we measured the error by creating a detailed mesh from the coarse simulation by trilinear interpolation and computing the distance of their displacements. In addition, we defined the evaluation cost as the time for coarsening a $2 \times 2 \times 2$ elements to an element. Since the evaluation cost of DDFEM depends on the database size and the search algorithm, we then prepared the sorted index of the database at precomputation, and searched them by quick search at runtime.

Figure 3.9:Top shows the result of this experiment for the samples with randomly generated Young's modulus setting using [0,10] GPa range. This material parameter range is included in the training dataset for our regression forests. Our regression forests successfully reduce the error on a level with Chen et al.'s method [15] while native coarsening approach causes a large error. Next, Figure 3.9:Bottom shows the result by the samples with the range of Young's modulus [100,10000] GPa that is clearly out of range of the trained dataset for DDFEM*. The result shows our method can also handle this range of inputs and returns good results with no additional training while DDFEM requires additional trainings for new data. It is enabled by our scaling parameter separation algorithm. In addition, the evaluation cost of

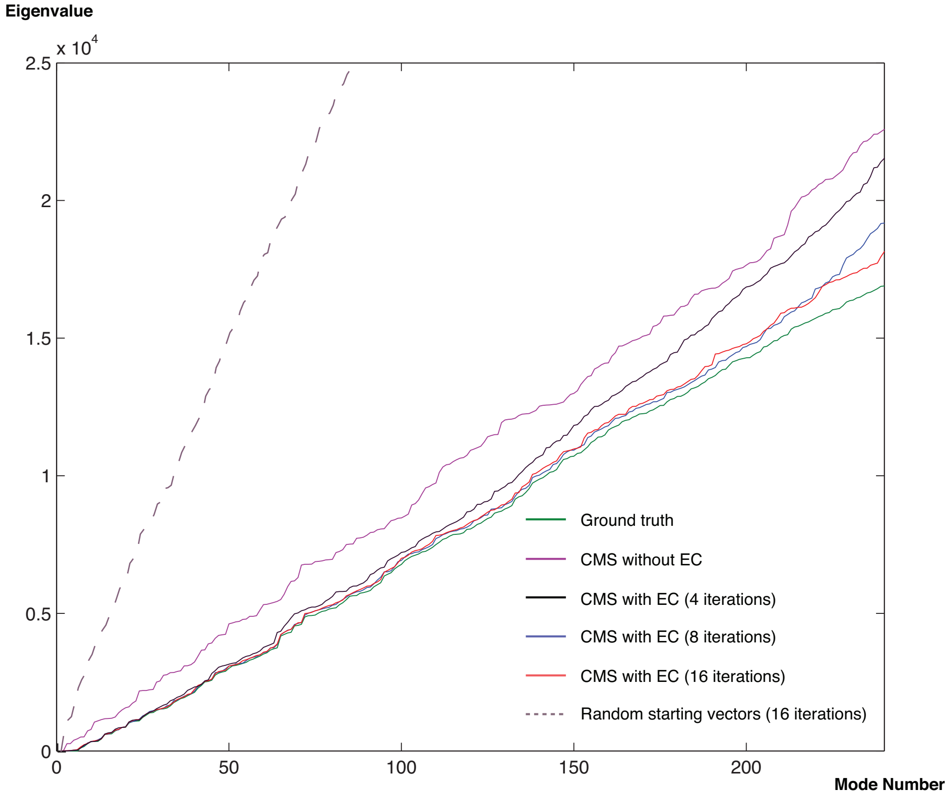


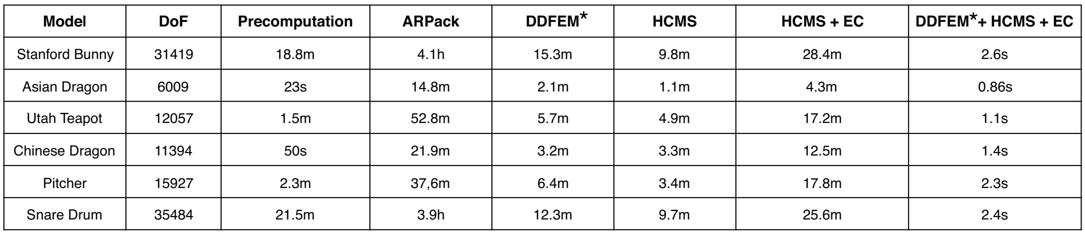
Figure 3.10: The accuracy of HCMS with/without EC. Our error correction algorithm dramatically improves the accuracy within a few iterations. The horizontal axis: the mode number, the vertical axis: the eigenvalues.

our DDFEM* is constant even if the amount of dataset is increased and much faster than DDFEM whose cost is increased in proportion to the amount of the dataset.

Hierarchical Component Mode Synthesis: For the evaluation of HCMS, we decomposed each model of Figure 3.11 into 10∼20 subdomains and hierarchically merged them. When two subdomains are merged, we retained $min(N_{sub}, 512)$ DoFs where N_{sub} denotes the total DoFs of the two subdomains.

Figure 3.10 shows the error comparison of eigenvalue computation of HCMS with/without error correction (EC) and the ground truth using the Chinese dragon model. We can see that our error correction algorithm converges very quickly in only a few iterations and efficiently reduces the error. In addition, the break line in Figure 3.10 shows comparison of the error correction using the result of HCMS and randomized (with $N(0, 1)$ Gaussian) and orthogonalized vectors as the starting vectors. We can see that the iteration with randomized starting vectors does not converge within a few iterations. This shows that the approximate solution of HCMS is a good starting vectors for the subspace iteration method.

Next, we show the comparisons between the modal deformations by standard modal derivatives and our method's (HCMS and HCMS + EC) at the 7th mode in Figure 3.8 for example. The results show that our HCMS can capture



Model	DoF	Precomputation	ARPack	DDFEM*	HCMS	HCMS + EC	DDFEM*+ HCMS + EC
Stanford Bunny	31419	18.8m	4.1h	15.3m	9.8m	28.4m	2.6s
Asian Dragon	6009	23s	14.8m	2.1m	1.1m	4.3m	0.86s
Utah Teapot	12057	1.5m	52.8m	5.7m	4.9m	17.2m	1.1s
Chinese Dragon	11394	50s	21.9m	3.2m	3.3m	12.5m	1.4s
Pitcher	15927	2.3m	37,6m	6.4m	3.4m	17.8m	2.3s
Snare Drum	35484	21.5m	3.9h	12.3m	9.7m	25.6m	2.4s

Figure 3.11: Computation time comparison of modal analysis. We computed the first 256 modes for all model.

rough motions of the elastic object in both lower and higher frequency domains, and the error correction algorithm successfully brings the approximate solution close to to the ground truth within an additional few minutes.

Combination of Extended Data-Driven FEM and HCMS: Figure 3.11 shows the computational times of each modal analysis method using DDFEM*, HCMS with/without EC, and their combination with several models, respectively, while ARPack denotes the standard modal analysis (conventional method). The precomputation column in Figure 3.11 represents the precomputation times taken for each model (The voxelization and the eigenproblem solves for the volumetric Laplacian matrix). The computation times of DDFEM* include the time for the online coarsening procedure. We achieve two orders of magnitudes acceleration with each of DDFEM* and HCMS + EC respectively, and three orders of magnitudes acceleration in total compared to the conventional modal analysis by their combination. The exact computational time using the combination method becomes 0.5$\sim$3.0 secs, which is acceptable for interactive evaluation.

Figure 3.12 shows comparison between two deformation trajectories produced with standard modal derivatives (ground truth) and our fast approximate modal analysis method after applying a unit force impulse at the location pointed by the white arrow in the top thumbnails. We also provide the time series of magnitudes of the displacement at the dragon's nose in Figure 3.12:Bottom. The two trajectories plot quite a similar form, which shows that our method gives good approximation for the conventional approach with much faster operation.

Finally, Figure 3.13 shows the spectrograms of the sound produced by a rigid body physics animation using ground truth and our combination method. Naturally, the spectrogram of the ground truth includes more high frequency components than that of ours because it uses a detailed FEM mesh. However, our result can capture a better portion of the major components enough for our optimization problem.

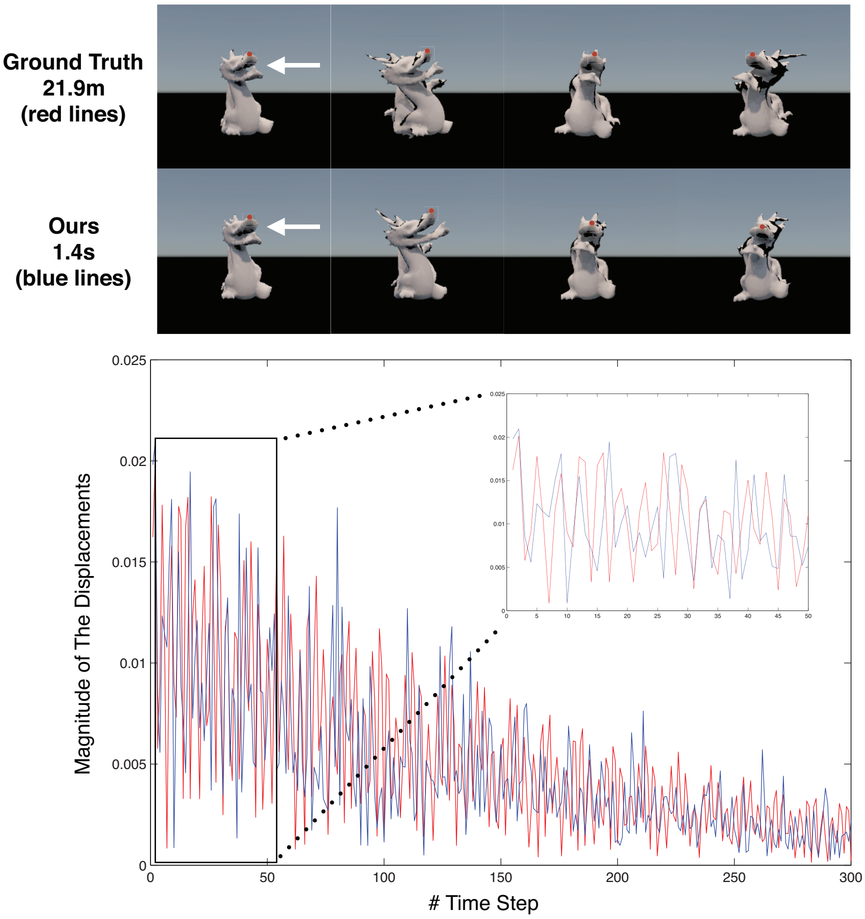


Figure 3.12: Comparison of between two deformation trajectories of the dragon's nose (red circle at the top thumbnails) produced with standard modal derivatives (red) and our method (DDFEM* + HCMS + EC) (blue). The white arrows at the top thumbnails represent the applied force impulse to drive them.

3.8.2 Physically-Based Sound Design

In this subsection, we demonstrate our physically-based sound design framework. For all the examples, we used the results of our fast approximate modal analysis to render the sound. We set the weights in Eq. (3.4) to $w_f = 1.0$, $w_a = 10.0$, $w_r = 10^{-5}$ in our experiment.

Basic Sound Assignment: Figure 3.14 shows an example of assigning two sounds to a frying pan model. The frying pan consists of a handle made of wood and plate made of iron. We stuck the pan at the handle and the plate, and recorded the respective sounds. We used these recorded sounds as input to the system. In this example, 30 extracted target modes were extracted from these two sound clips, and we controlled the first 30 modes of the model excluding the six rigid modes. The two spectrograms at the top row in Figure 3.14 represent the rendered sounds when each position is struck before the optimization. The spectrograms at the middle row are target sounds, and at the bottom row are the results after one minute of optimization. Appar-

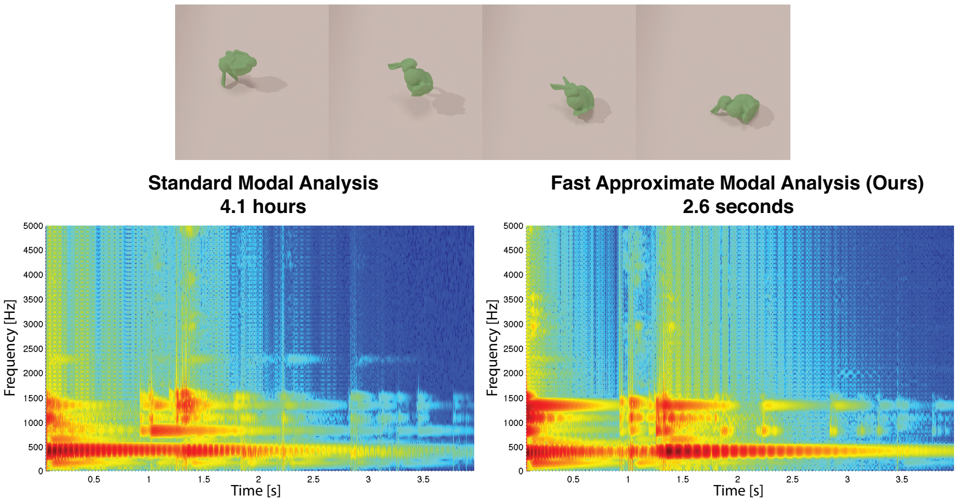


Figure 3.13: Comparison of the modal sound synthesis from a simple rigid body physics simulation between standard modal analysis (left) and our fast approximate modal analysis (right).

ently, the two spectrograms after the optimization closely resemble each target sound. In addition, even if a different position from the one assigned is struck, the sound characteristics of the target sounds near the position is produced in a physically plausible manner (Figure 3.14:Bottom). This result shows that over-fitting is not a serious problem in our optimization. Furthermore, our approach requires less amount of example sounds for designing the sound of an object, which reduces the user's effort. For example, in [132], there is a frying pan example which is similar to our experiment. They used five example sounds to design the sound of the plate alone (except the handle) while we used only one example sound for each part.

Interactive Editing: Next, we demonstrate an example of the interactive editing procedure of physically-based sound using a teapot model. Please see the supplemental video for an interactive demonstration. We first assigned a sound caused by a metal plate being hit to the teapot's body and started the optimization. During the optimization, we checked the intermediate result on the UI view (Figure 3.3) and stopped the iteration when the sound of the object was sufficiently close to the given target sound. In this timing, all the positions indicated sounds similar to the target sound. To append more varied sound properties, we assigned two additional target sounds (two different metal sounds) to the lid and spout one by one. As seen in this example, the user can design the sound of an object while running the optimization and the user's edits are immediately reflected in the simulation within a few seconds. The user can iteratively re-edit the sound property of the object with checking the intermediate results. This type of interactive physically-based sound design workflow has not been presented before.

Imaginary Sound Assignment: Our system allows the user to assign imaginary target sounds to a 3D model (Figure 3.15). In this example, we

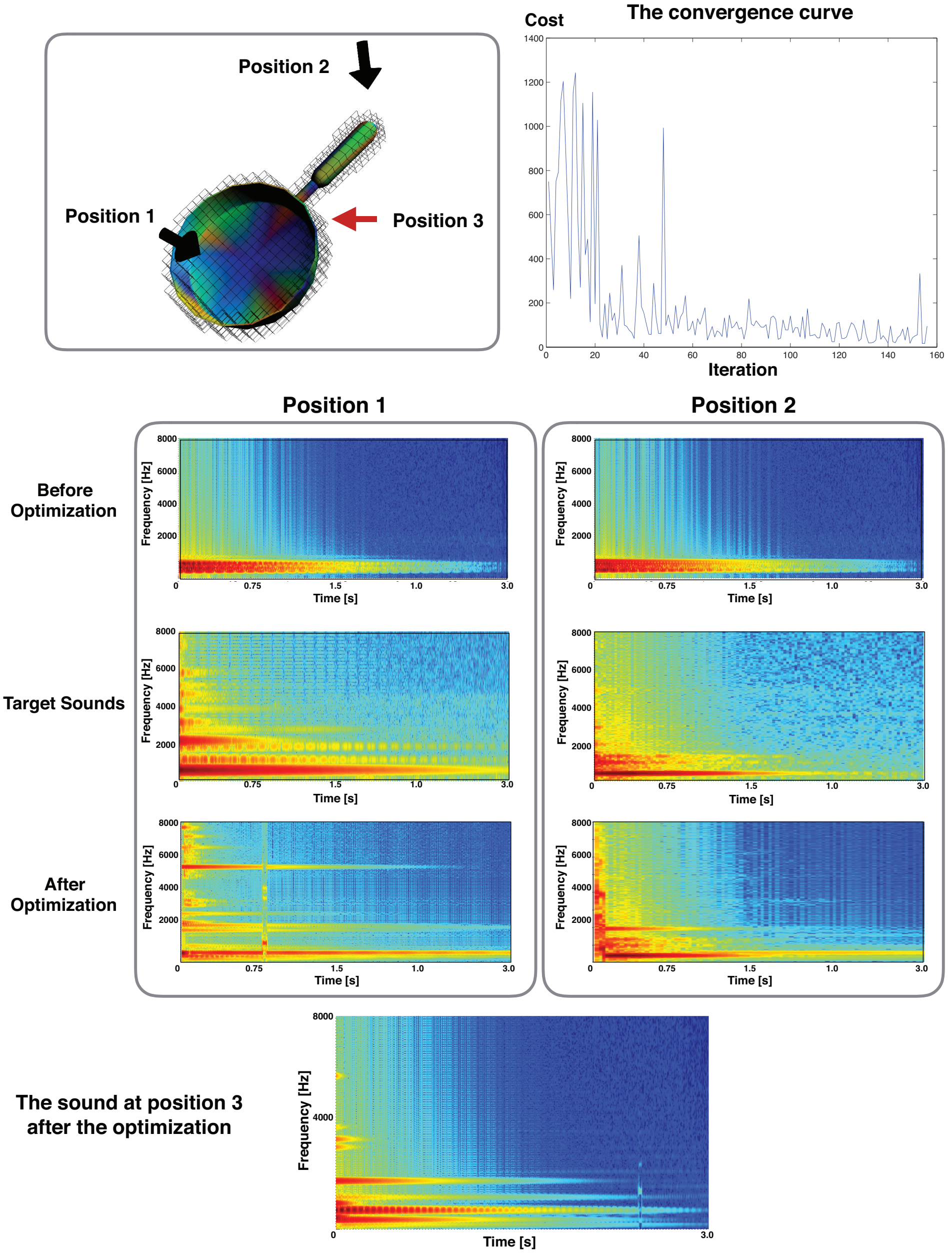


Figure 3.14: The result of assigning two target sounds to a frying pan model after a minute of optimization. We assigned the metal sound to position 1 (plate) and wooden sound to position 2 (handle). Top right shows the convergence curve of the cost (Eq. 3.4)

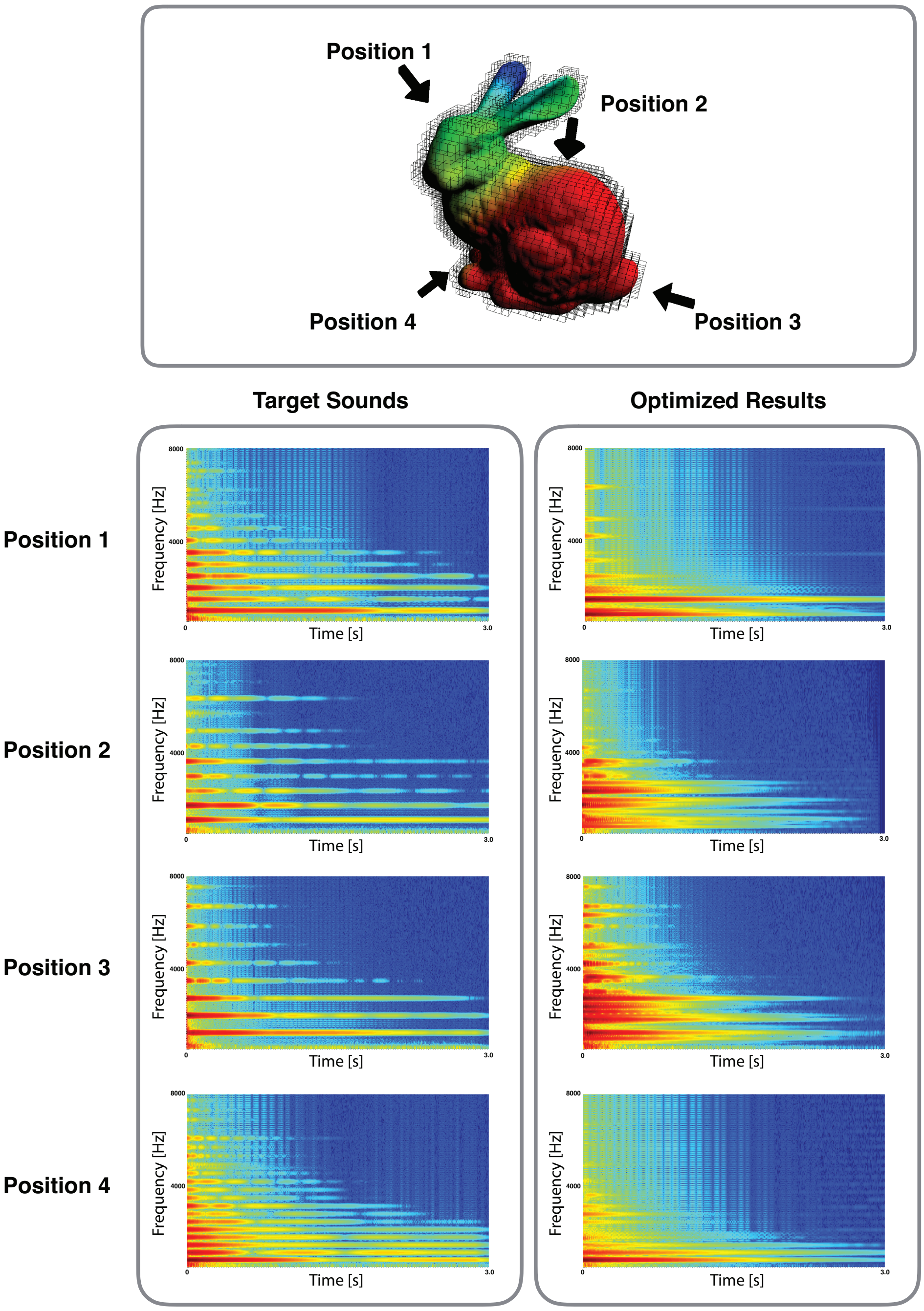


Figure 3.15: The result of assigning four target sounds to stanford bunny model after four minutes of optimization.

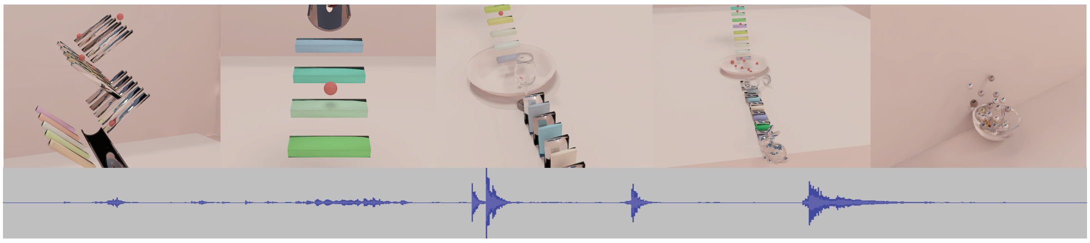


Figure 3.16: The result animation including various objects.

assigned a piano C4 sound to the head of the bunny, a piano E4 sound to the body, a piano G4 sound to the tail, and a piano F3 sound to the leg, and optimized. The result after four minutes of optimization is shown in Figure 3.15 by spectrograms. Although an object that has such sounds does not exist in the real world, our system produces a physically convincing result. This is also an advantage of our approach compared to the previous method [111].

A Complicated Scenario Example including Various Objects: Finally, we demonstrate a complicated scenario involving several objects as shown in Figure 3.16. We include this animation scene in the accompanying video. In this scene, the sounds of the all objects are designed by our system, and all sounds are triggered automatically by rigid body simulations. It took 30 minutes for us to design sound of all the object in the scene using our tool, one week for rigid-body simulator setting, and three days for visual rendering.

3.9 Conclusion and Future Work

We presented a novel example-based design framework of physically-based sound of a 3D model using material optimization. It allows physically-based sound design without considering unintuitive physical parameters for the user. This achieves practical design workflow which was difficult in previous methods. In addition, our system runs at an interactive rate that enables the user rapid try and error design procedure. This is achieved by our novel fast approximate modal analysis that consists of data-driven online coarsening of the mesh and hierarchical component mode synthesis with efficient error correction. We demonstrated our framework provides the user to intuitive design workflow of the sound of an object with a set of examples.

However, several limitations are observed in our work, which remain to be addressed in future work. A critical limitation is that our sound design technique cannot be used for fabrication because we employ continuous material optimization, which simulates materials that do not exist in the real world. Although the usage of combinatorial optimization using existing materials can also be considered, it is still difficult to make an object consist of various materials that have a large range of material parameters seamlessly using existing fabrication tools. Second, our current optimization does not consider the effects of sound radiation and propagation. However, radiation and propagation effects are important for sound design and thus we plan to extend our method to support them using Li et al. [79]'s technique in the future.

We only optimized Young's moduli in this paper. which limits the reproducibility of the example sounds. As an improvement, Poisson's ratios could be treated similarly by converting Young's moduli and Poisson's ratios to the 2-dimensional space of Lame' parameters [119] and performing the optimization in this linear space. However, the dimension of the feature vector for inputting data-driven FEM increases, and it could reduce the training efficiency of our regression forests. There is a similar problem for treating the densities. To treat such a large parameter space with a machine learning technique remains as future work.

Naturally, our fast approximate modal analysis pipeline can also be used for deformable animation. Applying our approach to large deformable simulation or combining it with [147] could be also useful and promising work. Finally, we use voxel elements. The fineness of the details of the model depends on the voxel resolution. To address this, adaptive coarsening could be useful. However, how to treat such an adaptive mesh with machine learning technique is non-trivial and remains as future work.

Chapter 4

Controlling Lyrics and Melodies for A Singing Voice Synthesizer in Realtime

4.1 Introduction

The use of a singing voice synthesizer such as VOCALOID [67] has become very popular. The singing synthesizer generates human singing voice by inputting melody (pitch) and the corresponding lyric (phoneme). However, there is little precedent of live improvisational performance using real-time singing voice synthesis even though there is a huge demand for it. This is mainly because it is very difficult to input song lyrics at a real-time rate; this is the problem we want to address in this paper.

A possible approach is to use automatic fitting of the predefined original lyrics to the melody currently being played using melody matching. However, this approach has two problems. First, players often modify the melody significantly including addition of grace notes and change of order in a live improvisational performance. Second, the same melodies often appear repeatedly in a song with different lyrics, making it very difficult to find the appropriate lyrics from melody alone in improvisational performance.

Another possible approach is to use speech recognition to input lyrics. This allows the user to improvise arbitrary lyrics during performance, but also presents several problems. First, recent popular speech recognition techniques are optimized for recognizing continuous speech as a whole, rather than for recognizing individual characters in a song separately for timed performance. Second, latency is inevitable in speech recognition, but is not acceptable for real-time musical performance. Finally, it is difficult for the player to listen to his or her own performance while vocalizing.

There are a few experimental systems that allow the user to input arbitrary Japanese lyrics during live performance using a combination of vowel and consonant keys [93][144]. However, they require the user to press two keys simultaneously to input a character, making it difficult to play fast songs.

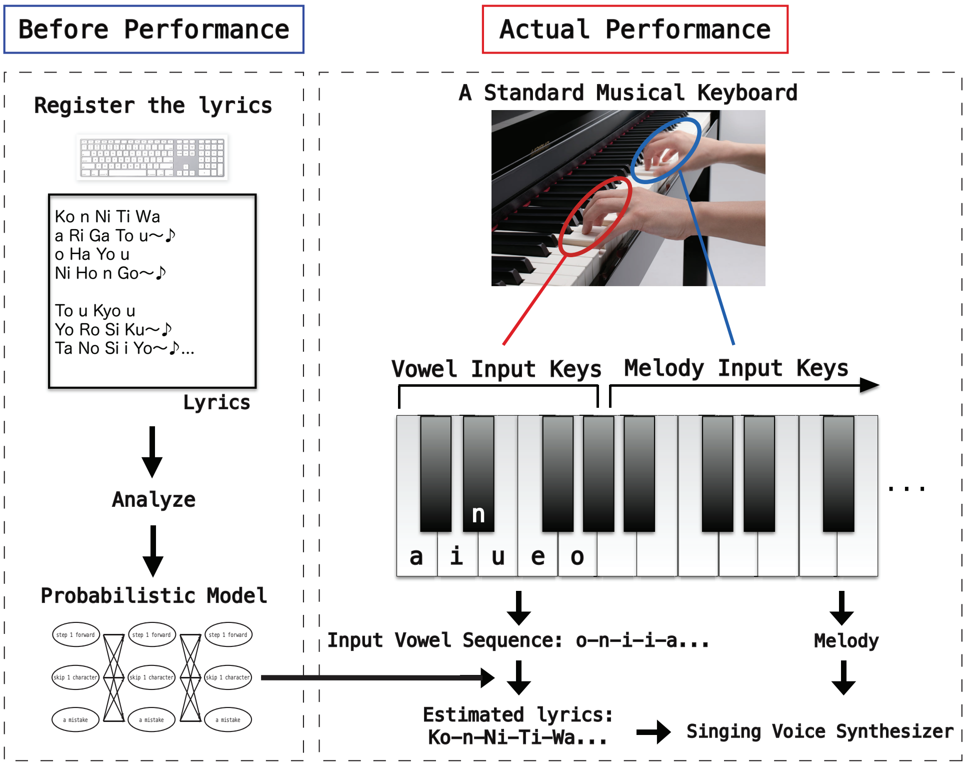
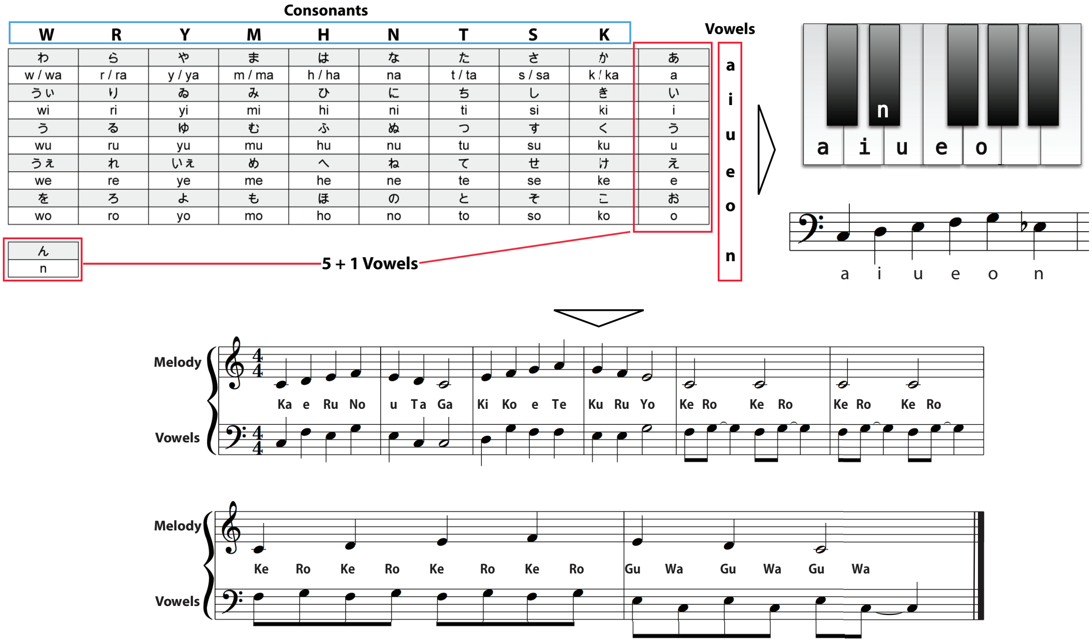


Figure 4.1: An overview of the proposed system. Our system consists of two steps, lyric registration step and actual performance step. At the lyrics registration step, the user registers the lyrics of the songs, and the system analyzes it. At the actual performance step, the user simultaneously inputs the vowel sequences and melodies using a musical keyboard, and the system estimates the plausible lyrics from them and synthesizes singing voice sounds.

To address these problems, we propose to use a vowel keyboard to input the lyrics during live improvisational performances (Figure 4.1: right). In our system, the user inputs the lyrics with one hand using a vowel keyboard and the melodies with the other hand using a musical keyboard simultaneously. Our system allows the user to modify the melodies of a song freely and to pick an arbitrary portion of predefined lyrics during a live performance.

Our system is designed for Japanese lyrics. In Japanese, a character consists of a consonant and a vowel (Figure 4.2: left). Hence, multiple Japanese characters match a given vowel. However, we can identify the most plausible character sequence in the predefined lyrics by finding the corresponding vowel sequence using a probabilistic alignment technique (Figure 4.1). Specifically, our system automatically finds a portion of the predefined lyrics whose vowel sequence matches well with the vowel sequence being input by the player. We use a Hidden Markov model for alignment.

There are only five vowels in Japanese, “a", “i", “u", “e", and “o". We also use a special character “n", hence we use six keys to input lyrics. This makes it possible to input vowels very rapidly without moving the hand to other locations in contrast to other methods that use many keys to input



Consonants

W	R	Y	M	H	N	T	S	K	
わ	ら	や	ま	は	な	た	さ	か	あ
w / wa	r / ra	y / ya	m / ma	h / ha	na	t / ta	s / sa	k / ka	a
うぃ	り	ゐ	み	ひ	に	ち	し	き	い
wi	ri	yi	mi	hi	ni	ti	si	ki	i
う	る	ゆ	む	ふ	ぬ	つ	す	く	う
wu	ru	yu	mu	hu	nu	tu	su	ku	u
うぇ	れ	いぇ	め	へ	ね	て	せ	け	え
we	re	ye	me	he	ne	te	se	ke	e
を	ろ	よ	も	ほ	の	と	そ	こ	お
wo	ro	yo	mo	ho	no	to	so	ko	o

ん
n

5 + 1 Vowels
Vowels
a
i
u
e
o
n
a i u e o
n
a i u e o n
Melody
Vowels
Ka e Ru No u Ta Ga Ki Ko e Te Ku Ru Yo Ke Ro Ke Ro Ke Ro Ke Ro
Melody
Vowels
Ke Ro Ke Ro Ke Ro Ke Ro Gu Wa Gu Wa Gu Wa

Figure 4.2: In Japanese, a character consists of a consonant and a vowel (left). For example, "Ka" is the combination of "K" and "a", "Su" is the combination of "S" and "u". There are 5 vowels and a special vowel (n) in Japanese. We mapped these vowels on a piano keyboard (center). Because of this, in our system, the lyrics can be represented as a standard musical score as the left hand part (right).

lyrics. Additionally, by mapping the vowel keyboard onto a traditional musical keyboard, one can represent lyrics as a standard musical score (Figure 4.2: right), enabling the user to practice the skills more easily.

One possible criticism is that one can use only the predefined lyrics in our system. It is not possible to compose completely novel lyrics during performance, However, in real improvisational performance, it is actually very rare to see the singer composes completely new lyrics during performance. They usually improvise novel melodies (or a little modified melodies from composition) using given lyrics at live performance, which can be handled using our method.

In addition to improvisational live performance, our target includes several situations. For example, when a composer (specifically non-vocalist) makes a melody for a given lyric, using our system, he or she can can try the idea more quickly than offline composition. An another example is a music arrangement of an existing song. An arranger can try the idea more easily using our system.

The contributions of this paper are as follows.

1. Our system allows the user to perform regular Japanese songs at the original tempo, including very high-speed songs.

2. Our system enables the user to rearrange the pieces of pre-defined lyrics

in a live improvisational performance.

3. We introduce the analogy of a musical score alignment technique to lyrics alignment.

4. We examine the feasibility of the method with performance evaluation and user studies.

4.2 Related Work

In this section, we describe the related works for controlling texts and melodies of a song for a singing voice synthesizer in realtime. In addition, we also review existing studies for accelerating text entry and related musical score alignment techniques.

4.2.1 Live Performance using Human-Like Synthesized Voice

There is a long history of live performance by synthesizing human-like voices. One of the oldest example uses analogue synthesizers that is called “Choirs voice" [22]. Singing voice synthesizer have also essentially originated as this choir voice. Traditional choirs voices are made by imitating human voices using subtractive synthesis that making a sound by engraving a signal source which has rich overtones by filters, or additive synthesis that generates various sounds by combining many sine waves [21]. Such choir voices have been widely used for live performance by many musicians. However, the sounds are not realistic, and limited to generate one or two phonemes during the performance.

An another origin of singing voice synthesizer is vocoder [27]. Vocoder is a widely used sound effecter for live performance that appends speaking like (robot voice) effect to other sounds (e.g., guitar, synthesizer) by modulation. It uses two input sources: modulator and carrier signals. The carrier provides the modulated signal (from guitar or synthesizer input) that determines timbre to talk and pitch while the modulator provides phoneme to talk by musician’s voice from microphone input. This allows musical instruments’ sound to speak by musician’s own voice. However, the vocoder sound is just a human like talking effect and not a realistic human voice.

As the development of digital synthesizers, modern choir voice uses PCM (pulse width modulation) synthesis alternatively. PCM uses multiple recorded sound samples and switch them by controllers. Thus, it provides realistic human singing voice for choir voice. A significant advantage of this method is allowing to switch multiple phonemes (at most 4∼5 phonemes) by some controllers (e.g., velocity, modulation wheel) during the performance. This enables musicians JAZZ scat like improvisational live performance [143] by a synthesizer, and the ability is currently implemented in many contemporary musical keyboards. However, these synthesizers can provide only a few major

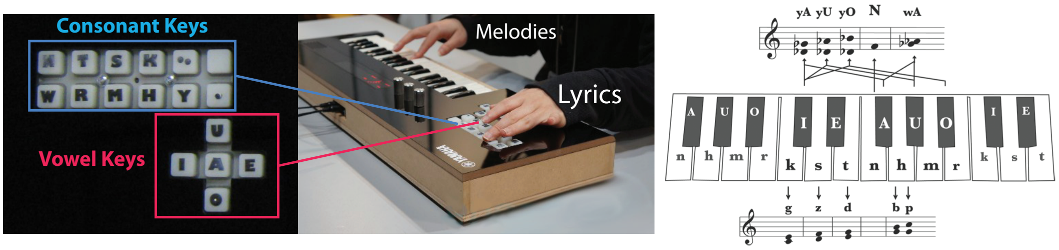


Figure 4.3: Yamamoto et al. [139] (left) and Formant.Bros' system [88] (right).

phonemes widely used for scat because of the difficulty of controlling many phonemes, which still limit the expressiveness of the choir voices.

4.2.2 Realtime Lyric Control for Singing Voice Synthesizer

Yamamoto et al. [144] used a combination of a dedicated special keyboard to input lyrics used by the left hand and a standard musical keyboard used by the right hand for improvisational performance (Figure 4.3: left). The lyrics keyboard is designed for Japanese, consisting of ten consonant keys and five vowel keys placed to fit the left hand. The user inputs a character by pressing a combination of a consonant key and a vowel key. However, it is very difficult for a typical player to press correct multiple keys simultaneously during a live performance.

Formant Bros. [93] assigned lyrics input keys to a common musical keyboard (Figure 4.3: right). A character can be input using the triplet three-key combination of a pitch key, a consonant key, and a vowel key. The benefit of this approach is that it enables description of lyrics and melodies as a standard musical score, making the method easy to learn. However, the consecutive triplet chord input is very difficult even for professional pianists. Thus, the approach remained at the level of playing very slow nursery rhymes, in contrast to regular songs played at a realistic speed (we assume the range of tempo of regular songs is about 50∼200 BPM [beats per minute]).

HANAUTAU [125] uses pitch detection from the user's voice inputted by microphone for melody and lyrics typed with both hands using a common QWERTY keyboard. However, using a QWERTY keyboard does not provide input at a speed sufficient to play common music adequately.

A case has been made to use a Flick text input method [105] for live performances using real-time singing voice synthesis to input lyrics. Although the Flick text input method is a very fast text entry method, , it still can't achieve sufficient input speed for singing a song. Additionally, because it requires two-step control (push and slide), it is difficult to adjust the timing to the music using that method.

DiVA [25] uses CyberGlove and several sensors and measured the hands gestures to control the lyrics. The gestures are trained and trigger a neural network with a given gestural language that associates one posture for each phoneme of English. However, the gestural control is difficult for fast songs.

Cantor Digitalis [31] has been used in several musical improvisations using singing voice synthesizer by multi-touch tablet. Their alphabet control is limited in only a few vowels (formants) and can't output the most characters including consonants as a language. Then, their system is inadequate for performing the lyrics of common songs. We address this issue.

4.2.3 Text Entry

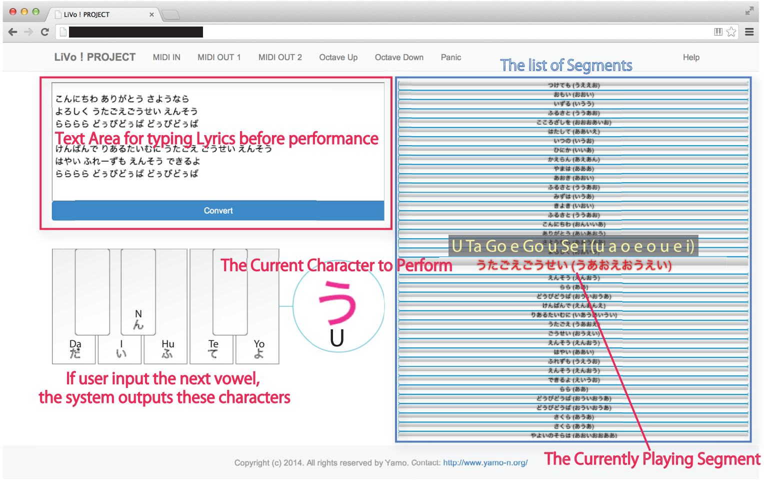
Text input predictions [33] [117] and query word suggestions [63] [154] are not applicable to our target purpose, because they pose two problems for our target. First, there is no way to input the first character of the word the user wants to input at a real-time rate. Second, there is no way to select a candidate in real-time.

Many studies have been conducted regarding word completion [14] [140] from the user's ambiguous input. The word completion methods modify or correct word input by the user including mistypes to form a plausible word. However, these approaches use lazy evaluation. Lazy evaluation estimates the correct word retroactively after the user inputs several words. Thus, it can't be used for our target, which requires outputting the characters individually.

4.2.4 Musical Score Alignment

Our algorithm for controlling a singing voice synthesizer is analogous to a probabilistic musical score alignment technique. Musical score alignment techniques estimate the current playing position of given music (audio or MIDI stream) in a musical score in a database, and use it for various applications such as generating musical accompaniments [100], and displaying the musical score using auto scroll [60] [78]. Unlike these methods, our alignment technique estimates the current playing lyrics position.

Recent studies regarding musical score alignment are categorized into two approaches. The first approach is to solve the problem by minimizing some metrics representing how two musical signals differ at each time [145]. This approach is vulnerable to the uncertainty of the user's performance including mistakes, tempo change, or other musical expressions. The second approach is to use probabilistic models [86] [100]. This approach is advantageous because it is robust against such uncertainties in the user's performance. We also use a probabilistic model, but for vowel sequence not for melody. The significant difference from realtime musical score alignment (specifically automatic accompaniment generation) is the timing control. The musical accompaniment generation system outputs the corresponding accompaniment for user's performance. In this case, the user does not control the playing timing of system's output directly. On the other hand, the user directly control the voicing timing in our system.



[redacted]

こんにちわ ありがとう さようなら
よろしく うたごえごうせい えんそう
ららららら どぅびどぅば どぅびどぅば
けんばんで りあるたいむに うたごえ ごうせい えんそう
はやい ふれーずも えんそう できるよ
ららららら どぅびどぅば どぅびどぅば

つけても (うええお)
おもい (おおい)
いずる (いうう)
ふるさと (ううあお)
こころざしを (おおおあいお)
はたして (ああいえ)
いつの (いうお)
ひにか (いいあ)
かえらん (あえあん)
やまは (あああ)
あおき (あおい)
ふるさと (ううあお)
みずは (いうあ)
きよき (いおい)
ふるさと (ううあお)
こんにちわ (おんいいあ)
ありがとう (あいあおう)
さようなら (あおうああ)
よろしく (おおいう)
うたごえごうせい (うあおえおうえい)
えんそう (えんおう)
らら (ああ)
どうびどうば (おういおうあ)
けんばんで (えんあんえ)
りあるたいむに (いあうあいうい)
うたごえ (うあおえ)
ごうせい (おうえい)
えんそう (えんおう)
はやい (ああい)
ふれずも (うえうお)
えんそう (えんおう)
できるよ (えいうお)
らら (ああ)
どうびどうば (おういおうあ)
どうびどうば (おういおうあ)
さくら (あうあ)
さくら (あうあ)
やよいのそらは (あおいおおああ)

LiVo ! PROJECT
LiVo ! PROJECT
MIDI IN
MIDI OUT 1
MIDI OUT 2
Octave Up
Octave Down
Panic
Help
The list of Segments
Text Area for typing Lyrics before performance
Convert
The Current Character to Perform
U Ta Go e Go u Se i (u a o e o u e i)
N
ん
Da
だ
I
い
Hu
ふ
Te
て
Yo
よ
う
U
If user input the next vowel,
the system outputs these characters
The Currently Playing Segment
Copyright (c) 2014. All rights reserved by Yamo. Contact: http://www.yamo-n.org/

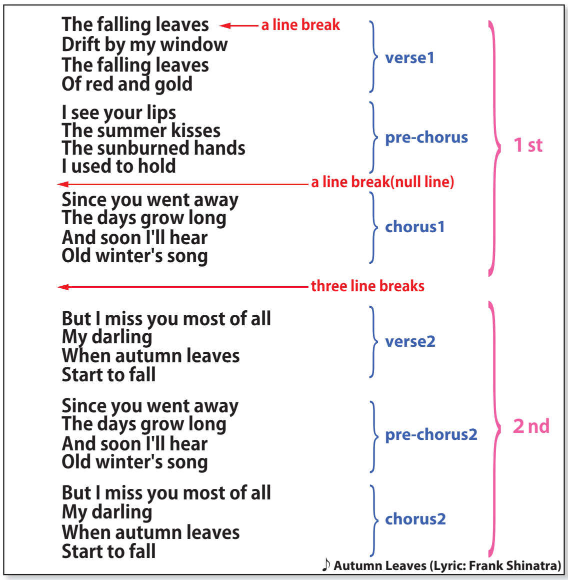
Figure 4.4: The user interface view of the LiVo system on a web browser. The user enters the lyrics in the top left text area before performance.

4.3 User Interface

An overview of our system is shown in Figure 4.1. Our system requires two steps. The first step is lyrics registration before performance. The second step is actual performance. At the lyrics registration step, the user registers the intended lyrics to be used. The user can register multiple lyrics at a time. At the performance step, the user simultaneously inputs vowel sequences using a vowel keyboard and melodies using a musical keyboard. The system estimates the plausible lyrics from the vowel sequences and synthesizes singing voice sounds. Note that the system does not use a melody sequence for estimation.

The vowel keys are assigned to a portion of a standard musical keyboard. In our prototype, each key mapping is set as “a" to C2, “i" to D2, “u" to E2, “e" to F2, “o" to G2, “n" to D#2, to fit all the keys in a palm (Figure 4.2: center). This makes it possible to represent a vowel sequence as a standard musical score, making it easier for players to practice the performance. In addition, the musical score for our system requires only monophonic phrases of six vowel keys, which is much simpler than that of Formant Bros. Actually, the musical score for our system is much easier than popular piano scores such as J.S.Bach and Chopin.

Our system doesn’t require the user to input the vowel sequences strictly in the order of the original lyrics, because the system estimates the plausible lyrics using a probabilistic model. This allows the user to jump to arbitrary positions in the lyrics including backtracking. Additionally, our system allows the user to make mistakes, freeing the player from paying excessive attention to vowel input. When the user jumps the position of the lyrics, the system would output a few wrong characters until following the user. However, these wrong characters at least have correct vowels, which limits the level of discomfort.



The falling leaves
Drift by my window
The falling leaves
Of red and gold

I see your lips
The summer kisses
The sunburned hands
I used to hold

Since you went away
The days grow long
And soon I'll hear
Old winter's song

But I miss you most of all
My darling
When autumn leaves
Start to fall

Since you went away
The days grow long
And soon I'll hear
Old winter's song

But I miss you most of all
My darling
When autumn leaves
Start to fall

♪ Autumn Leaves (Lyric: Frank Shinatra)

Figure 4.5: The text format for registering lyrics before performance. We use English here for explanation. The real system only takes Japanese alphabets as input.

The timing control is complicated because two keys must be pressed in a coordinated way. Our current implementation is as follows. If a pitch key has already been pressed, the system begins a new voice when a new vowel key is pressed. However, if no pitch key has been pressed, the system does not begin a new voice, when a new vowel key is pressed. If a vowel key has already been pressed, the system begins a new voice, when a new pitch key is pressed. If no vowel key is pressed, the system begins a new voice with the last vowel input by the user, when the user presses a new pitch key. We choose this asymmetric scheme because pitch keys serve as the main control, with vowel keys serving as a modifier. Note that previous systems [93] [144] begin a voice only when 3 keys (vowel, consonant, and pitch) are pressed together, creating difficulty in producing fast real-time performance.

4.4 Technical Details of Lyric Alignment

This section explains how we estimate the position that the user wants to perform in the lyrics from the vowel sequences input by the user. In the registration step, the system analyzes the lyrics entered by the user, and constructs a data structure to be used in the performance. In the actual performance step, we estimate the most plausible lyrics using a Hidden Markov Model (HMM) that encodes the behavior of the movements between consecutive characters in the lyrics and jumps during performance. We search the end point of the Viterbi path in this HMM using multi-agent search to find the best matching lyrics for the given vowel sequence. Note that the estimation solely depends

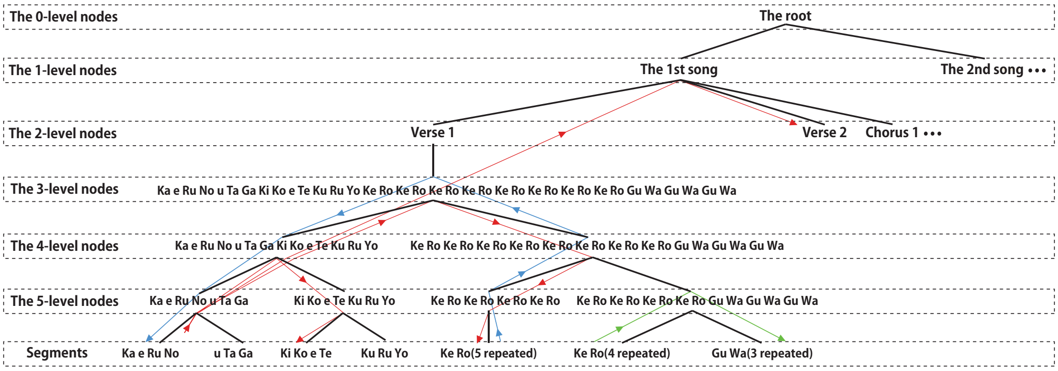


Figure 4.6: The hierarchical tree structure of the lyrics segments. The system constructs this structure according to the annotations of the user. The black lines denote the edges. The red, blue and green arrows denote possible jump movements by the user in this tree.

on the vowel input, and does not use melody information at all.

4.4.1 Lyric Registration Step before Performance

Figure 4.4 shows the user interface view of the system. The user types the lyrics in the the top-left text area and presses the “ convert ” button to finish. The text format for typing lyrics is shown in Figure 4.5. The system requires the user to annotate rough structures of the song (e.g., repeating, verse, chorus, or several phrases) manually using line breaks. With more line breaks inserted, the system interprets the point as a larger compartmental boundary. After the user completes text entry, the system decomposes the text into morphemes (a sequence of characters) using morphological analysis. We define each delimited morpheme as a “segment".

Commonly, a song has the hierarchical structure that consists of musical phrases [137]. We construct a hierarchical tree structure (called lyrics tree) of the segments using the user’s annotations (Figure 4.6). The lyrics tree is constructed by dividing the array of segments recursively from a large structure to a small structure according to the annotations (the smallest structure is a segment). The lyrics tree is used for determining the probabilities of each movement in the lyrics as described in the next subsection.

4.4.2 The User’s Behavior Model for Lyric Movements

We model how the user moves from one character to another character in the lyrics during a performance using a Hidden Markov Model (HMM) (Figure 4.7), incorporating various assumptions regarding typical movements. For example, progression to the next character is more likely than a jump to a distant location, and a jump to the head of a sentence is more likely than a jump to the middle of a sentence. We model this behavior using an Ergodic Hidden Markov Model [87] as follows:

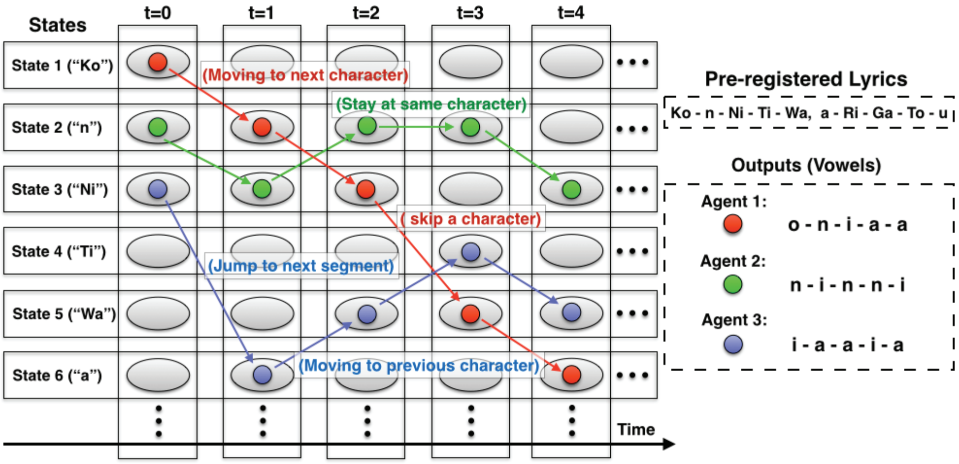


Figure 4.7: We model the user's movement between two characters in the lyrics during performance as Hidden Markov Model (HMM). A position in the pre-registered lyrics becomes a state.

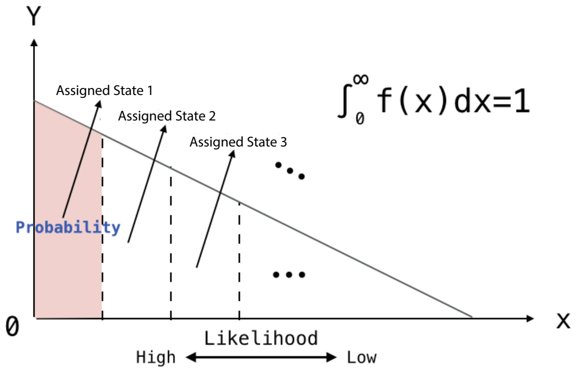


Figure 4.8: Each state transition probability of HMM is determined manually by the authors. we prepare a monotonic decreasing distribution function, and assign the divided area to them according to the likelihood of kinds of movement.

$$p(Z(1)...Z(T)|,\pi,\tau) = \pi\left(Z(1)\right)\prod_{t=1}^{T-1}\tau\left(Z(t), Z(t+1)\right). \tag{4.1}$$

Here $Z(t)$ represents the state at the time t, $\pi(Z(1))$ denotes the probability of being $Z(1)$ as the initial state, $\tau(Z(t), Z(t+1))$ represents the state transition probabilities from state $Z(t)$ to state $Z(t+1)$, and T represents the current time step. This HMM contains the unobserved states that represent the kinds of movements in the lyrics (such as moving to the next character, skipping one character, and making a mistake). In our HMM, an unobserved state corresponds to a position in the lyrics and a transition between unobserved states corresponds to a movement in the lyrics. Each transition generates a specific observable symbol deterministically, which is a vowel of a character. For example, if we have a pre-registered lyrics "Ko-n-Ni-Ti-Wa (segment 1), a-

Ri-Ga-To-u (segment 2)", the HMM produces an observable symbol "o" when moving to a state "Ko" or "To".

Each state transition probability of our HMM (probability of a movement in the lyrics) is computed according to the kind of movement. For example, moving to the next character is more likely than a jump to a distant location, and a jump to the head of a sentence is more likely than a jump to the middle of a sentence. The system first enumerates all the possible movements from the current state and sorts them by the likelihood by the kind of movement. For example, in the above example, if you are at "n", possible destinations are "Ni" (next character), "Ti" (skip a character), "n" (stay at same character), "Ko" (previous character), "a" (jump to the head of the next segment) in the order of likelihood. The system then computes the probability of a movement using a monotonically decreasing linear function that takes the position in the sorted list as input and returns probability as an output (Figure 4.8).

The likelihood of a movement is computed by traversing the lyrics tree. An example is shown in Figure 4.6. When the current state is one of the character at a leaf node, it can move to the next character within the same leaf node, or move up to parent nodes and then come down to some other leaf node like red, bule and green arrows in Figure 4.6. Each step movement in the tree is associated with a certain cost (e.g. moving to a next character has lower cost than moving to a higher level), and the system accumulates these costs during the traversal.

4.4.3 Estimation of The Performed Position in Lyrics during Performance

The most plausible position the user wants to perform in the lyrics can be computed as the end point of the Viterbi path that gives the highest accumulated state transition probability (minimum cost) among all the possible paths in the HMM. The Viterbi path is also required to output a vowel sequence that matches with the user inputs. We search this path using a multi-agent search algorithm [32].

The multi-agent search uses multiple interacting intelligent agents for finding the minimum cost path. Each agent is associated with a state in the HMM (a position in the lyrics), and moves to the next state according to the HMM (color circles in Figure 4.7). Since multiple destination states exist for a state, the system generates multiple copies of an agent and associates a copy with each destination state. However, if the movement is an impossible one, the system discards the copy. Each agent is scored by the accumulation of the state transition probabilities between the respective pairs of HMM states passed. For example, the score of the agent 1 in Figure 4.7 is determined by using $\pi_0^1 \times \tau_{12} \times \tau_{23} \times \tau_{35} \times \tau_{56}$, where π_0^1 denotes the initial probability of state 1 and τ_{ij} denotes the state transition probability from state i to state j. If the score of an agent becomes less than a threshold, the system destroys the agent.

Additionally, if multiple agents are reached at the same position in the lyrics, the system retains only the agent with the highest score. This procedure corresponds to a the pruning process in dynamic programming. Finally, we obtain the optimum position by selecting an agent has the maximum score at the current time.

4.5 Evaluation

4.5.1 Implementation

Figure 4.10: left shows the system setup for LiVo. The software for the system is written in Javascript, runs on web browser. We used YAMAHA NSX1-board for a singing voice synthesizer. The MIDI keyboard and the singing voice synthesizer are connected to the computer by USB, and the web browser communicates with them by Web MIDI API.

4.5.2 Accuracy of The Alignment Algorithm

First, we examined robustness against mistakes. Table 1 shows the song list used for our experiments. The original vowel sequences are defined as the vowel sequences extracted from the lyrics of the original songs. We generated random error vowels by the uniform sampling from six vowels (“a", “i", “u", “e", “o", and “n"), and inserted them into random positions between the two vowels of the original vowel sequences. The inserted positions are selected randomly according to an uniform distribution ranging between one and the total number of characters in the song minus one. Note that we prohibited inserting more than three error vowels sequentially. We inputted this vowel sequences containing error vowels into our system as a source and examined the error rate of the output characters compared to the original lyrics.

The relationships between the numbers of inserted errors and the output error rate are shown in Figure 4.11. Although over 30% errors cause a fatal decrease in accuracy, the system maintains more than 90% accuracy for 10∼20% errors. These results show that the system is reasonably robust against errors in vowel input.

As a second experiment, we examined robustness against jumps to irregular destinations in the lyrics. The songs used for this experiment are the same as those used for the previous experiment. We randomly rearranged the lines in the registered lyrics (see Figure 4.5), and used their vowel sequences as input. When the input vowel sequence moves to the next randomly rearranged line, the system first makes several errors because it expects the next line in the original lyrics. However, the system eventually identifies the correct line after observing several incoming vowel keys. We counted the maximum number of incorrect output characters before identifying the correct line.

The result of this experiment is shown in Table 1 as following capability.

Table 4.1: The song list used for experiments. The following column represent the numbers of output wrong characters after jumping the positions. These numbers represent higher abilities as lower number.

Title	Author	Characters	Following
1. Senbon Sakura	KurousaP	616	3
2. World is Mine	supercell	628	2
3. Melt	supercell	432	2
4. HatsuneMiku no Shoushitsu	BousouP	1373	4

The result shows that robustness against jumps to irregular destinations is strongly affected by the total number of characters in the lyrics. However, even for “HatsuneMiku no Shoushitsu", which has the most characters in our experiments, our system successfully finds the correct segments within a few characters, which is shorter than a single morpheme. This result shows that the system is reasonably robust against improvisational jumps during performance.

4.5.3 Playability

We examined the playability of our system by a professional pianist. We selected a famous Japanese song “SenbonSakura" (Author: KurousaP) for performance. This song is one of the Japanese songs with the fastest tempo, and contains very fast movements between the characters in the lyrics. We picked this song as a stress test for the system. The time for practice was one week, one hour per day.

A portion of the actual performance scene for this song after practice is shown in the supplemental video. The total length of the performance was four minutes, five seconds. The video shows that this song can be performed at the original tempo. The pianist adds several musical expressions including ad-lib modification of the melody. These expressions can’t be performed using any existing methods. Additionally, the system outputted plausible lyrics, even though the pianist often made mistakes and improvisational changes.

In addition, we recruited five participants and played the recorded sound of this performance for them. We requested them to report the number of times they felt an unnatural singing voice sound. No one reported this more than three times.

Furthermore, some users including authors have tried to play Formant.Bros’ system [93] and Yamamoto et al.’s system [144] for the comparisons of the playability. However, no one was able to play the song used in our experiment with their systems at the same tempo. This shows that it is almost impossible to play such a fast song with existing systems.

4.5.4 Workshop

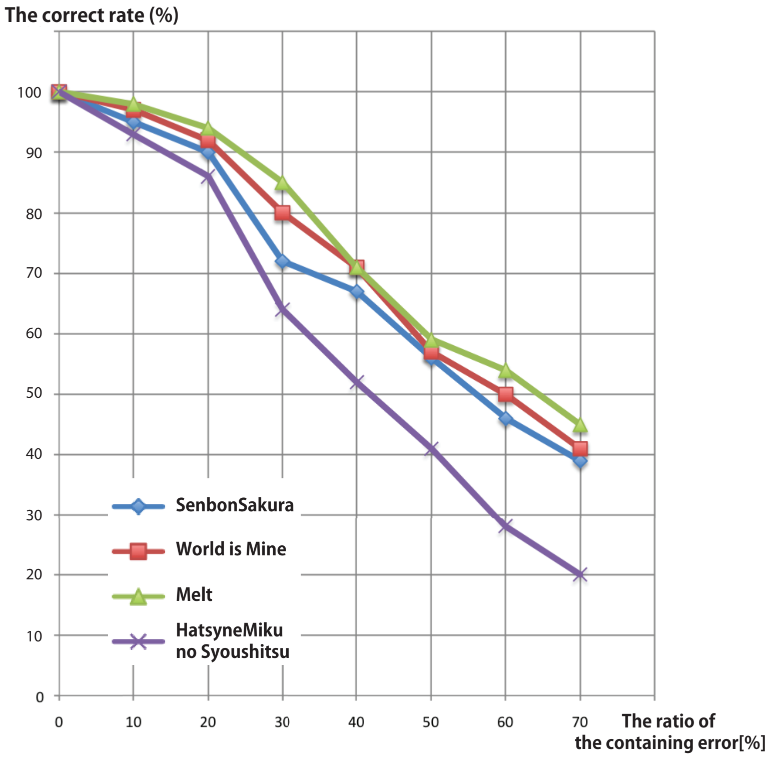


Figure 4.9: The robustness against mistakes during performance. The horizontal axis: The ratio[%] of numbers of the inserted error vowels to the total numbers of characters in the song. The vertical axis: The correct rate [0,1]. Higher value represents higher accuracy.

We held a workshop with ten amateur pianists, all of whom have experimented with piano or other musical keyboards for more than five years. We gave the players a musical score for our system and asked them to practice. We selected a standard jazz song "Autumn Leaves" (Music:Joseph Kosma, Lyrics:Junko Akiyama) with the tempo 120 [BPM] for practice. All participants had already known this song before the workshop. We printed the score at this study. The user checked the vocal singing sound only by hearing. The time to practice was three days, 30 minutes per day for all participants (Figure 4.10: center and right).

We monitored the ratio of the number of vowel input mistakes to the total number of characters of the song in the case of playing the musical scores strictly by repeating practices. The result is shown as Figure 4.11. This figure shows the relationships between the ratio of the different played notes to the original and practice times of each participant. We found that the ratio decreases rapidly with repeating practice. Then, we verified that our system doesn't require any further skills for common pianists.

After the practice, we requested the participants to play the song to an accompaniment for more than six choruses repeatedly, while modifying the melodies and jumping freely ad-lib. We provided musical score including several lyrics list for each participant during the experiment. As described in

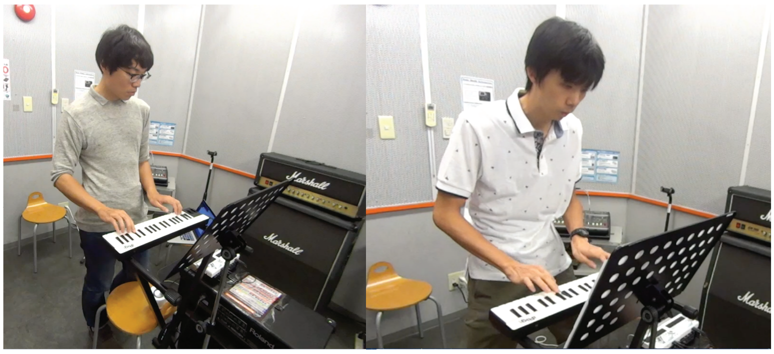


Figure 4.10: left: The system setup. center and right: Two scenes at the workshop.

§Introduction, most existing approaches, such as melody fitting, can't be used for this scenario. We included this actual scene in the supplemental video. The video shows each pianist playing the song while modifying the melody significantly ad-lib. The participants re-mixed the pieces of the predefined lyrics flexibly. This new type of musical expression is enabled by our system. Note that all participants we employed already had a skill for improvising music by piano before the experiment. So, special trainings for improvisation were not required, even including the control of lyrics.

After the workshop, We conducted individual interviews. At the interviews, we asked three questions. First, about the difficulty of the system: easy, neutral, difficult, impossible. If difficult or impossible, we were going to ask the reason. But, all participants answered it's easy. Second, about the playability. We asked whether the user has any dissatisfactions for expressing the musical phrases they want. Finally, we requested free comments.

Some participants mentioned that they feel discomfort at first because they already have substantial experience playing piano and can read musical scores, but the vowel keys we assigned on the musical keyboard don't correspond to the heard pitches of the synthesized sound. However, they also mentioned that the sense of discomfort decreased gradually as they adjusted to our system.

Nearly all participants said that the system allow input mistakes to some extent and it was a very nice feature. They added that practice for our system was substantially easier than they expected, because the lyrics are represented as a standard musical score. We consider these to be significant merits of our system.

Others said that they often worried about the latency of the “S" consonant when voicing. The characters that have the “S" consonant are sibilant. Sibilant characters are voiced after a comparatively long breath sound, and the timing of voicing the vowel is perceived as the rhythmic timing. Then, if the user presses a key to start a voicing sibilant on the exactly the timing in the rhythm, the sound will be perceived late. In the case of voicing by mouth, we usually overcome this problem through practice, intentionally beginning to voice a

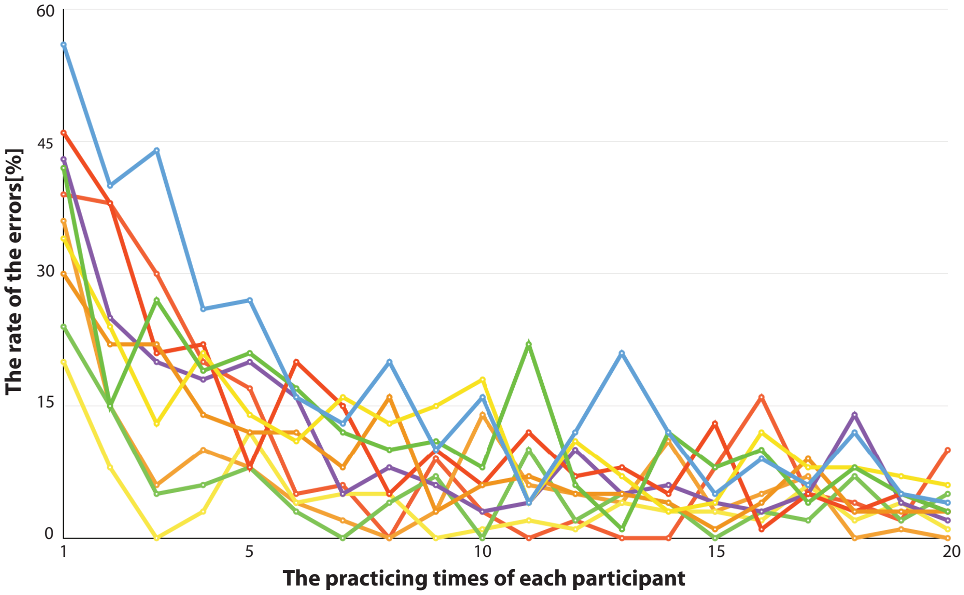


Figure 4.11: The amount of vowel input mistakes made by test participants plotted against the number of practices. The vertical axis shows the ratio of the errors to the total numbers of characters in the song and the horizontal axis shows the number of practice.

character with the “S" consonant slightly earlier. However, the participants had not mastered it for our system in our limited study. We think that this requires substantially more training.

4.6 Future Work

An important future work is more quantitative evaluation of the playability. Our system predicts the lyrics in the dataset from user’s input. However, how the result is responded to the user’s intend has been not sufficiently verified in our experiments. We could verify this by investigating the matching rate between a participant’s eyes focus on a musical or lyric score and the system’s output using eye tracking sensor. An another possible approach is to make the participant to follow requested lyrics including position jump on a musical score, and investigate the correct rate of the output. Such quantitative experiments remain for future work.

Several limitations are observed in our work, which remain to be addressed in future work. Our approach to controlling a singing voice synthesizer was dependent on the specific properties of the Japanese language, which is a $Mora$ language with only five vowels. Consequently, it is difficult to apply to other languages such as English. Thus, applying our method to various languages is a future direction. We are interested in how the structure of song lyrics and the user’s performance can be learnt with machine learning model at precomputation phase. For example, in our system the interpretation of the structure of a song depends on manual annotations by the user. We expect that it will be convenient to have the system automatically extract this structure by using natural language processing techniques. In addition, our method has many parameters, such as the state transition probabilities in the HMM, determined empirically. Automatic learning and adjustment of these parameters is an important task left for future work.

Another future direction is to control the continuous parameters of computational sound by low DoFs input device. In this chapter, we focused on controlling the lyrics and melodies of a song. In this context, lyrics are switching parameters (it switches from a character to an another character discontinuously). However, continuous parameters such as volume and timbre play quite important role in controlling computational sounds, which was not addressed in this chapter. By addressing this problem, we could control various computational sounds beyond a singing voice synthesizer. Thus, controlling continuous parameters by low DoFs input device is a challenging direction for future work.

4.7 Conclusion

In this paper, we proposed a practical user interface that enables the use of real-time singing voice synthesizer at an improvisational live performance by inputting the lyrics and melodies of songs simultaneously using a standard musical keyboard. The proposed system allows arbitrary movements within the lyrics including jumping, backtracking, and mistakes by estimating plausible lyrics from vowel sequences using a probabilistic model. Additionally, the lyrics for our system can be described using a standard musical score making it easier to learn. As a result, we achieved very flexible control of lyrics and melody using our system at real-time rate that enabled live improvisational performance of distinctive musical expressions.

Chapter 5

Fully Perceptual Based Calibration of 3D Audio Spatialization for A Specific User

5.1 Introduction

The human auditory system perceives the directions of incoming sounds using both ears. According to the direction from which a sound arrives to the head, an arrival time difference to the left and right ears can be determined. In addition, the sound is intricately diffracted by the shape of the person's head and ears. This diffraction effect depends on the frequency and incoming direction of the sound. Therefore, the spectrums of the sounds that arrive at each ear are modified. We can recognize the localization of the sound by these sound modifications. These two-channel transforms of the spectrums can be represented as finite impulse response filters and are called human-related transfer functions (HRTFs).

Three-dimensional (3D) spatialization of sounds in virtual environments (e.g., VR and games) requires HRTFs to reproduce incoming sounds from various directions using a two-channel headphone. However, HRTFs are highly specific to individuals because they depend considerably on the shape of the user's ears and head. We call the proper HRTFs for individual users as an individualized HRTFs. We know that inappropriate HRTFs can lead to improper localization of the sound source accompanied by an unexpected equalization of the timbre. Such improper localization especially includes front-back and up-down confusions [138, 96, 92]. Because of this, we must essentially measure the individualized HRTF for each user. The measurement procedure requires special equipment, including an anechoic chamber, as well as time-consuming and tedious efforts of the user. Thus, using individualized HRTFs for each end user has been impractical. This may explain why 3D sound rendering has not been as popular as visual rendering.

To address these problems, we propose a novel fully perceptual-based op-

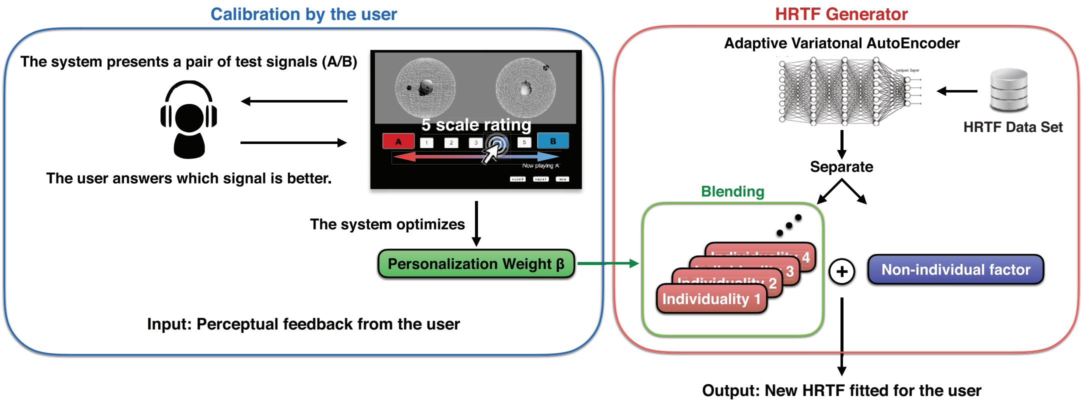

Calibration by the user
The system presents a pair of test signals (A/B)
5 scale rating
A
B
The user answers which signal is better.
The system optimizes
Personalization Weight β
Input: Perceptual feedback from the user
HRTF Generator
Adaptive Variatonal AutoEncoder
HRTF Data Set
Separate
Blending
4
3
2
Individuality 1
Non-individual factor
Output: New HRTF fitted for the user

Figure 5.1: **The concept of the system**. The user can calibrate their own Human Related Transfer Function (HRTF) for 3d audio spatialization. The system presents a pair of test signals, and the user feedbacks which one is perceptually better. Using this feedback task iteratively, our system optimizes a personalization weight for the user to obtain an individualized HRTF. The personalization weight blends individual factors of HRTF which are extracted from a public HRTF data set during training.

timization of HRTFs for individual users (Figure 5.1). Our system requires neither special equipment nor tedious measurement procedures. The user only needs to provide several feedback rating pairwise comparisons of test signals provided by the system based on his or her individual perceptions during calibration. This dramatically reduces the user's efforts at obtaining individual HRTFs. Our algorithm uses a novel adaptive variational AutoEncoder [71, 112] trained with a publicly available HRTFs data set. During training, it decomposes HRTFs in the data set into factors based on individual users and the rest. During calibration, our adaptive variational AutoEncoder generates individualized HRTFs for a new user by blending several individualities with personalization weight in nonlinear space. An advantage of this algorithm is that it does not require optimizations for all the spherical directions around the head because the personalization weight is shared within all the directions. Instead, it covers all directions by running optimizations for only a few candidate directions, which has been not addressed in previous studies.

We evaluate our algorithm by several quantitative validations and a user study. The cross validation shows that our algorithm has an ability to generate a fitted HRTF for a new user. In the experiment using synthetic data, our algorithm accurately predicts 3D acoustic field around an obstacle which demonstrates the ability to estimate a new HRTFs. Finally, we show that the individualized HRTFs obtained using our method outperforms best HRTFs in the data set in a user study with 20 users.

The contributions of this study are as follows.

The optimized HRTF can be used in an arbitrary platform.

Figure 5.2: After calibration, our system outputs the individualized HRTF in an arbitrary required format for each rendering platform.

1. We propose a fully perceptual-based optimization method to obtain individualized HRTFs for users, which dramatically reduces the user's effort (Section 4).

2. We present a novel adaptive variational AutoEncoder model that isolates factors based on individual users during training and synthesizes individualized HRTFs by blending them in a nonlinear space during calibration (Section 5).

3. We present a hybrid CMA-ES, assisted by a local Gaussian process regression, that accelerates sampling-based optimization of a black box system through user feedback by estimating gradient information (Section 6).

5.2 Related Work

In this section, we describe existing works for calibrating 3D audio spatialization and several techniques for representing human related transfer function in a reduced space.

Measurement: The straightforward approach to obtaining individual HRTFs involves the actual acoustic measurements in an anechoic chamber. Loud speaker arrays are spherically arranged around the subject's head and two small microphones are inserted into both ears. The subject sits with his or her head placed at the center of these spherical speaker arrays and is instructed to remain still during the long measurement periods. Specific test signals (such as sine sweep signals) is played one by one from the different loud speakers and the signals at the microphones are recorded. By comparing these recordings with those obtained from a microphone placed at the center

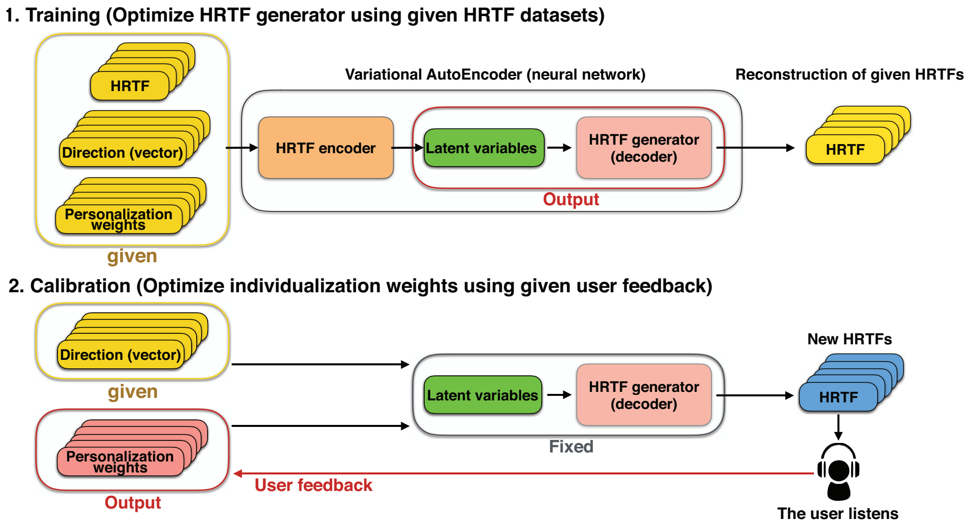


Figure 5.3: An algorithm overview. Our algorithm consists of two phases. At the first phase, we train a neural network using a public HRTF data set. At the calibration phase, the system shows test signals generated from the HRTF generator, and the user provides feedbacks according to his/her perceptual direction of the signal. Using this feedback information, the system optimizes personalization weight which is used for the input of HRTF generator to make a new HRTF for the user.

of the speaker arrays (excluding the subject), the individual HRTF can be computed. Many variants exist for conducting these measurements. However, because such measurements usually require expensive equipment as well as tedious procedures, using these measurements with each end user is impractical and thus prevents their widespread use. An alternative measurement approach is to use reciprocal method [157][89] that much reduces the measurement time. This approach swaps the loud speaker and the microphone positions. It inserts a micro-speaker into the subject's ear and places several microphones around the subject. To measure the HRTF, test signals are played from the inserted speakers and captured by the microphones. However, this has a limitation to capture only the HRTF at a narrow middle range frequency because it highly depends on the specification of such the small loud speaker.

Numerical Simulation: To avoid actual measurement in an anechoic chamber, many numerical simulation techniques for HRTF have been proposed. These techniques use the scanned 3D mesh of a human's head and ears, and solve an acoustic wave equation to simulate the sound propagation around the head. Two major approaches for solving this acoustic wave equation are the boundary element method (BEM) [62, 66, 39, 59] and the finite-difference time-domain (FDTD) method [94, 95, 141]. However, these techniques have three critical limitations. First, the simulation usually consumes tens of hours or a few days when using an auditory desktop computer. Second, the user must scan the 3D mesh data, which requires special equipment and additional effort. Third, the scanned mesh of the ears lack the details of the geometry that considerably affects the high frequency domain in an HRTF. To address the first problem, Alok et al. [91] improved the computational speed of the simulation using an adaptive rectangular decomposition technique (ARD). They achieved a computation time of less than 20 min for a broadband HRTF using auditory machine. However, it still requires the detailed mesh of head and ears which is difficult to obtain.

To avoid the tedious 3D mesh scanning procedure, DeepEarNet [64] estimates the 3D geometry of a user's ears from RGB photographs using a deep convolutional neural network and then numerically simulate the HRTF using BEM. This system extracts the feature parameters of the ears of the user from the photographs of the ears captured by the user him- or herself from two directions through manual annotations. However, sufficiently capturing the details of the ears in order to estimate the high frequency domain of the HRTF is difficult.

HRTF Optimization: To obtain fitted HRTF for a user, Zotkin et al. [156] attempted to select the best matching one from an existing data set. They measured pinna parameters of the user and selected an HRTF of a subject who has the closest pinna parameters. However, this has no guarantee that the picked one is the best HRTF for the user. Several studies have been conducted to optimize an HRTF for an individual user using machine learning techniques. Josef [47] asked users to adjust the principal component weights (PCW) of the HRTF manually using a slider on a user interface (UI). This was accomplished using listening tests. However, manipulating the unintuitive PCW parameters directly was found to be difficult. Yuancheng et al. [85] assumed the "blackbox" human auditory system, which takes a sound cue as input and returns the perceptual direction as a Gaussian process regression model and fits the HRTF for the "virtual" user model using AutoEncoder. Their results imply that nonlinear dimensionality reduction better reconstructs HRTFs than do linear space reduction methods such as principal component analysis (PCA). However, the AutoEncoder they employed, which was trained with the consolidated data of multiple subjects, may spoil the individualities of an HRTF. In addition, their experiments were merely a virtual simulation under rather limited conditions and had not been applied to actual problems.

A major approach to optimizing an HRTF for a new user using machine learning is to solve a regression problem that predicts a low-dimensional reduced HRTF from the anthropometric information of the user's head and ears. Principal Component Analysis (PCA) [52] and independent component analysis (ICA) [50], which reduce the dimensionality of HRTFs in linear space, have been widely used. Bilinski et al. [10] represented the sparse vector of a new subject's anthropometric features as a linear superposition of the anthropometric features of a training subset. They then obtained the individual HRTF by using those features as weights to interpolate the HRTF in the training dataset. Various nonlinear regression techniques such as neural networks [48] and support vector regression (SVR) [49, 136] have also been

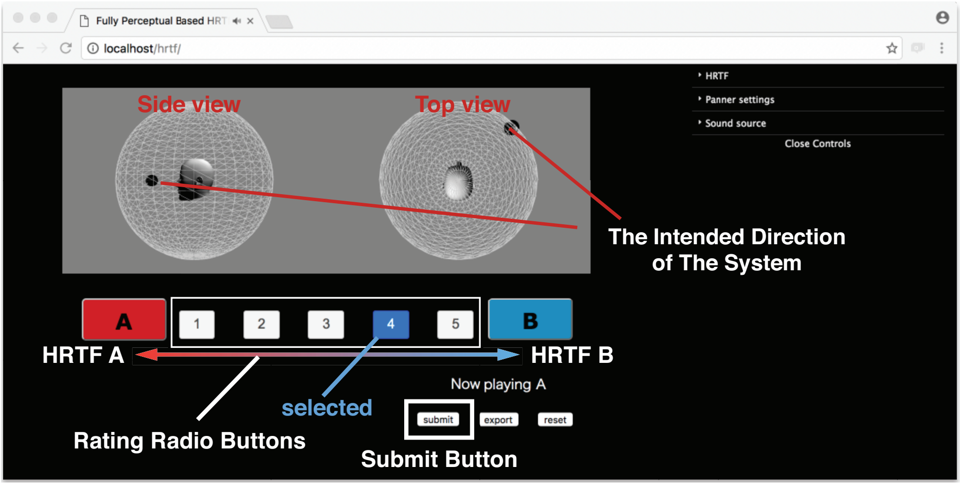


Figure 5.4: The user interface pane for gathering the user feedbacks. It runs on web browser. When the user push A (red) or B (blue) button, one of the test signal pair is played. The 3D graphics shows the intended direction of the system from the side and top views. The user rates the pair by 5 pt scale with radio buttons and submit it. Finally, the user exports his/her individualized HRTF data by pressing the export button.

proposed. However, these regression models cannot adequately express the complex relationships between anthropometric features and low-dimensional HRTFs.

Felipe et al. [37] proposed a state-of-the-art approach to optimize horizontal plane HRTFs using manifold learning through a nonlinear regression model. This approach uses anthropometric information that is actually measured for each user by professional authors. The dimensionality of the HRTF included in a data set is first reduced using IsoMap in a nonlinear space. Next, a regression problem is solved using a neural network. The neural network then processes the anthropometric data of the user and outputs a corresponding reduced (low-dimensional) HRTF. Finally, the system generates the individualized (high-dimensional) HRTF by means of a linear superposition of the neighboring vectors of the reduced HRTF in the IsoMap. However, in addition to the difficulty of measuring the anthropometrics precisely for each user, these anthropometric-based approaches mostly rely on low-dimensional heuristically defined anatomical measures, which are not necessarily sufficient to describe the complicated shape of the ears of humans.

5.3 Algorithm Overview

Our algorithm consists of two steps (Figure 5.3): 1) Training of an HRTF generator, which involves learning the individual and non-individual features from an HRTF dataset (§5.4). 2), Calibration of the HRTF generator, which involves individualizing an HRTF generator for each specific user (§5.5). In the first step, we train our HRTF generator using an HRTF data set (we used CIPIC data set [2]). The HRTF generator is a generative neural network model that is based on an extension of a conditional variational AutoEncoder [69, 121]. We extend it by adding 3D convolutional layers (§5.4.5) designed for HRTF input, as well as novel adaptive layers (§5.4.6) that separate the individuality and non-individuality factors of the users in a nonlinear space. This neural network takes a set of HRTFs, a continuous vector, and a one-hot vector as input. The continuous and one-hot vectors represent a sampled incoming direction of a sound and subject label (which subject's data are inputted), respectively. It then reconstructs HRTFs by extracting the latent variables as output. After the training of the generative model, our neural network can generate a new HRTF for a given direction around a head using the following three types of input: the latent variables, an intended direction, and a vector defined as personalization weight. Personalization weight represents the amount of contributions from individuals in the dataset in blending.

In the second step (calibration), we optimize the personalization weight to generate an individualized HRTF for the target user. This step involves interaction with the user, as described in §5.3. The system optimizes the personalization weight to minimize the difference between the intended spherical direction of the system and the direction perceived by the user. After this calibration, our system can output an arbitrary HRTF format. Most real-time rendering platforms (e.g., game engines) store HRTF data at discrete directions internally and interpolate them at runtime. The system outputs HRTFs at the required directions for each platform and these can be used by an arbitrary rendering scheme depending on each platform (Figure 5.2).

5.4 User Interface

The user obtains individualized HRTFs by running a one-time calibration that roughly consumes 15∼25 min. The calibration application runs on a web browser (Figure 5.4). The system first presents a pair of test signals and its intended direction. The user then plays the test sound by pressing an A/B selection button. Each of these two test signals is generated from different HRTFs (personalization weights), respectively, and has the same intended direction. We randomly select an audio source from 10 predefined test sounds (e.g., speech, helicopter, short music phrase) and then filter the audio using the generated HRTFs. The intended direction continuously moves spherically around a head and is shown as a moving sphere from side and top views. The user listens to the test signals and provides feedback by selecting one of the 5-scale options that represents the sound that is perceptually closer to the intended direction shown on the screen with "1" meaning that one of two test signals is definitely better, and "5" meaning that the other test signal is definitely better. Thus, "3" means neutral. By iterating this simple pairwise comparisons (approximately 150∼200 times), the system automatically indit

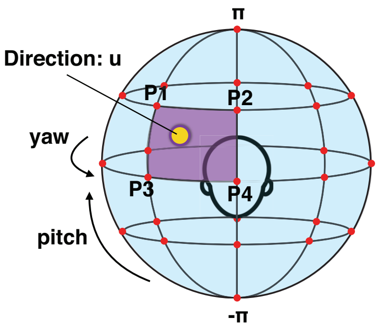


Figure 5.5: The representation for an incoming direction of a sound.

vidualizes the HRTFs by optimizing the personalization weight for the target user. The user can stop the calibration at an arbitrary timing (usually when the user satisfied or can not distinguish two test signals). This approach has two advantages. First, obtaining individual HRTFs for users is much easier with this than previous approaches. Second, the effects caused by the acoustic properties of the user's headphones can be considered by using the same headphones for calibration and runtime, something that was not addressed in previous studies.

5.5 Training with Public HRTF Data Set

5.5.1 Data Set

We used the publicly available CIPIC data set [2] which contains HRTFs of both ears actually measured in an anechoic chamber for 45 subjects at 25 azimuths and 50 elevations. In total, it includes 1250 sample directions of HRTF per subject and ear. Each set of data for a direction, subject, and ear is recorded as an impulse response of 200 wave samples with a 44.1kHz sampling rate audio file. Impulse signals are played by a loud speaker array spherically arranged around the head. They are recorded using two small microphones inserted into the ears of each subject. Instead of using the CIPIC angle representation (azimuth and elevation), we redefine the spherical coordinate as yaw and pitch (Figure 5.5). The yaw θ and pitch angle ψ are measured in a head-centered interaural-polar coordinate system. The yaw is the angle that varies from the back left $-\pi$ to the back right π. The pitch angle varies from the bottom $-\pi/2$ to the top $\pi/2$. We arrange all the sampling points on a unit sphere as 3D vertices and construct a sphere mesh with Delaunay triangulation. This triangulated unit sphere is used for obtaining HRTF data at an arbitrary direction by means of bilinear interpolation. Note that CIPIC dataset has a big hole at the bottom. Because of this, the bilinear interpolation could introduce some artifacts. To alleviate this, one can use arbitrary HRTF interpolation

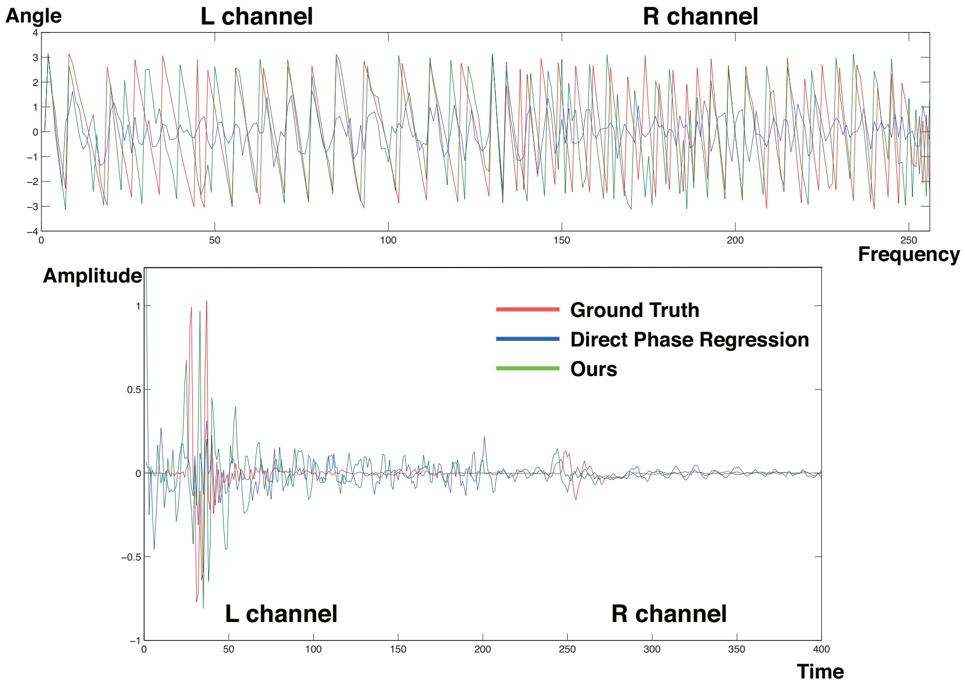


Figure 5.6: A comparison of phase (top) and time signal (bottom) reconstruction. Direct phase reconstruction approach (blue) outputs large error that causes unnatural noise in time signal.

methods [84, 28] alternatively.

5.5.2 Input Format

During training, our neural network take a sampled direction y (vector), a subject label (one-hot vector) s, and a set of HRTFs x from around the specified direction of the subject as input. It then outputs the reconstructed x' using an adaptive variational AutoEncoder. We train this model to have x' be similar to x. We represent a direction as $y \in \mathbb{R}^{26}$, whose elements are the weights to an overcomplete basis of 26 unit vectors evenly distributed in all directions $\in \mathbb{R}^3$ (red points in Figure 5.5). For a given direction vector $u \in \mathbb{R}^3$, the system identifies surrounding four unit vectors (P_1, P_2, P_3, P_4), and gives weights (w_1, w_2, w_3, w_4) to them so that $w_1 = s \cdot t$, $w_2 = (1-s) \cdot t$, $w_3 = s \cdot (1-t)$, $w_4 = (1-s) \cdot (1-t)$ where $Yaw_u = s \cdot Yaw_{P_1} + (1-s) \cdot Yaw_{P_2}$, $Pitch_u = t \cdot Pitch_{P_1} + (1-t) \cdot Pitch_{P_3}$. The weights of the other 22 unit vectors are set to zero. We do not use a 3D vector to represent the direction because a neural network tends to be insensitive to the fluctuation of continuous values on a node while being more sensitive to the binary-like activations on each node [24] The subject label $s \in \mathbb{R}^S$ becomes a one-hot vector that represents the inputted subject data. In our experiment, S became 45 dimensions because the CIPIC data set includes HRTFs of 45 subjects. If x is the p-th subject's, the corresponding element of s became 1; otherwise, 0.

Our neural network treats HRTF in a spectral domain. The original HRTFs

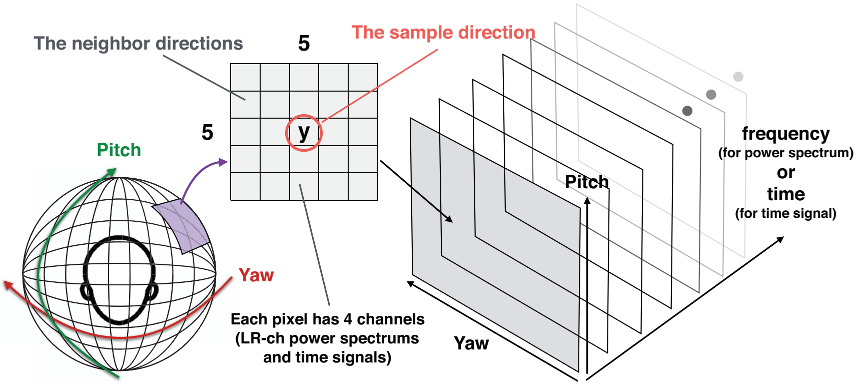

The neighbor directions
5
The sample direction
5
y
Pitch
Yaw
Each pixel has 4 channels (LR-ch power spectrums and time signals)
Pitch
Yaw
frequency (for power spectrum) or time (for time signal)

Figure 5.7: The input data structure of our neural network (We call HRTF patch). This HRTF patch has voxel like data structure, which encodes spatial correlations of HRTFs. Each voxel has four properties (LR channels of power spectrums and time signals) like color channels of image.

in a time domain can be recovered from both the power spectrum and phase information outputted from the system. For the power spectrum, we compute 256 rectangular-windowed FFTs of each HRTF impulse response and extract only the minimally required power spectrum (128 dimension vectors of LR channels). However, for phase information, we do not use spectral data (angles) directly because the reconstruction error of phase angles are quite sensitive. Actually, previous methods have not solved the regression problem of phase. Alternatively, we solve a rough regression problem of time domain signals, and estimate phase information from them. We use the first 128 samples of the time signal of an HRTF as input to our neural network. In summary, we reconstruct time signals through a neural network, and indirectly estimating only the phase angle (discarding the power spectrum information) from them. Using both the estimated phase and power spectrum, we reconstruct the final HRTFs. This indirect approach to estimating the phase reserves the rough shape of the impulse response, which is difficult to accomplish through direct phase estimation. Figure 5.6 shows the comparison of phase reconstruction between our method and direct phase estimation. The direct phase regression approach does not reconstruct the phase at all, but rather destroys the shape of the time signal. By contrast, our method successfully simulates the original phase information, thus preserving the time signal shape.

To construct the HRTF input data structure of x, we include not only the impulse response of the HRTF at the exact sampled direction y, but also its several surrounding neighboring impulse responses (Figure 5.7). In total, we sample 25 directions with 5×5 rectangular grid shapes, in which the center becomes the HRTF at the direction y. The stride of the grid is $\pm 0.08\pi$ rotations for both yaw and pitch on a unit sphere direction from the head. In addition, we obtain the power spectrums and LR time domain impulse responses at each direction using bilinear interpolation in frequency domain on the unit

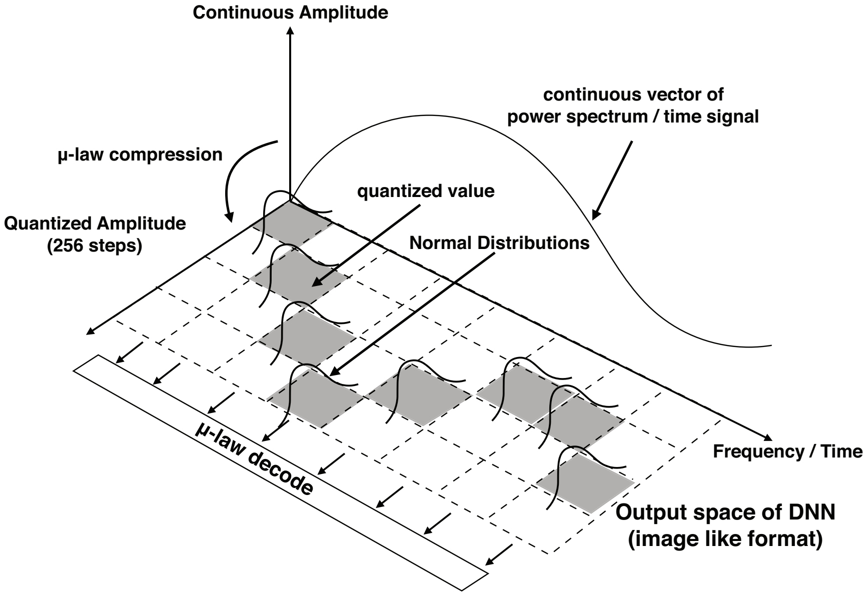


Figure 5.8: The output data structure .

sphere [139, 76]. To reconstruct an interpolated time signals, we use both interpolated power spectrums and phases. We call this 5×5 grid that stores HRTF information as the *Patch*. This patch representation is expected to encode the correlations with surrounding directions. Finally, this input data structure becomes a 3D voxel patch with 5 × 5 × 128 dimensions (128 power spectrums or 128 sample time signals) and each voxel has four color channels (power spectrums and time signals of LR channels) as shown in Figure 5.7. This becomes the input x of our neural network.

5.5.3 Output Format

Our neural networkreconstructs the HRTF x' to minimize the difference between the input and output HRTF with an AutoEncoder manner. However, solving a regression problem of a signal that shows a large fluctuation (e.g., time domain audio signal and power spectrum) using a generative neural network is difficult. This is because a neural network smoothes the output throughout the training data. Therefore, the trained result tends to be an "averaged signal," which causes a fatal error. To address this, we use a quantized format similar to WaveNet [133]. WaveNet predicts time domain audio signals using an image-like quantized format (width: time, height: amplitude), which successfully solves a regression problem of large fluctuated signals. Similarly, we quantize the power spectrums and time signals of HRTFs into 256 steps using μ-law compression. As a result, the output format becomes an image-like representation (Figure 5.8). Unlike in WaveNet, we do not use one-hot vectors for the final layer nor the SoftMax function. The SoftMax function generalizes all the output of the neural network into [0, 1] probabilities. This is equivalent to solving an unconstrained optimization problem which requires extensive

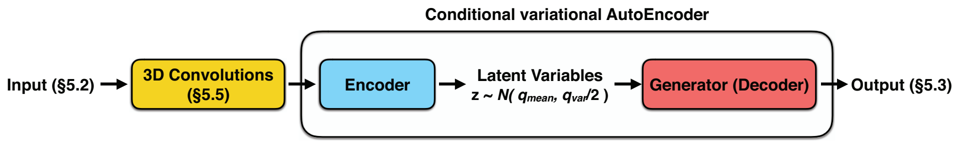


Figure 5.9: Our neural network architecture. Our neural network is an extension of a conditional variational AutoEncoder, which reconstructs HRTF from the inputted HRTF through the latent variables.

training data. However, the size of our setting's training data is considerably less than that of WaveNet, which can lead to optimization failure. Alternatively, we construct an array of normal distributions on each quantized vector, in which mean values are equal to each quantized value. In addition, we set all the variances to 5 (Figure 5.8) and minimize the mean squared error of these multiple normal distributions. This addresses the aforementioned problem because it is equivalent to constraining the value range of the solution. To generate a final HRTF, we first compute each quantized value by maximum likelihood estimation from the output and then obtain the result by decoding the quantized values using inverse μ-law compression.

5.5.4 DNN Architecture

Figure 5.9 shows our neural network architecture which has three blocks (please see Appendix E for the detail). Our neural network is an extension of a conditional variational AutoEncoder (Appendix D). Three extensions are used: 1) We introduce 3D convolutions for the input, which are specifically designed for our HRTF patch. 2) We propose an adaptive layer that decomposes the individual and non-individual factors during the training. 3) We introduce residual networks to prevent gradient vanishment. This neural network uses an HRTF patch x, a sample spherical direction y, and the subject label s as input, and reconstructs x' by extracting the latent variables. We divide the HRTF patch x by each channel and input them separately as the power spectrum channels of LR x_{fl} and x_{fr} and the time signals of LR x_{pl} and x_{pr}. Similar to the conventional variational AutoEncoder, the architecture has an encoder and a decoder. The encoder extracts latent variables z_{mean} and z_{var} from the input, and the decoder outputs reconstructed HRTFs in the format described in the previous section from the sampled latent variables z.

5.5.5 3D Convolutional Layer for HRTF Patch

As shown in Figure 5.9, we embed 3D convolutional layers for the input of the variational AutoEncoder at the encoder. We expect these convolutions to encode the correlations between the HRTF at the sample direction and its surrounding neighboring directions within an HRTF patch. A typical 3D

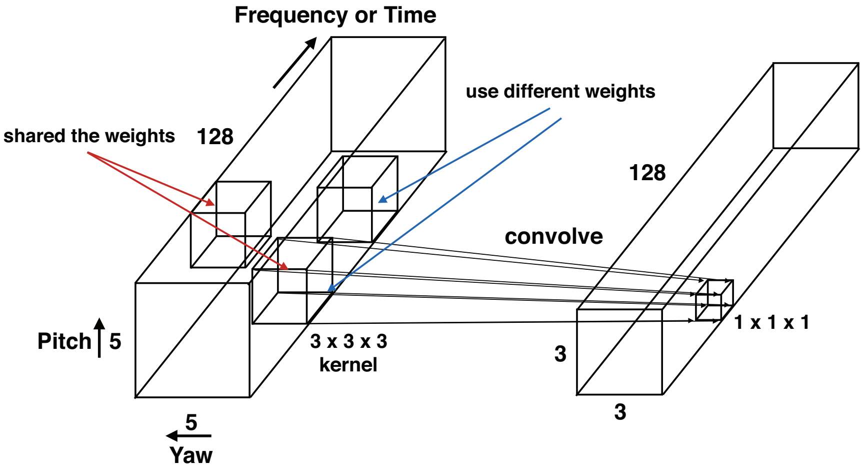


Figure 5.10: 3D convolutional layer for HRTF patch.

convolutional layer [127] in a neural network shares the filter coefficients of the kernel over the weight tensor. Instead of using this type of layer, we employ a convolutional layer that shares the kernel coefficients only in spatial domains (the yaw and pitch axis) but uses different filters along the frequency axis (Figure 5.10). This is because the spectral correlation of an HRTF with its surroundings generally has a different structure between the lower and higher frequencies as a result of the frequency dependent diffraction by the subject's head and ears.

We use two convolutional layers for each channel (for four total channels). We set the kernel size of each convolution as 3×3×3 (yaw × pitch × frequency axis), and add zero padding to the frequency axis only. For all directions, we set the stride as 1. Thus, the first convolutional layer transforms each channel of a patch from 5×5×128 to 3×3×128, and the second layer further transforms them to 1×1×128. Note that we did not add bias parameters to this convolutional layer in our experiment.

5.5.6 Adaptive Layer

Our primary technical contribution is decomposing the individual and non-individual factors from an HRTF dataset during training. This technique uses a novel type of neural network layer called an adaptive layer, which isolates the latent individualities into a tensor from the weight matrix in an unsupervised manner. In addition to the input vector x, this adaptive layer uses a one-hot vector s during training and a continuous vector β during runtime as input.

Our adaptive layer is based on tensor factorization that employs stochastic gradient descent optimization [72]. A common layer in a neural network can be written as a combination of linear and nonlinear functions,

$$y = F(x) = f(Linear(x)) = f(\boldsymbol{W} \cdot x + b), \tag{5.1}$$

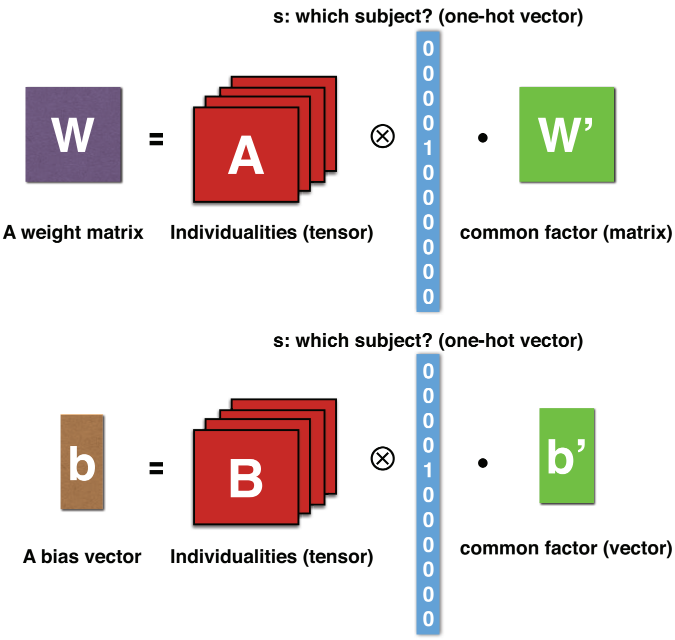


Figure 5.11: An adaptive layer that decomposes the function approximation into individual feature and non-individual feature. A one-hot vector s works like switching function depending on the inputted subject's data. $\otimes$ denotes tensor product.

where $x \in \mathbb{R}^M$ and $y \in \mathbb{R}^N$ are the input and output of this layer, respectively. $Linear()$ denotes a linear layer function of the neural network. $\boldsymbol{W} \in \mathbb{R}^{N \times M}$ is a matrix, $b \in \mathbb{R}^M$ is a bias vector, and $f()$ is an arbitrary nonlinear function (e.g., Sigmoid). We decompose this $\boldsymbol{W}$and b as follows by introducing a new parameter s (Figure 5.11):

$$y = f(Adapt(x,s)) = f(\boldsymbol{A} \otimes_3 s \cdot \boldsymbol{W}' \cdot x + \boldsymbol{B} \otimes_3 s \cdot b'), \tag{5.2}$$

where $s = [s_1, ..., s_S]^T$, $s_k \in \{0,1\}$ is a one-hot vector that represents the subject to which the inputted data belongs. $\boldsymbol{A} \in \mathbb{R}^{N \times M \times S}$, and $\boldsymbol{B} \in \mathbb{R}^{M \times M \times S}$ are tensors. $\boldsymbol{W}' \in \mathbb{R}^{N \times M}$ is a matrix, and $b' \in \mathbb{R}^M$ is a vector. $\otimes_d$ is the dot product between the d-mode expansion of a tensor and vector. We replace the linear layers in the variational AutoEncoder with this adaptive layer, and iteratively input the HRTF data of a randomly selected subject and direction during optimization. The adaptive layer gradually inserts the individualities of the inputted HRTF patch into the tensor $\boldsymbol{A}$ and $\boldsymbol{B}$ as if the s becomes the switcher with respect to the selected subject. In addition, non-individualities that are shared with all subjects are included in the matrix $\boldsymbol{W}'$ and the vector b' during stochastic optimization.

This adaptive layer allows us to interpolate, emphasize, diminish, and blend each individuality in the trained dataset by adjusting the personalization weight vector $\beta \in \mathbb{R}^S$ at runtime rather than using s as an additional input. To achieve a similar but not necessarily identical objective, several ap-

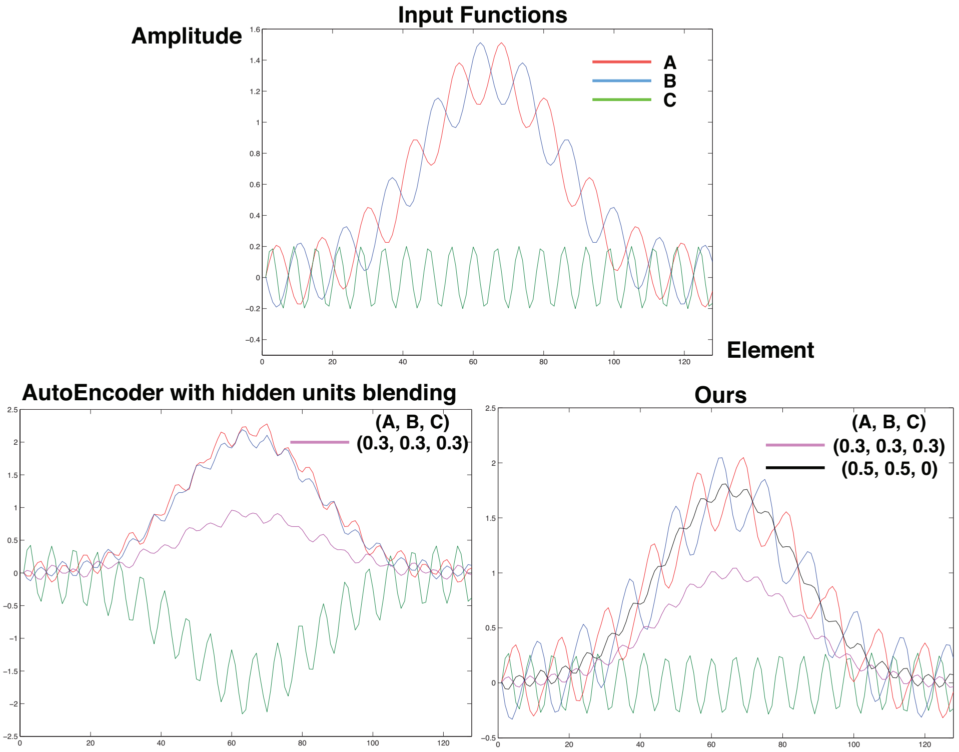


Figure 5.12: A comparison between hidden units interpolation approach (bottom-left) and our adaptive layers (bottom-right). We trained two networks with three nonlinear functions A, B and C (Top). Red, blue, green lines at bottom two graphs represent the reconstructed functions respectively. Purple lines denote a blending of three functions equally, and black line denotes a blending of A and B. Hidden units interpolation approach diminishes the details of each function while our method preserves them.

proaches exist that morph data into different categorized data continuously by interpolating several sampled hidden units extracted with AutoEncoder (e.g., using procedural modeling of a 3D mesh [153] and controlling and stylizing the human character motion [45]). However, these approaches are limited in terms of their ability to distinguish many nonlinear functions, which is crucial to solving our target problem. Figure 5.12 shows a comparison of three simple functions morphing between hidden units interpolation approach and our approach after performing the same number of iterations (although this number is unfavorable with our approach). We use a dual-stacked AutoEncoder as shown in Figure 5.13) for our experiment. (Note that for hidden units interpolation, we replace each adaptive layer with a common fully connected layer.) The hidden units interpolation approach diminishes each feature of the functions. This is crucial to solving our target problem because sharp peaks and dips in the spectral domain are commonly important specifications for an HRTF. However, our adaptive layer successfully enables us to reconstruct the details of each feature and blend them.

In addition, we introduce residual neural networks [42] into the adaptive

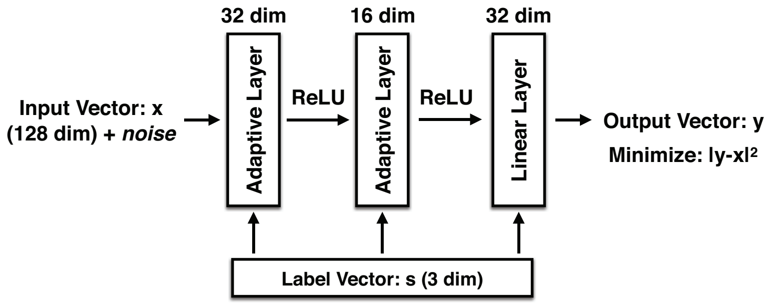


Figure 5.13: AutoEncoder network for validation of our adaptive layer.

layers in the encoder (Figure 6.2) and decoder (Figure 6.3) in order to reduce the training error of deeper neural networks. A residual network has a shortcut connection as given by the following equation:

$$y = Adapt(x, s) + \boldsymbol{U} \cdot x, \tag{5.3}$$

where $\boldsymbol{U}$ denotes a matrix to project the input x into the output dimension space of y.

5.6 Optimizing for an Individual User

After training, we calibrate the HRTF generator (the decoder of the neural network) to obtain an individualized HRTF for a user. To this end, we assume the individuality of the optimized HRTF for an individual user can be expressed as a blending of the trained individualities of HRTFs in the dataset in a nonlinear space. Thus, we now replace the binary subject label s in Eq.(5.2) with a continuous personalization weight $\beta = [\beta_1, ...\beta_S]^T$ which is called personalization weight. When this β is used, $Adapt(x, s)$ becomes $Adapt(x, \beta)$. Each β_i takes [0,1] continuous value while s is the binary one-hot vector, and is constrained as $\sum_i^S \beta_i = 1$. Finally, the adaptive layer in this phase is reformulated to

$$y = Adapt(x, \beta) = f(\boldsymbol{A} \otimes_3 \beta \cdot \boldsymbol{W'} \cdot x + \boldsymbol{B} \otimes_3 \beta \cdot b'). \tag{5.4}$$

This representation means the optimized individualization transformation matrices to the user can be expressed as $\boldsymbol{A} \otimes_3 \beta$ and $\boldsymbol{B} \otimes_3 \beta$, which are blendings of the individualities of the subjects included in the trained data set. Similarly, the latent variables z, which are necessary for generating a new HRTF, are also transformed using β as:

$$\begin{aligned} \bar{z}_{mean} &= \boldsymbol{Z}_{mean}(y) \cdot \beta, && (5.5) \\ \bar{z}_{var} &= \boldsymbol{Z}_{mean}(y) \cdot \beta, && (5.6) \\ \bar{z} &\sim \mathbb{N}(\bar{z}_{mean}, \frac{1}{2}\bar{z}_{var}), && (5.7) \end{aligned}$$

where $\boldsymbol{Z}_{mean} \in \mathbb{R}^{L\times S}$, $\boldsymbol{Z}_{var} \in \mathbb{R}^{L\times S}$ are matrices in which each column is the pre-computed latent vector $(z^1_{mean}(y), ...z^S_{mean}(y))$ and $(z^1_{var}(y), ...z^S_{var}(y))$ that correspond to the subject. Furthermore, $z^s_{mean}(y)$ and $z^s_{var}(y)$ are switched by the direction y. Note that $z^s_{mean}(y)$ and $z^s_{var}(y)$ are pre-computed using the trained model for each direction before this step is performed. L denotes the dimensions of the latent variables, and we use 32 for our experiments. We use the blended $\bar{z}$ for the latent variables in the individual feature vector of the user.

The system optimizes the personalization weight vector β for an individual user by fixing the other parameters $\boldsymbol{A}$ and $\boldsymbol{B}$, as well as the matrices $\boldsymbol{W}'$ and bias vectors b'. This approach has the advantage of dramatically reducing the DoFs of the design variables for optimization purposes because it can eliminate the need for multiple optimization runs when considering all spherical directions. This means optimizing only a blending vector β covers the individualities of the user through all directions.

5.6.1 Optimizing Personalization Weight through User Feedback

The optimization procedure for the personalization weight β is interactive with the user as described in §5.3. The user gives relative scores for two individualization weights β_i and β_j. With this input, our optimization problem is reformulated into a minimization problem arg $\min_\beta Q(\beta)$, where the absolute cost values $Q(\beta)$ are computed from the relative scores as described in a later section. By running this procedure iteratively, the system optimizes the β.

To optimize this black box system, Bayesian optimization [13] is widely used. However, it requires absolute evaluation values at each step, which are not immediately available in our setting. Alternatively, we use a hybrid optimization scheme [17] as an evolutional strategy (we use CMA-ES [40]) and a gradient descent approach (we use the BGFS-quasi-Newton method). The optimization procedure is shown in Algorithm 1. We introduce local Gaussian process regression (GPR) [88] to accelerate the optimization. This method estimates the local landscape of the cost function from discrete sampling to obtain the gradients. Figure 5.14 shows a comparison between with and without using gradient information estimated by GPR. We minimize the EggHolder function for this evaluation using four conditions (N=8, 32). CMA-ES requires N times of function evaluation (pair-wise comparisons) at each iteration. In our target problem, the number of samplings should be small because the sampling size is proportional to the user's effort. However, when the number of samplings N seeded by CMA-ES is few, the optimization without GPR tends to be trapped by a bad local minima. As shown in Figure 5.14, our technique addresses this problem and converges to a better solution with considerably fewer iterations than in previous methods.

At a single iteration during optimization, we first sample N sets of β sam-

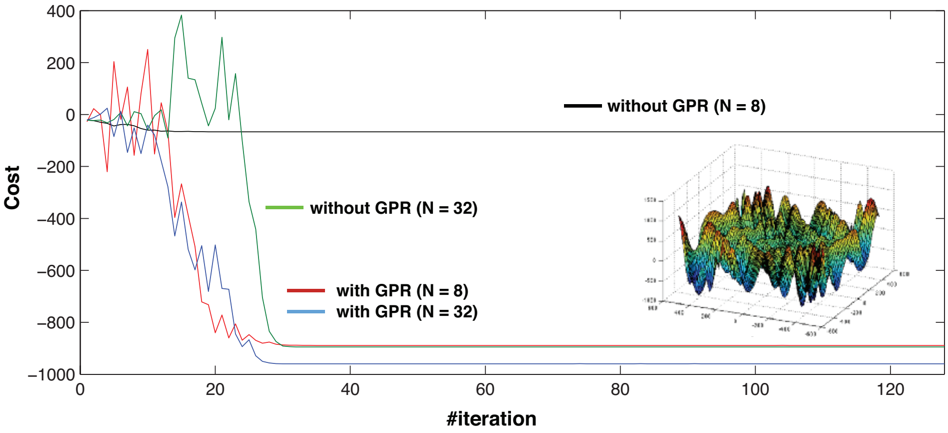


Figure 5.14: Comparison of convergence curves between CMA-ES with/without GPR. We use EggHolder function for this evaluation. When the number of samplings seeded by CMA-ES is fewer, the optimization without GPR fails to bad local minima while our technique converges to better solution with much fewer iterations.

plings $\beta_1, ..., \beta_N$ using common CMA-ES procedure (we used $N = 8$). Let $\mathbb{P}$ be a set of pairs of indices $(1, 2), ..., (N-1, N)$. For each $(i, j) \in \mathbb{P}$, the system generates two test signals $S(\beta_i)$, $S(\beta_j)$, presents the pair side by side to the user, and requests the user to rate each to generate a relative score. After collecting the user feedback for all pairs, the system stores $N/2$ pairs of different βs and their relative scores. After computing the absolute value of the cost q for each β samplings using these relative scores, we estimate the local landscape of the continuous cost function $Q(\beta)$ from the discrete q. We represent $\mathbb{Q} = (q_1, ..., q_N)$ as a set of indices of sampling points. Using this estimated cost function, the system computes the gradients, and updates the covariance matrix in CMA-ES with quasi-Newton. We detail this procedure in the next subsection.

Estimating the Local Landscape of the Cost Function: The system requests that the user provides feedback regarding the two test signals that correspond to each β pair. The system then stores relative scores for the β pairs $\mathbb{P}$. Given these relative scores, we compute the absolute value of the sampling cost q for each sampling β. Our formulation is derived from Koyama et al. [73], which estimates the consistent goodness field of high dimensional parameters through unreliable crowd sourced rating tasks. Their approach solves a minimization problem with two constraints:

$$\arg\min_{q} \left(E_{relative}(q) + \omega E_{continuous}(q) \right), \tag{5.8}$$

where $\omega > 0$ balances the two constraints (we set 5.0). $E_{relative}(q)$ is the relative

ALGORITHM 1: Hybrid CMA-ES assisted by Local GPR

```
Until convergence
    Generate $\beta$ samplings using Eq.(5.14)
    and make $P$ test pairs $\mathbb{P}$.
    Gathering the user feedbacks for each pair.
    Computes absolute cost $q$ for each $\beta$ samplings.
    Estimating the cost function of each sampling using GPR
    Sort the samplings by the order of the cost
    to form the new parent population in CMA-ES
    The weighted mean $y_w^{(g)}$ is computed
    from the new parent population .
    Quasi-Newton updates of $y_w^{(g)}$
    using the gradients estimated by GPR.
    Update the covariance matrix $C^{(g)}$ and global step size
    in CMA-ES using Chen et al. [17],
    respectively.
end
```

score-based constraint and is represented as

$$E_{relative}(q) = \sum_{(i,j)\in\mathbb{P}} ||q_i - q_j + d_{i,j}||^2, \tag{5.9}$$

where $d_{i,j}$ denotes the offset determined by the rating between i-th and j-th samples.

$$d_{i,j} = \begin{cases} 1 & (relative\ score\ =\ 1) \\ 0.5 & (relative\ score\ =\ 2) \\ 0 & (relative\ score\ =\ 3) \\ -0.5 & (relative\ score\ =\ 4) \\ -1 & (relative\ score\ =\ 5). \end{cases} \tag{5.10}$$

Note that the sign of $d_{i,j}$ is opposite to Koyama et al., because our optimization is a minimization problem. In addition, we enforce the continuity of the cost function by $E_{continuous}(q)$:

$$E_{continuous}(q) = \sum_{i\in\mathbb{Q}} ||q_i - \sum_{i\neq j}(1 - \frac{|\beta_i - \beta_j|}{\sum_{i\neq k}|\beta_i - \beta_k|})q_j||^2. \tag{5.11}$$

In this equation, we constrain the absolute costs of two sampling β to become closer when the distance of the two β diminishes. This minimization problem Eq.(5.8) can be solved as a linear least square problem.

Finally, we estimate the local landscape of the cost function $Q(\beta)$ of the β samplings using multidimensional GPR. We include discrete q samplings obtained by Eq.(5.8) into GPR. This approximate function can be used to estimate the gradients, which are required by the quasi-Newton method as described in the following paragraph. Note that although GPR is expensive with high dimensions, it is not a serious problem in our case because the dimension of a design parameter would not increase to such a high dimension.

Optimization: We employ a hybrid optimization scheme [17] of an evolutional strategy to minimize $Q(\beta)$ with respect to β. Note that we used CMA-ES [40] and a gradient descent approach (quasi-Newton method.) This hybrid approach first updates the design parameter using gradient information to search for the local optima and to escape from bad local optima. The evolutional strategy aspect generates the offspring (sampling) using two characteristic variation operators, and additive Gaussian mutation alternately:

$$q^g = QuasiNewtonUpdate(z^g), \tag{5.12}$$

$$z^{(g+1)} = q^{(g)} + \rho^{(g)} \boldsymbol{B}^{(g)} \boldsymbol{D}^{(g)} y^{(g)}, \tag{5.13}$$

$$\boldsymbol{B}^{(g)} \boldsymbol{D}^{(g)} y^{(g)} \ \sim \ \boldsymbol{N}(0, \boldsymbol{C}^{(g)}), \tag{5.14}$$

where $z^{(g)}$, g and $\rho^{(g)}$ are the design parameters, iteration step and a global step size respectively. $y^{(g)} \ \sim \ N(0, \boldsymbol{I})$ are independent realizations of a normally distributed random vector with zero mean and covariance matrix equal to the identity matrix $\boldsymbol{I}$, and $C^{(g)}$ denotes the covariance matrix which is computed using $q^{(g)}$ and $z^{(g)}$ (please see [17] for details).

For our initial guess, we first randomly generate the offsprings $y^{(g)}$ within a range [0, 10] by means of Gaussian distribution, and then enforce a constraint $\sum y^{(g)} = 1$ by dividing all the offsprings by $\sum y^{(g)}$. This constraint can be considered as a portion of the user feedback function, and we can optimize CMA-ES with the common range [0,10]. In addition, we assume β has a sparsity because most of optimized HRTF can be represented by a combination of a few HRTFs included in dataset. Thus, the optimizer randomly drop the elements under the average to zero in β offsprings of CMA-ES (we dropped them to 30% probability). We found this constraint reduces training error.

5.7 Validation

5.7.1 Implementation

We implemented our neural network algorithm using C++ (with AVX2 operations) and CUDA from scratch. Our CPU had an Intel Core i7 6900K 3.2 GHz, RAM 128 GB. Our GPU was an NVIDIA Geforce GTX1080x3. The training consumed approximately 8 hours. The GUI application for calibration ran on a web browser. This web application communicates with a GPU server (same machine used for the training) in the background. The calibration algorithm was written in Javascript. The front end GUI application sends βs to the GPU server, and the GPU server generates HRTFs. For our user study, we used the AKG K240 headphones.

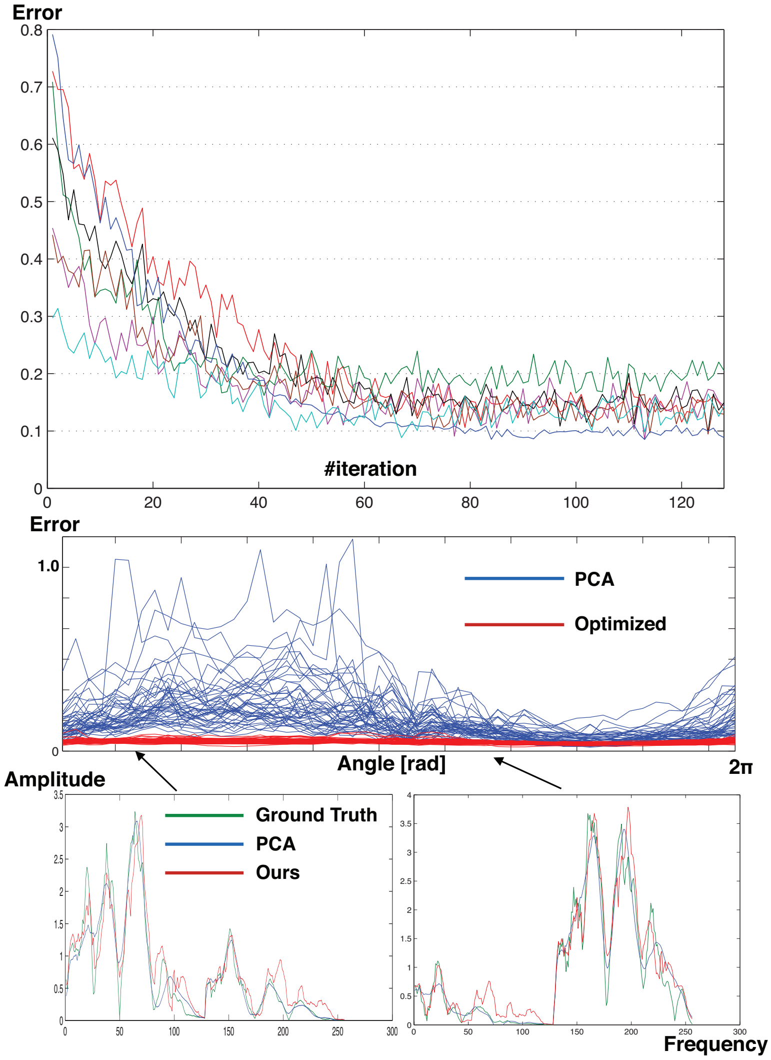


Figure 5.15: The result of cross validation. Top: convergence curve of each optimization. Middle: A comparison of estimated errors between PCA and our algorithm on the horizontal plane. Bottom: A comparison between the power spectrum of a target (blue) and optimized result (red).

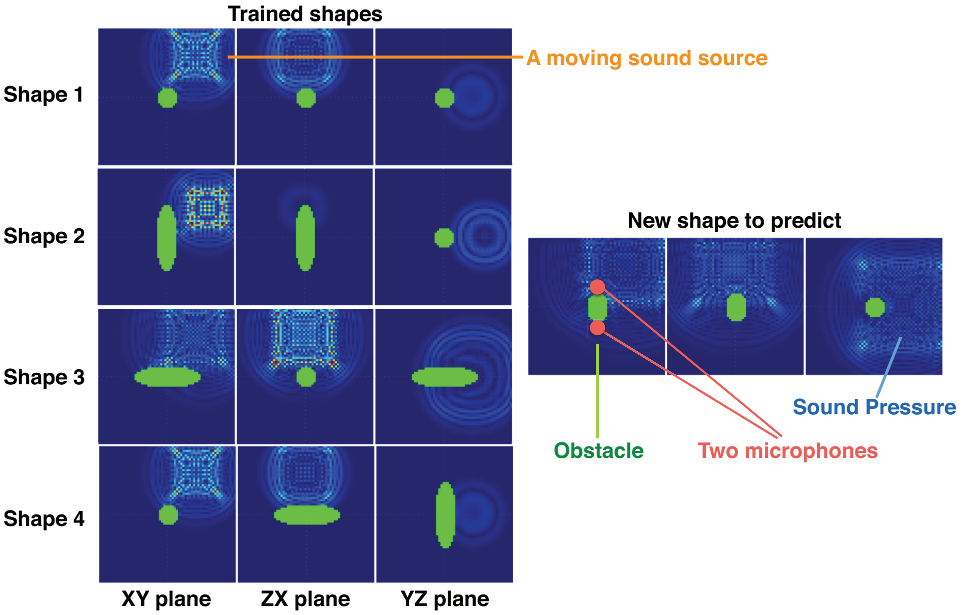


Figure 5.16: Synthetic data generation. We conducted 3d acoustic simulation. Green region represents an obstacle.

5.7.2 Quantitative Validation

Cross Validation: Confirming the assumption that the individuality of a user can be represented accurately by blending some individualities of other subjects (dataset) is crucial. To confirm this, we applied a cross-validation test to the dataset. After training our neural network with the data of 44/45 subjects, we optimized the personalization weight β to approximate the HRTF data of the remaining subject. In this experiment, we optimized β using a hybrid CMA-ES. For the cost function to minimize, we used a squared mean error of power spectrums and phases between data outputted from the system and the target (the rest subject's data). We conducted this cross validation for all subjects in the CIPIC dataset (45 times). Figure 5.15 (top) shows the convergence curves of several optimizations, and Figure 5.15 (middle) shows a comparison of the prediction error between PCA (32 axis) and our algorithm on the horizontal plane for all the subject. In addition, Figure 5.15 (bottom) shows power spectrums of a subject at two directions between optimized and target HRTFs. These figures show that all optimizations converge to a similar level of error, and our system can approximate the HRTF of a new user by blending the individualities of the trained dataset. Note that we computed PCA using HRTF at all the directions in this experiment. Naturally, PCA computed for each direction would provide much higher score than this result. However, it requires more than ten thousands of PCA vectors for representing HRTF at all the direction, and it is impractical for our target problem.

Validation with Synthetic Data: We validated the ability of our algorithm to generate appropriate HRTFs by using synthetic data. For synthetic data

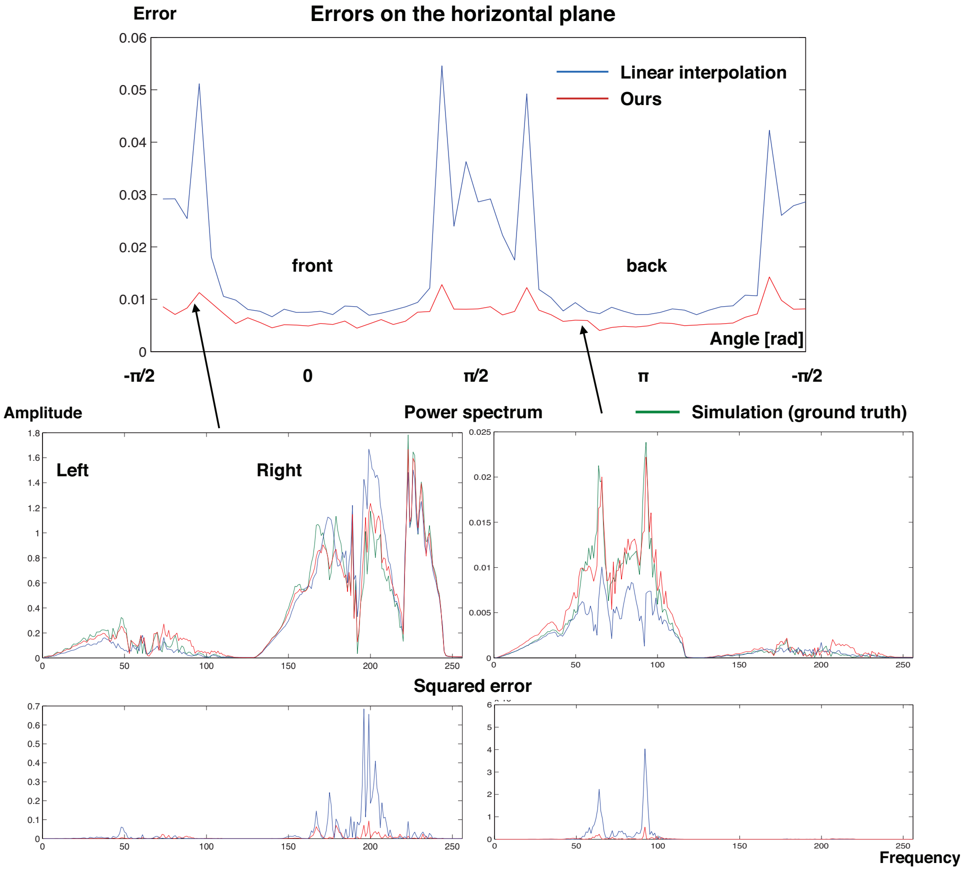


Figure 5.17: The result for predicting a new virtual HRTF. Top row shows the errors on the horizontal plane of the obstacle. Bottom shows comparisons of power spectrums between our neural network and simple linear interpolation at two directions.

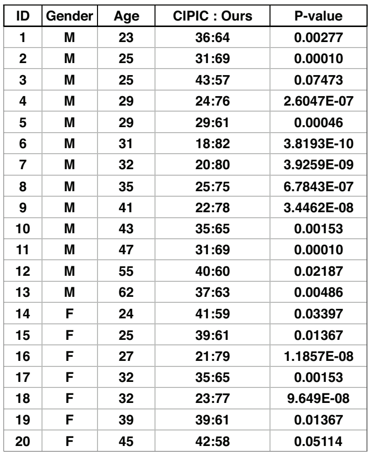
generation, we simulated an open-space 3D acoustic field around an elliptically shaped obstacle using the finite difference time domain (FDTD) method with a perfectly matched layer [94, 95, 141]. FDTD is an established method used to simulate an impulse response. We set up a three-meter cubic domain with 128×128×128 uniform grid as the simulation field. The simulations were conducted using four shapes as obstacles. Each shape was an oval sphere and had different radii (shape 1: x. 0.2m, y. 0.2m z. 0.2m, shape 2: x. 0.7m, y. 0.2m z. 0.2m, shape 3: x. 0.2m, y. 0.7m z. 0.2m, shape 4: x. 0.2m, y. 0.2m z. 0.7m). We recorded the sound pressure at two opposite sides of each obstacle, modeled on the ears of humans (Figure 5.16). A sound source was spherically rotated around the obstacle from a position of one meter from the center of the obstacle, and generated an impulse response. We recorded the first 256 samples of sound pressure from the moment each impulse was started. These simulated sounds became virtual HRTFs, and we used them for training data in our algorithm.

We evaluate our method by predicting virtual HRTFs around a new shape obstacle that is not included in the training data. We set the new shape as the intermediate shape between shape 1 and 2 (x. 0.45m, y. 0.2m z. 0.2m). Figure 5.16 shows a comparison of the errors from the simulated HRTF between our neural network and simple linear interpolation of simulation 1 and 2 on the horizontal plane. To generate this HRTF with our system, we set β=(0.5, 0.5, 0, 0). With all directions, our algorithm had fewer errors than did linear interpolation, which means that our algorithm can generate appropriate HRTFs. Figure 5.17: bottom shows a comparison of the predicted results of power spectrum at two directions when using both our method and linear interpolation. Apparently, our neural network can estimate specific peaks and dips than can linear interpolation.

5.7.3 User Study

We employed 20 participants (male:female = 13:7, age: 22∼62) and optimized HRTFs for them using our system. The experiment consisted of three steps. In the first step, we investigated the best-fitted CIPIC HRTF for each participant. Here, we conducted a progressive comparison [122] to estimate the best-fitted HRTF. Therefore, a participant compared 44 pairs of HRTFs because 45 CIPIC HRTFs exist. We showed each participant 44 pairs of test sounds convolved by two CIPIC HRTFs using the same GUI in our system (FIgure 5.4); the participant indicated the best among them. This task consumed 15∼20 min. The best CIPIC HRTFs were used in the third step as baselines to evaluate our optimized HRTF.

We next requested that the participants calibrate their HRTFs using our system. We ordered each participant to answer at least 100 pairwise comparisons. We did not decide the maximum times of comparisons and when a



ID	Gender	Age	CIPIC : Ours	P-value
1	M	23	36:64	0.00277
2	M	25	31:69	0.00010
3	M	25	43:57	0.07473
4	M	29	24:76	2.6047E-07
5	M	29	29:61	0.00046
6	M	31	18:82	3.8193E-10
7	M	32	20:80	3.9259E-09
8	M	35	25:75	6.7843E-07
9	M	41	22:78	3.4462E-08
10	M	43	35:65	0.00153
11	M	47	31:69	0.00010
12	M	55	40:60	0.02187
13	M	62	37:63	0.00486
14	F	24	41:59	0.03397
15	F	25	39:61	0.01367
16	F	27	21:79	1.1857E-08
17	F	32	35:65	0.00153
18	F	32	23:77	9.649E-08
19	F	39	39:61	0.01367
20	F	45	42:58	0.05114

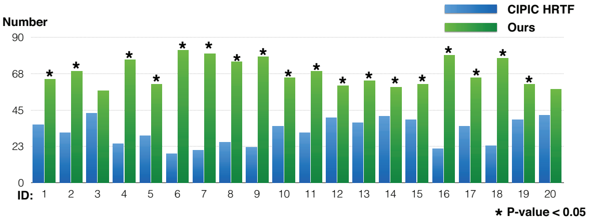


Figure 5.18: The result of user study. The third column shows how many numbers of options are selected as better HRTF for each participant between best fitted CIPIC HRTF and optimized HRTF by our system.

participant indicated that he or she was satisfied, we stopped the calibration. We measured calibration time and the number of mouse clicks (number of pairwise comparisons) for each participant. The participants used the UI described in §5.3. The calibration consumed 20∼35 minutes for each participant. The number of pairwise comparison was 109∼202 times. This calibration time was much shorter than the actual measurement for obtaining fitted HRTFs.

After each calibration, we conducted a blind listening test to compare the HRTFs obtained using our method and the best fitted CIPIC HRTFs. In this step, we showed each participant 100 pairs of test sounds. One of the test sounds in each pair was convolved by an optimized HRTF and the other was convolved by the best CIPIC HRTF for the participant. The test sound to convolve was randomly selected from 10 prepared sounds (e.g., short music, helicopter, and speech) and played 100 times. We requested that each participant use the same GUI as during the calibration to select one test sound from each pair that showed better spatialization . We requested that each participant select only either 1 or 5 from among the option buttons. The order (A or B) in which test sounds were played using optimized HRTF and best CIPIC HRTF for each presented pair was random. We did not inform the participants whether the selection was the optimized HRTF or the best CIPIC HRTF. Figure 5.18 shows the number of times the better HRTF was selected by each participant. These results show that the optimized HRTFs were significantly better for almost all participants (p-value<0.05 by Chi-squared test for 18/20 subjects) than were the best CIPIC HRTFs, indicating that our system successfully optimizes HRTFs for individual users.

5.8 Limitations and Future Work

This study presented a fully perceptual-based HRTF optimizer for individual users using a machine learning technique. However, several limitations are remained for future work to overcome.

First, it is not clear how well the dataset we used spans the space of HRTF. Investigating the variance of HRTFs is an important future work. Second, evaluating the absolute quality of final results after calibration is difficult. To analyze this, more user studies (including the evaluation of azimuth error, elevation error, front-back reversal rate, and externalization percentage) should be conducted in future work. Third, although our approach achieves much faster optimization of HRTF compared to existing methods, approximately 30 minutes of calibration time is still long. In our user study, we found almost all the optimized individualization weight vectors became "very sparse ". This means a few elements in the vector are large and other elements are close to zero. This observation implies that we could decrease the calibration time by reducing the dimensions in the individualization weights that are approaching to zero during the calibration, or using sparse coding techniques for the individualization weights in future work. Finally, our approach for training the individualities would not scale well when the number of subjects in the dataset is much larger. A possible solution is clustering the subjects beforehand (using some features of HRTF like principal component vectors), and reducing the number of "domain" DoFs.

Our adaptive layer could be used in a wide range of other applications (e.g., from mesh morphing to animation generation). However, the following problem remains: training time increases in proportion to different categories of individuality, as the number of training parameters (tensors at each layer) increase. To address this, clustering and reducing the DoFs of the extracted individualities in each adaptive layer should be considered. Recent studies using a Gram matrix at each layer in a DNN for image styling [34] is expected to be useful in solving this problem.

Chapter 6

Conclusion

In this thesis, we have addressed the problems that interfere with the improving the interactivity of computational sound by proposing several user interfaces using precomputation. This chapter contains a brief summary and discussions of the work, together with some possibilities for future research.

6.1 Summary

In computational sound, traditional approach uses fixed set of sound data that is prepared beforehand, and simply plays these sounds at runtime. However, such limited interactions reduce the virtual experience of digital computer entertainment. Unfortunately, although various methods have been proposed to address this problem, it is difficult to use them in actual scene because of several reasons. For example, sound rendering techniques using physical simulation achieve rich interactions with the user without preparing many sound clips beforehand. However, it is difficult to design the sound's timbre by manipulating unintuitive physical parameters directly. For generating rich sounds, various synthesizers such as singing voice synthesizer were developed. Although these synthesizers can feedback to the user with various changing sound's texture, the user has a difficulty to control their enormous parameters using standard input devices. To render appropriate 3D spatial sounds that response the user's movement, the researches for HRTF have a long history. However, because the measurement cost for HRTF is too expensive, it is difficult to use these techniques for auditory users.

We addressed these problems by reducing each computational cost or user's operation cost of computational sound techniques by three novel user interfaces using precomputation.

First, for designing physically based sound, we developed an example based method that does not require the user to consider unintuitive physical parameters, and makes physically based sound designable. This used an interactive material optimization that was accomplished by a dramatically fast vibrational analysis using precomputed mesh simplification algorithm using machine learning and hierarchical component mode synthesis. Second, for making a singing voice synthesizer controllable, we presented a method to estimate latent lyrics as higher DoF parameters from the input of lower DoF control device using machine learning of lyrics dataset at precomputation phase. Finally, in the calibration of 3d spatial audio, we proposed a machine learning model that allows adaptation of the system to a specific user using individual and non-individual factors of dataset which are extracted at precomputation. This method makes existing 3D audio spatialization techniques usable for auditory users. Thus, in this thesis, we introduced three algorithms to make interactive computational sound techniques usable in practical scene.

- Example Based Design Interface by Precomputation.
- Controlling high DoFs parameters with low DoFs input device by precomputation.
- Extraction and reduction of the essential factors from a dataset by precomputation.

These algorithms were designed to improve the interactivity of computational sound applications at runtime with precomputation. We demonstrated the effectiveness of these approaches by implementing various systems.

6.1.1 Example Based Design Interface by Precomputation

We demonstrated an example based approach to design physically based sound of an object. Our algorithm inversely optimizes internal material distribution of the 3D model from assigned sounds by the user. A difficulty for achieving this was that the computational cost of vibrational analysis was expensive because our material optimization requires it iteratively. If an optimization takes a long time, the trial and error design procedure of the user becomes difficult. To address this, we developed an acceleration technique for a vibrational analysis by precomputation. Our technique consists of data-driven finite element coarsening of the mesh and hierarchical component mode synthesis with efficient error correction. Our data-driven online coarsening extended Chen et al.'s method [15], and reduced the mesh size at runtime using machine learning technique at precomputation phase. It can handle a large range of continuous material settings by reducing the material parameter space, and can be evaluated with a constant cost for a large amount of datasets using regression forests. Additionally, our highly parallelized hierarchical component mode synthesis using precomputed mesh segmentation extended conventional methods [134] to efficiently compute approximate solutions of modal analysis, and our error correction algorithm efficiently improved its accuracy. Using our method, our example based design framework runs at interactive rate, and this provides practical design workflow for the user.

6.1.2 Controlling High DoFs Parameters with Low DoFs Input Device by Precomputation

We explored an algorithm for controlling enormous parameters of a singing voice synthesizer by standard input device (e.g., piano keyboard). The core of our interest for this target was how the low DoFs (a few keys) of such input device can be mapped to high DoFs parameters (lyrics and melodies) by reducing the user's operation cost. To address this, we trained a machine learning model by pre- defined lyric dataset at precomputation phase. At runtime, we assigned vowels onto a piano keyboard, and identified the most plausible character sequence in the predefined lyrics by finding the corresponding vowel sequence using a probabilistic alignment technique. This allows real-time control of both high DoFs lyrics and melodies of a song with a low DoFs keyboard. Our system does not require the user to input the vowel sequences strictly in the order of the original lyrics, because the system estimates the plausible lyrics using a probabilistic model. This allows the user to jump to arbitrary positions in the lyrics including backtracking. Additionally, our system allows the user to make mistakes, freeing the player from paying excessive attention to vowel input. Our method was not limited for the parameter of a singing voice synthesizer, and can be used for controlling various kinds of high DoFs parameters.

6.1.3 Extracting The Essential Factors from A Dataset by Precomputation for Runtime Calibration

We introduced a method to calibrate 3D audio spatialization for a specific user as a first step of fully perceptual calibration approach. The difficulty for achieving this was that HRTF for 3D audio spatialization has a large parameter space, and can not be explored directly. To address this, we demonstrated an algorithm to extract essential factors (individual factors) from a dataset and reduce its dimensionality at precomputation phase. Using this trained model, we calibrated HRTF for a new user by blending the extracted factors in nonlinear space at runtime. This had an advantage to efficiently explore a large design space with reduced parameters that reduces the user's calibration cost. We evaluated our algorithm by several quantitative validations and a user study. We demonstrated that the calibrated HRTFs obtained using our method outperformed best HRTFs in the data set in a user study with 20 users. Such two steps machine learning technique, automatic feature extraction by precomputation and optimizing it at runtime, could be widely used for other domain problems.

6.2 Future Directions

In this section, we discuss future research for improving the interactivity of other domain problems by precomputation that was shown to be necessary in this study.

6.2.1 Example Based Design Interface

Our example based interface for physically based sound design inversely optimizes physical parameters from the intended result, which provides intuitive design workflow for the user. There is a significant advantage to obtain the corresponding input parameters from an intended example. Similarly, there are many applications that have a benefit by employing an example based interface (e.g., animation design for deformable objects, image texture design). However the computational cost for the inverse problem solving which is required for an example based method is usually much expensive than the forward computation. In such situations, using precomputation similar to our method for accelerating the computation could be useful. Also in sound applications, simulation method of sound radiation from a vibrating object would be an interesting target. Widely used simulation method uses boundary element method (BEM) for precomputation [57], which is expensive to execute at an interactive rate. This limits the object's shape and vibrational properties to be static. It might be addressed by accelerating this BEM with similar approach to our method. For example, we could train a machine learning model by object's shape and material as input and the radiation specification as output.

6.2.2 Context-Aware Control

Our method for realtime control, that learns possible contexts at precomputation phase and achieves the user's context aware control at runtime, is essentially equivalent to the prediction of the user's next movement according to the inputs until current time. Recent several studies using recurrent neural networks including long-short term memory (LSTM) have similar motivations [35]. For example, by predicting the next movement of hand writing on a touch panel of the user, the response would be quickly and the user experience might be increased. Our method also could be applied similar target. There are two advantages of our method compared with their methods. First, our method have faster interaction responses. Second, our method requires quite fewer training data. Thus, in the problem domains such as music performance and sport in which our advantages are effective, our method could be useful.

6.2.3 Calibrating High Dimensional Design Parameters for A Black Box Function

Our two steps algorithm, that extracts the essential factors of a dataset at precomputation phase and use them for runtime optimization, could be used for many other domain problems that treat black box systems including human perceptions and tastes (e.g., image enhancement, flavor adjustment for a customer, and sound equalizing by hall concert public address operator). Although our method described in this thesis extracted the individualities by subject, we can alternatively extract the factors by an arbitrary group. For example, to develop a new good synthetic seasoning for a specific user, we could gather a dataset of the relationships between rating of test seasonings and each subjects's favorite food by crowd sourcing, and extract the individualities by favorite food at precomputation phase. At runtime, we could optimize the blending weights for generating a new seasoning for a new user. An interesting issue in this problem is that although the extracted factors are based on favorite foods of gathered subjects, the new user does not need to care about them and only tests the generated seasoning from the system. Thus, we could hide the latent specification used for gathering dataset, and focus the target user on only the target to design.

An another future direction is to extend the format of individual factors. For example, although we used a vector for representing the individuality in this thesis, extending the vector to a tensor alternatively is an interesting issue. By using a tensor, it has a possibility to extract the individuality determined by combination of multiple properties. For example, we might extract the specific factor of peoples who like pizza and coke. However, the computational cost for training would be exponentially increased according to the rank of the tensor. Low rank decomposition of the tensor would address this problem, but it remains as future work.

References

[1] Aean, M. A. The missing link: Modal synthesis. in representations of musical signals. MIT Press Cambridge, pp. 269–298.

[2] Algazi, V. R., Duda, R. O., Thompson, D. M., and Avendano, C. The cipic hrtf database. In *IEEE Workshop on Applications of Signal Processing to Audio and Electroacoustics* (2001), pp. 99–102.

[3] Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein gan. In *arXiv:1701.07875* (2017).

[4] Arthur, D., and Vassilvitskii, S. k-means++: The advantages of careful seeding. *In Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematic* (2007), 1027–1024.

[5] Bamptom, M. C. C., and Craig, R. R. Coupling of substructures for dynamic analyses. *AIAA Journal 6*, 7 (1968), 1313–1319.

[6] Barbic, J., and James, D. L. Real-time subspace integration for st. venant-kirchhoff deformable models. In *ACM Transaction on Graphics* (2005), vol. 24.

[7] Bathe, K.-J. The subspace iteration method - revisited. *Computers and Structures 126* (2013), 977–983.

[8] Bathe, K. J., and Ramaswamy, S. An accelerated subspace iteration method. *Computer Methods in Applied Mechanics and Engineering 23* (1980), 313–331.

[9] Bharaj, G., Levin, D. I. W., Tompkin, J., Fei, Y., Pfister, H., Matusik, W., and Zheng, C. Computational design of metallophone contact sounds. In *ACM Trans. on Graphics (SIGGRAPH Asia 2015)* (2015), vol. 34, pp. 223:1–223:13.

[10] Bilinski, P., Ahrens, J., Thomas, M., Tashev, I., and Platt, J. Hrtf magnitude synthesis via sparse representation of anthropometric features. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Proces* (2014).

[11] Blaauw, M., and Bonada, J. A neural parametric singing synthesizer. In *arXiv:1704.03809* (2017).

[12] Bridson, R. Fluid simulation for computer graphics. In *CRC Press* (2015).

[13] Brochu, E., Brochu, T., and de Freitas, N. A bayesian interactive optimization approach to procedural animation design. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2010), pp. 103–112.

[14] Buschek, D., Schoenleben, O., and Oulasvirta, A. Improving accuracy in back-of-device multitouch typing: A clustering-based approach to keyboard updating. In *IUI'14, ACM Press* (2014), ACM.

[15] Chen, D., Levin, D. I. W., Sueda, S., and Matusik, W. Data-driven finite elements for geometry and material design. In *ACM Trans. on Graphics (SIGGRAPH 2015)* (2015), vol. 34, pp. 74:1–74:10.

[16] Chen, S.-H., and Pan, H. H. Guyan reduction. *Communications in Applied Numerical Methods 4*, 4 (1988), 549–556.

[17] Chen, X., Liu, X., and Jia, Y. Combining evolution strategy and gradient descent method for discriminative learning of bayesian classifiers. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation* (2009), no. 8, pp. 507–514.

[18] Choi, M. G., and Ko, H.-S. Modal warping: Real- time simulation of large rotational deformation and manipulation. In *IEEE Transactions on Visualization and Computer Graphics* (2005).

[19] Christensen, C. L., Koutsouris, G., and Rindel, J. H. Estimating absorption of materials to match room model against existing room using a genetic algorithm. In *Forum Acusticum* (2014).

[20] Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). In *The International Conference on Learning Representations (ICLR)* (2016).

[21] Cook, P. R. Real sound synthesis for interactive applications. In *A K Peters/CRC Press* (2002).

[22] Cook, P. R. La bella voce e la macchina (the beautiful voice and the machine): A history of technology and the expressive voice. In *Net Art and Public Art* (2017).

[23] Corbett, R., van den Doel, K., Lloyd, J. E., and Heidrich, W. Timbrefields: 3d interactive sound models for real-time audio. *Presence: Teleoperators and Virtual Environments - Special section: Advances in interactive multimodal telepresent systems 16*, 6 (2007), 643–654.

[24] Courbariaux, M., and Bengio, Y. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. In *arXiv* (2016).

[25] d'Alessandro, C., Nicolas, Wang, J., Pritchard, R., and Fels, S. Bringing bio-mechanical modelling of the opal complex as a mapping layer for performative voice synthesis. In *9th International Seminar on Speech Production (ISSP)* (2011), ACM.

[26] Dhulekar, L., and Shah, S. K. Musical audio beat tracking using hidden markov model. In *International Journal of Science and Research)* (2012).

[27] Dudley, H. The vocoder. In *Bell Laboratories Record* (1939), vol. 18.

[28] Duraiswaini, R., Zotkin, D., and Gumerov, N. Interpolation and range extrapolation of hrtfs [head related transfer functions]. In *ICASSP* (2004).

[29] Engel, J., Resnick, C., Roberts, A., Dieleman, S., Eck, D., Simonyan, K., and Norouzi, M. Neural audio synthesis of musical notes with wavenet autoencoders. In *arXiv:1704.01279* (2017).

[30] Fan, Y., Qian, Y., Xie, F.-L., and Soong, F. K. Tts synthesis with bidirectional lstm based recurrent neural networks. In *Interspeech* (2014).

[31] Feugere, L., d'Alessandro, C., and Doval, B. Performative voice synthesis for edutainment in acoustic phonetics and singing: a case study using the cantor digitalis. In *5th International ICST Conference* (2013), ACM.

[32] Gargi, U., and Gossweiler, R. Beat tracking based on multiple-agent architecture - a real-time beat tracking system for audio signals. In *The Second International Conference on Multiagent Systems* (1996), ACM.

[33] Gargi, U., and Gossweiler, R. Quicksuggest: Character prediction for improved text entry on web appliances. In *5th International ICST Conference* (2013), ACM.

[34] Gatys, L. A., Ecker, A. S., and Bethges, M. Image style transfer using convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[35] Graves, A. Tts synthesis with bidirectional lstm based recurrent neural networks. In *arXiv:1308.0850* (2013).

[36] Gregor, M., and Carsten, D. Precomputing sound scattering for structured surfaces. In *Proceedings of the 14th Eurographics Symposium on Parallel Graphics and Visualization* (2014).

[37] Grijalva, F., Martini, L., Goldenstein, S., and Florencio, D. Anthropometric-based customization of head-related transfer functions using isomap in the horizontal plane. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (2014).

[38] Grzeszczuk, R., Terzopoulos, D., and Hinton, G. Neuroanimator: Fast neural network emulation and control of physics-based model. In *ACM Trans. on Graphics (SIGGRAPH)* (1998).

[39] Gumerov, N. A., O ’ Donovan, A. E., Duraiswami, R., and Zotkin, D. N. Computation of the head-related transfer function via the fast multipole accelerated boundary element method and its spherical harmonic representation. In *J. Acoust Soc. Am* (2010), vol. 127.

[40] Hansen, N., Muller, S., and Koumoutsakos, P. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). In *Evolutionary Computation* (2003), no. 11, pp. 1–18.

[41] Hauser, K. K., Shen, C., and O'Brien, J. F. Interactive deformation using modal analysis with constraints. In *In Proc. of Graphics Interface* (2003), pp. 247–256.

[42] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition* (2015).

[43] Heisele, B., Ho, P., and Poggio, T. Face recognition with sup- port vector machines: Global versus component-based approach. In *ICCV* (2001).

[44] Holden, D., Komura, T., and Saito, J. Phase-functioned neural networks for character control. In *ACM Transactions on Graphics (SIGGRAPH)* (2017).

[45] Holden, D., Saito, J., and Komura, T. A deep learning framework for character motion synthesis and editing. *ACM Transaction on Graphics (SIGGRAPH), 35*, 4 (2016), 138:1–138:11.

[46] Holden, D., Taku, J. S., and Komura. Neural network ambient occlusion. In *ACM SIGGRAPH ASIA Technical Briefs* (2016).

[47] Holzl, J. A global model for hrtf individualization by adjustment of principal component weights. In *Diploma Thesis* (2014).

[48] Hu, H., Zhou, L., Ma, H., and Wu, Z. Hrtf personalization based on artificial neural net- work in individual virtual auditory space. In *Applied Acoustics* (2008), vol. 69, pp. 163–172.

[49] Huang, Q., and Fang, Y. Modeling personalized head- related impulse response using support vector regressions. In *J. Shanghai Univ* (2009).

[50] Huang, Q., and Zhuang, Q. Hrir personalisation using support vector regression in independent feature space. In *Electron. Letter* (2009), vol. 45.

[51] Hurty, W. C. Dynamic analysis of structural systems using component modes. *AIAA Journal 3* (1965), 678–685.

[52] Iida, P., Ishii, Y., , and Nishioka, S. Personalization of head-related transfer functions in the median plane based on the anthropometry of the listener ’ s pinnae. In *J. Acoust. Soc. America.* (2014).

[53] Iizuka, S., Simo-Serra, E., and Ishikawa, H. Let there be color!: Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. In *ACM Transactions on Graphics (SIGGRAPH)* (2016).

[54] Iizuka, S., Simo-Serra, E., and Ishikawa, H. Globally and locally consistent image completion. In *ACM Transactions on Graphics (SIGGRAPH)* (2017).

[55] Ioffe, S., and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)* (2015).

[56] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial nets. In *CVPR* (2017).

[57] James, D. L., Barbic, J., and Pai, D. K. Precomputed acoustic transfer: output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Transactions on Graphics (TOG) 25*, 3 (2006), 987–995.

[58] James, D. L., and Fatahalian, K. Precomputing interactive dynamic deformable scenes. In *ACM Transactions on Graphics (SIGGRAPH)* (2003).

[59] Jin, C. T., Guillon, P., Epain, N., Zolfaghari, R., van Schaik, A., Tew, A. I., Hetherington, C., and Thorpe, J. Creating the sydney york morphological and acoustic recordings of ears database. In *IEEE Transactions on Multimedia* (2014), vol. 16.

[60] Joder, C., and Essid, S.and Richard, G. A conditional random field framework for robust and scalable audio-to-score matching. In *IEEE TASLP, Vol.19, No.8* (2011), ACM.

[61] K Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognitions. In *CILR* (2016).

[62] Kahana, Y., and Nelson, P. A. Boundary element simulations of the transfer function of human heads and baffled pinnae using accurate geometric model. In *Journal of sound and vibration* (2007), pp. 552–579.

[63] Kamvar, M., and Baluja, S. Query suggestions for mobile search: Understanding usage patterns. In *ACM Computer Human Interaction (CHI)* (2008), ACM.

[64] Kaneko, S., Suenaga, T., and Sekine, S. Deepearnet: individualizing spatial audio with photography, ear shape modeling, and neural networks. In *AES Conference on Audio for Virtual and Augmented Reality* (2016).

[65] Karras, T., Aila, T., Laine, S., Herva, A., and Lehtinen, J. Audio-driven facial animation by joint end-to-end learning of pose and emotion. In *ACM Transactions on Graphics (SIGGRAPH)* (2017).

[66] Katz, B. F. Boundary element method calculation of individual head-related transfer function. i. rigid model calculation. In *The Journal of the Acoustical Society of America* (2001).

[67] Kenmochi, H., and Ohshita, H. Vocaloid - commercial singing synthesizer based on sample concatenation. In *INTERSPEECH* (2007), ACM.

[68] Kim, D., Koh, W., Narain, R., Fatahalian, K., Treuille, A., and O'Brien, J. F. Near-exhaustive precomputation of secondary cloth effects. In *ACM Transactions on Graphics (SIGGRAPH)* (2013).

[69] Kingma, and P, D. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems* (2014).

[70] Kingma, D., and Ba, J. P. Adam: A method for stochastic optimization. In *CoRR abs/1412.6980* (2014).

[71] Kingma, D. P., and Welling, M. Auto-encoding variational bayes. In *The International Conference on Learning Representations (ICLR)* (2014).

[72] Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. In *IEEE Computer,* (2009), vol. 42, IEEE, pp. 30–37.

[73] Koyama, Y., Sakamoto, D., and Igarashi, T. Crowd-powered parameter analysis for visual design exploration. In *Proceedings of the 27th annual ACM symposium on User interface software and technology (UIST)* (2014), pp. 65–74.

[74] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classi- fication with deep convolutional neural networks. In *Proc. NIPS* (2012).

[75] Ladický, L., Jeong, S., Solenthaler, B., Pollefeys, M., and Gross, M. Data-driven fluid simulations using regression forests. In *ACM Trans. on Graphics (SIGGRAPH Asia 2015)* (2015), vol. 34, pp. 199:1–199:9.

[76] Langendijk, E., and Bronkhorst, A. Fidelity of three-dimensional-sound reproduction using a virtual auditory display. In *J. Acoust. Soc. Am* (2000).

[77] Leeuw, J. D. Derivatives of generalized eigen systems with applications. Tech. rep., UCLA Department of Statistics, 2007.

[78] Lemouton, S., and Schwarz, D. Score following: State of the art and new developments. In *New Interfaces for Musical Expression (NIME)* (2003), ACM.

[79] Li, D., Fei, Y., and Zheng, C. Interactive acoustic transfer approximation for modal sound. *ACM Transactions on Graphics (TOG) 35*, 1 (2015).

[80] Li, S., Huang, J., de Goes, F., Jin, X., Bao, H., and Desbrun, M. Space-time editing of elastic motion through material optimization and reduction. *ACM Trans. Graph. 33*, 4 (2014), 108:1–108:10.

[81] Liao, J., Yao, Y., Yuan, L., Hua, G., and Kang, S. B. Visual attribute transfer through deep image analogy. In *arXiv:1705.01088* (2017).

[82] Liu, M.-Y., Breuel, T., and Kautz, J. Unsupervised image-to-image translation networks. In *arXiv:1702.01478* (2017).

[83] Lloyd, D. B., Raghuvanshi, N., and Govindaraju, N. K. Sound synthesis for impact sounds in video games. In *In Symposium on Interactive 3D Graphics and Game* (2011), ACM, p. 7.

[84] Luo, Y., Zotkin, D. N., Daume, H., and Duraiswami, R. Kernel regression for head-related transfer function interpolation and spectral extrema extraction. In *ICASSP* (2013).

[85] Luo, Y., Zotkin, D. N., and Duraiswami, R. Virtual autoencoder based recommendation system for individualizing head-related transfer functions. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics* (2013).

[86] Maezawa, A., Okuno, H. G., Ogata, T., and Goto, M. Polyphonic audio-to-score alignment based on bayesian latent harmonic allocation hidden markov model. In *International Conference on Acoustics, Speech and Signal Processing(ICASSP)* (2011), ACM.

[87] Masi, G., and Stettner, L. Ergodicity of hidden markov models. In *Journal of Mathematics of Control, Signals and Systems*, vol. 17.

[88] Matheron, G. Principles of geostatistics. In *Economic Geology* (1963), pp. 1246–1266.

[89] Matsunaga, N., and Hirahara, T. Reexamination of fast head-related transfer function measurement by reciprocal method. In *J. Acoust Soc. Ja* (2010), vol. 31, 6.

[90] Maxwell, C. B., and Bindel, D. Modal parameter tracking for shape-changing geometric objects. In *Proceedings of the 10th International Conference on Digital Audio Effects (DAFx 07)* (2007).

[91] Meshram, A., Mehra, R., and Manocha, D. Efficient hrtf computation using adaptive rectangular decomposition. In *AES 55th International Conference* (2014).

[92] Middlebrooks, J. Virtual localization improved by scaling non-individualized external-ear transfer functions in frequency. In *J. Acoust. Soc. Am. 106* (1999).

[93] Miwa, M., and Sakonda, N. Brothers' button to phoneme transfer standard for international language. In *Departmental Bulletin Paper for Nagoya University of Arts and Science, Vol.6* (2013), ACM.

[94] Mokhtari, P., Takemoto, H., Nishimura, R., and Kato, H. Computer simulation of hrtfs for personalization of 3d audio. In *In Universal Communication, IEEE. ISUC ' 08. Second International Symposium* (2008), pp. 435–440.

[95] Mokhtari, P., Takemoto, H., Nishimura, R., and Kato, H. Computer simulation of kemar ' s head-related transfer functions: verification with measurements and acoustic effects of modifying head shape and pinna concavity. In *Principles and Applications of Spatial Hearing* (2010), pp. 179–194.

[96] Moller., H., Sorensen., M., C.B, J., and HammershOi. Binaural technique: do we need individual recordings? In *J. Audio Eng. Soc. 44, 451e469* (1996).

[97] Monks, M., Oh, B. M., and Dorsey, J. Audioptimization: goal- based acoustic design. In *IEEE Computer Graphics and Applications* (2000), vol. 20.

[98] Mozer, M. C. A focused back- propagation algorithm for temporal pattern recognition. In *Hillsdale, NJ: Lawrence Erlbaum Associates* (1995).

[99] Nakamura, M., Koyama, Y., Sakamoto, D., and Igarashi, T. An interactive design system of free-formed bamboo-copters. In *Computer Graphics Forum (Pacific Graphics)* (2016).

[100] Nakamura, T., Nakamura, E., and Sagayama, S. Acoustic score following to musical performance with errors and arbitrary repeats and skips for automatic accompaniment. In *Sound and Music Computing Conference (SMC)* (2013), ACM.

[101] Nalbach, O., Arabadzhiyska, E., Mehta, D., Seidel, H.-P., and Ritschel, T. Deep shading: Convolutional neural networks for screen- space shading. In *EGSR* (2017).

[102] Nava, G. Inverse sound rendering: In-situ estimation of surface acoustic impedance for acoustic simulation and design of real indoor environments. In *Ph.D. dissertation, University of Tokyo* (2006).

[103] Ng, J. Y.-H., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., and Toderici, G. Beyond short snippets: Deep networks for video classification. In *CVPR* (2015).

[104] O'Brien, J. F., Cook, P. R., and Essl, G. Synthesizing sounds from physically based motion. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques (SIGGRAPH 2001)* (2001), pp. 529–536.

[105] P, M. Flick input for realtime singing synthesizer. In *http://www.nicovideo.jp/watch/sm17357529* (2013), ACM.

[106] Pai, D. K., van den Doel, K., James, D. L., Lang, J., Lloyd, J. E., Richmond, J. L., and Yau, S. H. Scanning physical interaction behavior of 3d objects. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques (SIGGRAPH 2001)* (2001), pp. 87–96.

[107] PFN. Paintschainer. In *https://github.com/pfnet/PaintsChainer* (2017).

[108] Rabiner, L. A tutorial on hidden markov models and selected applications in speech recognition. In *IEEE* (1989).

[109] Ren, Z., Wang, R., Snyder, J., Zhou, K., Liu, X., Sun, B., Sloan, P.-P., Bao, H., and andBaining Guo, Q. P. Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. In *ACM Transaction on Graphics* (2006).

[110] Ren, Z., and Yeh, H. Synthesizing contact sounds between textured models. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques (SIGGRAPH 2001)* (2001), pp. 537–544.

[111] Ren, Z., Yeh, H., and Lin, M. C. Example-guided physically based modal sound synthesis. *ACM Transactions on Graphics (TOG) 32*, 1 (2013), 987–995.

[112] Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *The International Conference on Machine Learning (ICML)* (2014).

[113] Ribeiro, A., and Igarashi, T. Sketch-editing games: Human-machine communication, game theory and applications. In *Proc . User Interface Software and Technology (UIST)* (2012).

[114] Sak, H., Senior, A., and Beaufays, F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Interspeech* (2010).

[115] Saksela, K., Botts, J., , and Savioja, L. Optimization of absorption placement using geometrical acoustic models and least squares. In *The Journal of the Acoustical Society of America* (2015), vol. 137.

[116] Sifakis, E., and Barbic, J. Fem simulation of 3d deformable solids: A practitioner ' s guide to theory, discretization and model reduction. In *ACM SIGGRAPH Courses* (2012).

[117] Silfverberg, M., MacKenzie, S., and Korhonen, P. Predicting text entry speed on mobile phones. In *The ACM Conference on Human Factors in Computing Systems (CHI)* (2000), ACM.

[118] Simo-Serra, E., Iizuka, S., Sasaki, K., and Ishikawa, H. Learning to simplify: Fully convolutional networks for rough sketch cleanup. In *ACM Transactions on Graphics (SIGGRAPH)* (2016).

[119] Skouras, M., Thomaszewski, B., Coros, S., Bickel, B., and Gross, M. Computational design of actuated deformable characters. In *ACM Trans. on Graphics (SIGGRAPH 2013)* (2013), vol. 32, pp. 82:1–82:10.

[120] Sloan, P.-P., Kautz, J., and Snyder, J. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *ACM Transactions on Graphics* (2002), vol. 21.

[121] Sohn, K., Lee, H., and Yan, X. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems* (2015).

[122] Takahama, R., Kamishima, T., and Kashima, H. Progressive comparison for ranking estimation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)* (2016).

[123] Takahashi, N. Aenet: Learning deep audio features for video analysis. In *arXiv:1701.00599* (2017).

[124] Takahashi, N., Gygli, M., Pfister, B., and Gool, L. V. Deep convolutional neural networks and data augmentation for acoustic event detection. In *Interspeech* (2016).

[125] Takemoto, T., Baba, T., and Katayori, H. A real-time singing generator using typing and humming "hanautau" based on an agile software development. In *Interaction, Information Processing Society of Japan* (2014), ACM.

[126] Tokuda, K., Yoshimura, T., Masuko, T., Kobayashi, T., and Kitamura, T. Speech parameter generation algorithms for hmm-based speech synthesis. In *IEEE International Conference on Acoustics, Speech, and Signal Processing* (2000).

[127] Tran, D., Bourdev, L., Fergus, R., Torresani, L., and Paluri, M. Learning spatiotemporal features with 3d convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)* (2015).

[128] Ueda, Y., Uchiyama, Y., Nishimoto, T., Ono, N., and Sagayama, S. Hmm-based approach for automatic chord detection using refined acoustic features. In *ICASSP* (2003).

[129] Um, K., Hu, X., and Thuerey, N. Liquid splash modeling with neural networks. In *arXiv:1704.03809* (2017).

[130] Umetani, N., Koyama, Y., Schdmit, R., and Igarashi, T. Pteromys: Interactive design and optimization of free-formed free-flight model airplanes. In *ACM Transaction on Graphics (SIGGRAPH 2014)* (2014).

[131] Umetani, N., Mitani, J., Igarashi, T., and Takayama, K. Designing custom-made metallophone with concurrent eigenanalysis. In *In Proceedings of the Conference on New Interfaces for Musical Expression (NIME)* (2010), ACM, pp. 26–20.

[132] van den Doel, K., Kry, P. G., and Pai, D. K. Foleyautomatic: physically-based sound effects for interactive simulation and animation. *In Proceedings of the 28th annual conference on Computer graphics and interactive techniques (SIGGRAPH 2001)* (2001), 537–544.

[133] van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K.-r. Wavenet: A generative model for raw audio. In *CoRR abs/1609.03499* (2016).

[134] Wang, H., Lu, T., Au, O. K.-C., and Tai, C.-L. Spectral 3d mesh segmentation with a novel single segmentation field. *Graphical Models 76*, 5 (2014), 440–456.

[135] Wang, Y., Skerry-Ryan, R., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., Le, Q., Agiomyrgiannakis, Y., Clark, R., and Saurous, R. A. Tacotron: A fully end-to-end text-to-speech synthesis model. In *arXiv:1703.10135* (2017).

[136] Wang, Z., and Chan, C. F. Hrir customization using common factor decomposition and joint support vector regression. In *Eur. Signal Process. Conf* (2013).

[137] Watanabe, K., Matsubayashi, Y., Inui, K., and Goto, M. Automatic japanese lyric generation based on the hierarchical sructure of the song. In *The Association for Natural Language Processing* (2014), ACM.

[138] Wenzel, E., Arruda, D. J., and Kistler, D. Localization using non-individualized head-related transfer functions. In *J. Acoust. Soc. Am. 94* (1993).

[139] Wenzel, E., and Foster, S. Perceptual consequences of interpolating head-related transfer functions during spatial synthesis. In *in Proceedings of the 1993 Workshop on Applications of Signal Processing to Audio and Acoustics* (1993).

[140] Wobbrock, L., and Wigdor, J. Typing on flat glass: Examining ten-finger expert typing patterns on touch surfaces. In *ACM Computer Human Interaction (CHI)* (2011), ACM.

[141] Xiao, T., and Liu, Q. H. Finite difference computation of head-related transfer function for human hearing. In *The Journal of the Acoustical Society of America* (2003).

[142] Xu, H., Li, Y., Chen, Y., and Barbič, J. Interactive material design using model reduction. *ACM Transactions on Graphics (TOG) 34*, 2 (2015), 14.

[143] YAMAHA. Tyros.

[144] Yamamoto, K., Kagami, S., Hamano, K., and Kashiwase, K. The development of a text input interface for realtime japanese vocal keyboard. In *Journal of Information Processing, Vol.21, No.2* (2013), ACM.

[145] Yamamoto, R., Sako, S., and Kitamura, T. Robust on-line algorithm for realtime audio-to-score alignment based on a delayed decision and anticipation framework. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2013), ACM.

[146] Yamasaki, S., Nishiwaki, S., Yamada, T., Izui, K., and Yoshimura, M. A structural optimization method based on the level set method using a new geometry-based re-initialization scheme. *International Journal for Numerical Methods in Engineering 12* (2010), 1580–1624.

[147] Yang, Y., Li, D., Xu, W., Tian, Y., and Zheng, C. Expediting precomputation for reduced deformable simulation. In *ACM Trans. on Graphics (SIGGRAPH Asia 2015)* (2015), vol. 34, pp. 243:1–243:13.

[148] Yang, Y., Xu, W., Guo, X., Zhou, K., and Guo, B. Boundary-aware multidomain subspace deformation. In *IEEE Transactions on Visualization and Computer Graphics* (2013), vol. 19, pp. 1633–1645.

[149] Yina, J., Vossb, H., and Chena, P. Improving eigenpairs of automated multilevel substructuring with subspace iterations. *Computers and Structures 119* (2013), 115–124.

[150] Young, S., Evermann, G., Kershaw, D., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., and Woodland, P. The htk book. In *http://htk.eng.cam.ac.uk* (2017).

[151] Yua, Y., Jang, I. G., Kimc, I. K., and Kwaka, B. M. Nodal line optimization and its application to violin top plate design. *Journal of Sound and Vibration 329* (2010), 4785–4796.

[152] Yua, Y., Jang, I. G., and Kwaka, B. M. Topology optimization for a frequency response and its application to a violin bridge. *Journal Structural and Multidisciplinary Optimization 48* (2013), 4627–636.

[153] Yumer, M. E., Asente, P., Mech, R., and Kara, L. B. Procedural modeling using autoencoder networks. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)* (2015), ACM.

[154] Zhang, X., Zilles, S., and Holte, R. Improved query suggestion by query search. In *The 35th German Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence* (2012), ACM.

[155] Zheng, C., and James, D. L. Toward high-quality modal contact sound. In *ACM Trans. on Graphics (SIGGRAPH 2011)* (2011), vol. 30, p. 38.

[156] Zotkin, D. N., Duraiswami, R., and Davis, L. S. Rendering localized spatial audio in a virtual auditory space. In *IEEE Transactions on Multimedia, vol. 6(4)* (2004).

[157] Zotkin, D. N., Duraiswami, R., Grassi, E., and Gumerov, N. A. Fast head-related transfer function measurement via reciprocity. In *J. Acoust Soc. Am* (2006), vol. 120.

[158] Zwicker, E., and Fastl, H. *Psychoacoustics: Facts and models, 2nd updated edition*. Vol. 254. Springer New York, 1999.

Appendix A

Modal Sound Synthesis

Modal sound synthesis is based on the traditional linear modal analysis techniques [1]. Modal analysis builds a reduced space for the elastic motion equation using of the solution of the generalized eigenvalue problem of the stiffness and mass matrices

$$\boldsymbol{KU} = \Lambda \boldsymbol{MU} \tag{6.1}$$

where Λ is a diagonal matrix, containing the eigenvalues $(\lambda_1, ...\lambda_r)$, and $\boldsymbol{U}$ is a matrix in which each column is a eigenvector $(u_1, ...u_r)$ corresponding to the eigenvalue of Eq. (6.1). We can decouple the system by retaining r $(r << n)$ number of the eigen-pairs that have larger energies into the form

$$\ddot{q} + \boldsymbol{C}\dot{q} + \boldsymbol{\Lambda} q = \boldsymbol{U}^T f_{ext} \tag{6.2}$$

where q is the generalized displacements, and $u = \boldsymbol{U}q$. The solution of Eq. (6.2) are a bank of modes that are attenuated sinusoids that has various frequencies and amplitudes. The i-th mode is represented as

$$q_i = a_i e^{-d_i t} sin(2\pi f_i t + \theta_i) \tag{6.3}$$

$$f_i = \frac{\sqrt{\lambda_i}}{2\pi} \tag{6.4}$$

where t is the time, f_i is the frequency of the mode, d_i is the damping coefficient, a_i is the excited amplitude, and θ_i is the initial phase. In general, θ_i is ignored and then, i-th mode parameter is represented as $\phi_i = (f_i, d_i, a_i)$. As the result, a collection of mode parameters $(\phi_1, ..., \phi_r)$ determines the vibration specification of the structure, and modal sound synthesis uses it as the sounding wave.

Appendix B

Gradient Computation for Vibrational Optimization

For the gradient computation in our design problem, we need the derivative of each mode frequency and amplitude with respect to the reduced design parameter z. The mode frequencies and amplitudes are functions of the eigenpairs, and the derivatives of k-th eigenvalue λ_k and eigenvector u_k with respect to z is represented as

$$\begin{aligned} \frac{\partial \lambda_k}{\partial z} &= u_k^T \left(\frac{\partial \boldsymbol{K}}{\partial z} \right) u_k \\ \frac{\partial u_k}{\partial z} &= -(\boldsymbol{K} - \lambda_k \boldsymbol{M})^+ \left(\frac{\partial \boldsymbol{K}}{\partial z} \right) u_k, \end{aligned} \tag{6.5}$$

where $superscript^+$ denotes the pseudo inverse (Moore-Penrose inverse). We approximate this pseudo inverse using the result of the already obtained generalized eigenproblem by $(\boldsymbol{K} - \lambda_k \boldsymbol{M})^+ \risingdotseq \boldsymbol{U}(\boldsymbol{\Lambda} - \lambda_k \boldsymbol{M})^+ \boldsymbol{U}^T$. For the details of derivation of the derivative of eigenpair, please read [77]. In addition, the derivative of the stiffness matrix $\boldsymbol{K}$ with respect to z is represented as

$$\frac{\partial \boldsymbol{K}}{\partial z} = \sum_{e=1}^{M} \frac{\partial \boldsymbol{K}}{\partial Y_e} \frac{\partial Y_e}{\partial z} = \sum_{e=1}^{M} \frac{\partial \boldsymbol{K}}{\partial Y_e} \Phi_e, \tag{6.6}$$

where all but 24 entries (3 dimensions × 8 nodes) in $\frac{\partial \boldsymbol{K}}{\partial z}$ are zero, and $\boldsymbol{K}$ is a linear function for Y_e, then the computational costs are cheap.

Appendix C

Hybrid Optimization Scheme of Evolutional Strategy and Gradient Descent Approach

We employ a hybrid optimization scheme [17] of evolutional strategy (we used CMA-ES[40]) and gradient descent approach (we used Quasi-Newton method) for solve our design problem.This hybrid approach first updates the design parameter using the gradient informations for searching the local optima, and to escape from bad local optima, the evolutional strategy part generates the offspring by two characteristic variation operators, and additive Gaussian mutation alternatively.

$$q^g = QuasiNewtonUpdate(z^g), \tag{6.7}$$

$$z^{(g+1)} = q^{(g)} + \rho^{(g)} \boldsymbol{B}^{(g)} \boldsymbol{D}^{(g)} y^{(g)}, \tag{6.8}$$

$$\boldsymbol{B}^{(g)} \boldsymbol{D}^{(g)} y^{(g)} \ \sim \ N(0, C^{(g)}), \tag{6.9}$$

where $z^{(g)}$, g and $\rho^{(g)}$ are the design parameters, iteration step and a global step size respectively. $y^{(g)} \ \sim \ N(0, \boldsymbol{I})$ are independent realizations of a normally distributed random vector with zero mean and covariance matrix equal to the identity matrix $\boldsymbol{I}$, and $C^{(g)}$ denotes the covariance matrix which is computed using $q^{(g)}$ and $z^{(g)}$ (please read [17] for the details).

Appendix D

Variational AutoEncoder

Our network architecture is based on variational AutoEncoder [71, 112] Variational AutoEncoder is a generative model of a deep neural network. We used conditional variational AutoEncoder [69, 121]. It consists of a decoder $p_\theta(x, y|z)$ and the variational posterior encoder $q_\phi(z|x, y)$, where x, y, and $z$are input, description label, and latent variable respectively, and produces the parameters of each distribution after a series of non-linear transformations. Both the model (θ) and variational (ϕ) parameters will be jointly optimized with stochastic gradient variational Bayes (SGVB) algorithm according to a lower bound on the log-likelihood. This parametrization allows us to capture most of the salient information of x and y in the embedding z. By choosing a Gaussian posterior $q_\phi(z|x, y)$ and standard isotropic Gaussian prior $p(z) \sim N(0, I)$ we can obtain the following lower bound.

$$\begin{aligned} log\ p_\theta(y|x) &= -KL(q_\phi(z|x)||p_\theta(z|x)) \\ &\quad +\mathbb{E}_{q_\phi(z|x)}\left[-log\ q_\phi(z|x) + log\ p_\theta(x, z)\right] \\ &\geq -KL(q_\phi(z|x, y)||p_\theta(z)) \\ &\quad +\mathbb{E}_{q_\phi}(z|x, y)\left[log\ p_\theta(y|x, z)\right] \end{aligned} \tag{6.10}$$

and the empirical lower bound is written as

$$\begin{aligned} log\ p_\theta(y|x) \quad &\geq \quad -KL(q_\phi(z|x, y)||p_\theta(z)) \\ &\quad +\tfrac{1}{L}\textstyle\sum_{l=1}^{L} log\ p_\theta(y|x, z^{(l)}), \end{aligned} \tag{6.11}$$

where $KL()$ denotes Kullback-Leibler divergence. Finally, the total loss to minimize can be formulated as

$$\begin{aligned} L &= |x' - x|^2 \\ &- \frac{1}{2}Mean\left(\sum(1 + z_{var} - z_{mean}^2 - e^{z_{var}})\right), \end{aligned} \tag{6.12}$$

where $Mean()$ represents the mean average. Note that we use only the HRTF data at the sample direction (the center position in a patch although we sampled 5×5 directions for input) for original input x becomes a 512 dimensions vector.

Appendix E

DNN Architecture for HRTF

Figure 6.1, Figure 6.2 and Figure 6.3 show each block of our neural network. In these figures, red arrows denotes adaptive layers described in §5.6, and blue arrows denotes common linear layer. We divide the HRTF patch x by each channel and input them separately as the power spectrum channels of LR x_{fl} and x_{fr} and the time signals of LR x_{pl} and x_{pr}. Similar to the conventional variational AutoEncoder, the architecture has an encoder (Figure 6.2) and a decoder (Figure 6.3). The encoder extracts latent variables from the input, and the decoder outputs reconstructed HRTFs in the format described in the previous section from the sampled latent variables.

Figure 6.4 shows the equations, where $ELU()$ denotes an exponential linear unit [20]. $Conv()$ is the 3D HRTF patch convolution and $Adapt()$ represents our adaptive layer (which we describe in a later section). The middle of the network has feed forward connections that represent residual networks. $Merge()$ joints the four channels of vectors into a single vector. At the encoder, the system first decomposes the HRTF patch into four channels (left power spectrum x_{fl}, right power spectrum x_{fr}, left time signal x_{pl}, and right time signal x_{pr}), and inputs each channel into independent convolutional layers. After applying the convolutions, the system merges the four channels into a single vector as x_0 and transforms it into x_1 by applying an adaptive layer in order to reduce the number of dimensions. This thus becomes an input vector of the variational AutoEncoder. Using these two vectors, which represent a sample direction and subject label, respectively, as well as the input vector after the convolutions, the encoder of our variational AutoEncoder generates the mean z_{mean} and variation z_{var} vectors of a Gaussian distribution (latent variables in Figure 6.3). The latent variables can be generated from this Gaussian distribution $z_p \sim \mathbb{N}(z_{mean}, \frac{1}{2}z_{var})$. At the decoder, the system uses the two vectors y and s that match the encoder and latent variables z_p, and reconstructs the center HRTFs of the input HRTF patches of the four channels (L-ch power spectrum, R-ch power spectrum, L-ch time signal, and R-ch time signal) through residual adaptive network layers.

For optimization, we use the mini-batch Adam algorithm [70] with mini-batch size 16. We set the numbers of the layer N as 4. In our experiment,

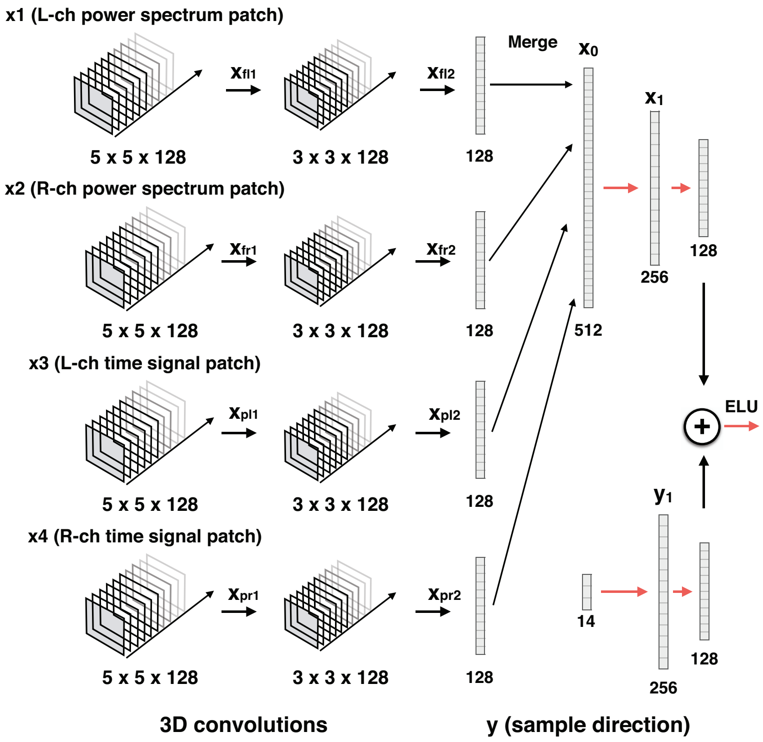


Figure 6.1: 3D convolutions block architecture.

the number of the layers is not so sensitive to the result. In addition, we insert the batch normalization layers [55] before all nonlinear units (ELU function). We randomly restart the optimizers at each layer independently during training. We found that this layer-dependent stochastic restart accelerates the convergence of the network.

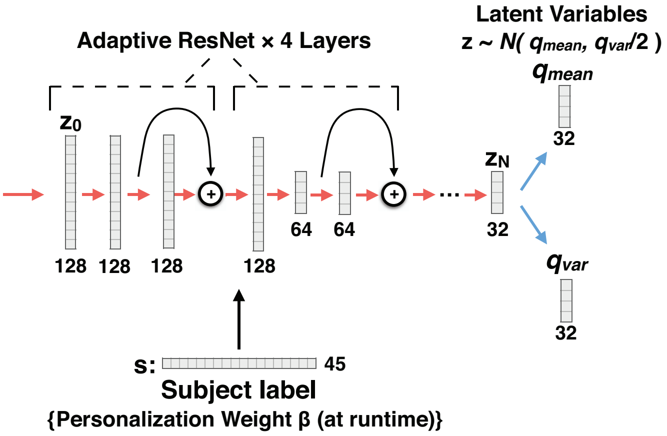


Figure 6.2: The encoder block extracts latent variables from input.

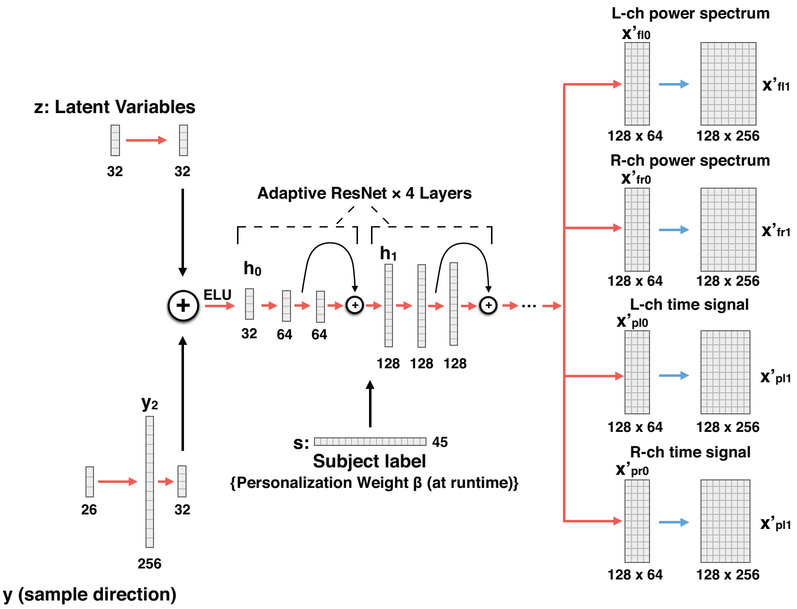
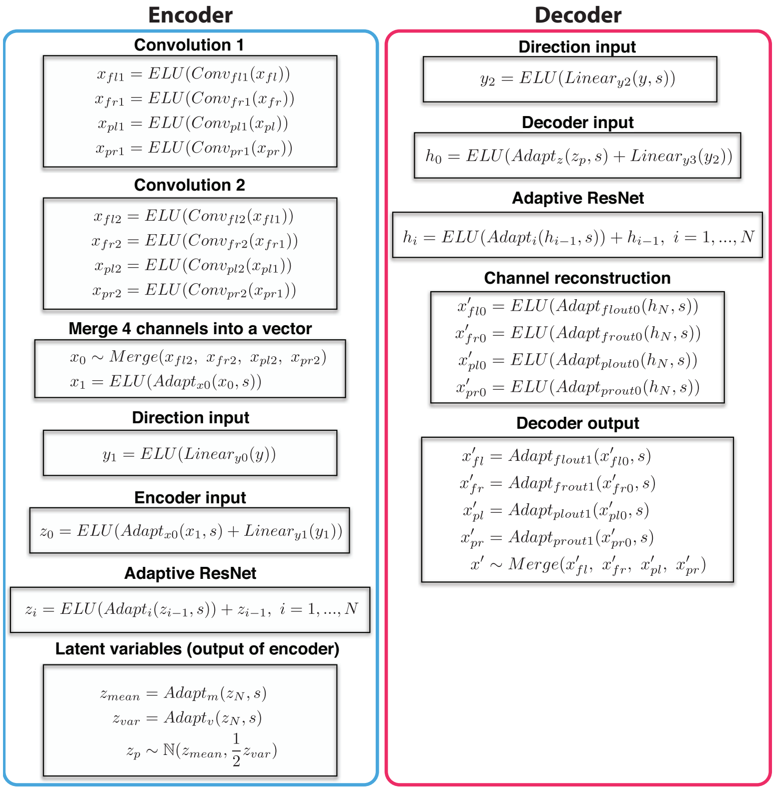


Figure 6.3: The decoder block generates reconstructed HRTF from latent variables.



Encoder

Convolution 1

$$x_{fl1} = ELU(Conv_{fl1}(x_{fl}))$$
$$x_{fr1} = ELU(Conv_{fr1}(x_{fr}))$$
$$x_{pl1} = ELU(Conv_{pl1}(x_{pl}))$$
$$x_{pr1} = ELU(Conv_{pr1}(x_{pr}))$$

Convolution 2

$$x_{fl2} = ELU(Conv_{fl2}(x_{fl1}))$$
$$x_{fr2} = ELU(Conv_{fr2}(x_{fr1}))$$
$$x_{pl2} = ELU(Conv_{pl2}(x_{pl1}))$$
$$x_{pr2} = ELU(Conv_{pr2}(x_{pr1}))$$

Merge 4 channels into a vector

$$x_0 \sim Merge(x_{fl2},\ x_{fr2},\ x_{pl2},\ x_{pr2})$$
$$x_1 = ELU(Adapt_{x0}(x_0, s))$$

Direction input

$$y_1 = ELU(Linear_{y0}(y))$$

Encoder input

$$z_0 = ELU(Adapt_{x0}(x_1, s) + Linear_{y1}(y_1))$$

Adaptive ResNet

$$z_i = ELU(Adapt_i(z_{i-1}, s)) + z_{i-1},\ i = 1, ..., N$$

Latent variables (output of encoder)

$$z_{mean} = Adapt_m(z_N, s)$$
$$z_{var} = Adapt_v(z_N, s)$$
$$z_p \sim \mathbb{N}(z_{mean}, \frac{1}{2} z_{var})$$

Decoder

Direction input

$$y_2 = ELU(Linear_{y2}(y, s))$$

Decoder input

$$h_0 = ELU(Adapt_z(z_p, s) + Linear_{y3}(y_2))$$

Adaptive ResNet

$$h_i = ELU(Adapt_i(h_{i-1}, s)) + h_{i-1},\ i = 1, ..., N$$

Channel reconstruction

$$x'_{fl0} = ELU(Adapt_{flout0}(h_N, s))$$
$$x'_{fr0} = ELU(Adapt_{frout0}(h_N, s))$$
$$x'_{pl0} = ELU(Adapt_{plout0}(h_N, s))$$
$$x'_{pr0} = ELU(Adapt_{prout0}(h_N, s))$$

Decoder output

$$x'_{fl} = Adapt_{flout1}(x'_{fl0}, s)$$
$$x'_{fr} = Adapt_{frout1}(x'_{fr0}, s)$$
$$x'_{pl} = Adapt_{plout1}(x'_{pl0}, s)$$
$$x'_{pr} = Adapt_{prout1}(x'_{pr0}, s)$$
$$x' \sim Merge(x'_{fl},\ x'_{fr},\ x'_{pl},\ x'_{pr})$$

Figure 6.4: The equations of our DNN.