Descriptor Control of Sound Transformations and Mosaicing Synthesis

Graham Coleman

TESI DOCTORAL UPF / 2015

Directors de la tesi:

Dr. Xavier Serra i Dr. Jordi Bonada
Department of Information and Communication Technologies

upf. Universitat Pompeu Fabra Barcelona

To the memories of my late aunt, Cynthia Hope Emrich,
and of my late godfather, Carl Thomas Poss.

Acknowledgements

First, I would like to acknowledge some of the great teachers from Athens (GA) who were partly responsible for my development, including Kevin Manglitz, who challenged all of us in his class (and introduced us to competitive math in Mathcounts); and Sukla Bhattacharyya, whose AP chemistry class was a first formative experience in science. For my musical development I'd like to thank the Huthmakers, Buddy, Dixie, and Anna; and Milton Masciadri, who trained me as a double bass player.

Thanks to Sean Molley (GA GHP '97) who helped me get a start in programming, to Andrew Osborne, who adopted me to his advisor's organic chem lab for the summer, and to William York, for additional research experience writing chemical structure translator scripts.

At Georgia Tech, where I came into contact with experts in computer audio, including Maribeth Gandy, who taught an inspiring class and who supported a first independent project, Mitch Parry, Mark Guzdial, Parag Chordia, who taught an MIR class, and Jason Freeman. Thanks to Aaron Bobick, who brought me into his Pattern Recognition lab (then GVU), and to his students Amos Johnson and Raffay Hamid who led me to collaborate on their work in Activity Recognition in video.

Thanks to Xavier Serra, who took me into the fold at MTG, to Jordi Bonada, who did the main advising work for the project, and to Fernando Villavicencio and Esteban Maestre, who also advised in different stages of the project. It was an honor to meet and work alongside many other researchers and creators; e.g. fellow members of the Audio Processing group: Jordi Janer, Ricard, Sašo, Maarten, Merlijn, Martí, Marc, and Alfonso;

entering PhD class of Ferdinand, Joan, Inês, Piotr, and Moha; Mathieu, Vincent, Cyril, Amaury, Sertan, Gerard, Anna, Frederic, Owen, Josh, Javi, Stefan, Carles, Dani, Marco, Justin, Agustín, Martin, Dmitri, Oscar, Alex, Martin K., Enric, Sergi, Rafael, Emilia, Perfe, and Ping, among others.

I would also like to thank the listening test subjects who gave their time and musical opinions.

Along the way, several gave advice regarding the project, those included Gabor Lugosi, Albert Ferrer, and Bob Sturm. Additionally, many have answered queries about their work, and Michael Grant and Stephen Becker have provided guidance in using their convex optimization toolkit, TFOCS.

To my colleagues at Hörtech, where I had the great luck to be able to continue in audio programming and research, including my colleagues at F&E: Volker Hohmann, Tobias Herzke, Giso Grimm, Daniel Berg, Thomas Wittkop, Jan-Hendrik Fleßner, and many colleagues at the Uni. Oldenburg, and especially to those who gave feedback regarding the design of the listening tests such as Thomas Bisitz, Rainer Huber, Markus Meis, and Matthias Vormann.

And especially to those who proofread and critiqued parts of the manuscript: including Kamil Adiloglu, Matthias Vormann, Dan MacKinlay, and Gerard Llorach.

Computer clusters were also important for generating audio results; but as they cannot yet be directly thanked, I thank their representatives, Hendrik Kayser, of the MEDI-Schroeder cluster at the Uni. Oldenburg, and Iván Jiménez, of the HPC DTIC cluster at the Uni. Pompeu Fabra.

A special thanks to the groups and individuals who provided a musical home in Barcelona. I refer to the Cor de Cambra Anton Bruckner, the Barcelona Laptop Orchestra; and also to Andrés Lewin and PHONOS, Gerard Roma, Josep Comajuncosas, and Sergi from NIU, for their support and collaboration in livecoding activities.

To my parents, brother, and Xana; without whose love and support I would be lost.

Additionally:
This work was supported by a doctoral scholarship from the Department of Information and Communication Technologies (DTIC) of Pompeu Fabra University, and partially by the EU-ICT SAME Project (ref. number 215749).

Abstract

Sampling, as a musical or synthesis technique, is a way to reuse recorded musical expressions. In this dissertation, several ways to expand sampling synthesis are explored, especially mosaicing synthesis, which imitates target signals by transforming and compositing source sounds, in the manner of a mosaic made of broken tile.

One branch of extension consists of the automatic control of sound transformations towards targets defined in a perceptual space. The approach chosen uses models that predict how the input sound will be transformed as a function of the selected parameters. In one setting, the models are known, and numerical search can be used to find sufficient parameters; in the other, they are unknown and must be learned from data.

Another branch focuses on the sampling itself. By mixing multiple sounds at once, perhaps it is possible to make better imitations, e.g. in terms of the harmony of the target. However, using mixtures leads to new computational problems, especially if properties like continuity, important to high quality sampling synthesis, are to be preserved.

A new mosaicing synthesizer is presented which incorporates all of these elements: supporting automatic control of sound transformations using models, mixtures supported by perceptually relevant harmony and timbre descriptors, and preservation of continuity of the sampling context and transformation parameters. Using listening tests, the proposed hybrid algorithm was compared against classic and contemporary algorithms, and the hybrid algorithm performed well on a variety of quality measures.

Resum

El mostreig, com a tècnica musical o de síntesi, és una manera de reutilitzar expressions musicals enregistrades. En aquesta dissertació s'exploren estratègies d'ampliar la síntesi de mostreig, sobretot la síntesi de "mosaicing". Aquesta última tracta d'imitar un senyal objectiu a partir d'un conjunt de senyals font, transformant i ordenant aquests senyals en el temps, de la mateixa manera que es faria un mosaic amb rajoles trencades.

Una d'aquestes ampliacions de síntesi consisteix en el control automàtic de transformacions de so cap a objectius definits a l'espai perceptiu. L'estratègia elegida utilitza models que prediuen com es transformarà el so d'entrada en funció d'uns paràmetres seleccionats. En un cas, els models són coneguts, i cerques númeriques es poden fer servir per trobar paràmetres suficients; en l'altre, els models són desconeguts i s'han d'aprendre a partir de les dades.

Una altra ampliació es centra en el mostreig en si. Mesclant múltiples sons a la vegada, potser és possible fer millors imitacions, més específicament millorar l'harmonia del resultat, entre d'altres. Tot i així, utilitzar múltiples mescles crea nous problemes computacionals, especialment si propietats com la continuïtat, important per a la síntesis de mostreig d'alta qualitat, han de ser preservades.

En aquesta tesi es presenta un nou sintetitzador mosaicing que incorpora tots aquests elements: control automàtic de transformacions de so fent servir models, mescles a partir de descriptors d'harmonia i timbre perceptuals, i preservació de la continuïtat del context de mostreig i dels paràmetres de transformació. Fent servir proves d'escolta, l'algorisme híbrid proposat va ser comparat amb algorismes clàssics i contemporanis: l'algorisme híbrid va donar resultats positius a una varietat de mesures de qualitat.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

This dissertation explores, and attempts to expand experimental methods in sampling synthesis, the technology based on the use and manipulation of recorded sounds. This is done through two main strategies. One strategy is to incorporate automatically controlled sound transformations, which is a problem of general interest in computer music and a complex problem in itself. Another strategy is to incorporate mixtures of sounds, which is not possible using existing synthesis methods. These strategies are applied to a setting analogous to and inspired by image texture transfer.

1.1 Motivation

Recorded sound and music have over the last century become increasingly ubiquitous. Mobile devices can hold large music collections and stream larger music collections from the Internet. With an affordable field recorder with solid-state electronics and high quality microphones, new sounds can be recorded in high fidelity by anyone. These sounds can then be edited with one of many Digital audio workstation (DAW) applications to yield produced sound or music recordings, or be used as assets for interactive performance.

Sound transformations (sound effects) play a large role in the production, editing, and performance of sound. Many qualities of the sounds can be modified, including pitch and time-scale, aspects of timbre and sense of space; allowing them to fit in (or stick out) of a given context.

In the DAW, the composer chooses sound sources from their own ideas;

isolates and manipulates audio clips manually, as if cutting tape and taping together the cuts; and applies effects and modulates their parameters, as if pushing buttons and turning knobs on guitar pedals. This alone, in an interactive process, is enough to help create works of high complexity which would be difficult to create using only physical tools.

Still, the range of possibilities offered by sampling and multiple sound effects necessitates many new choices that must be made by the composer.

But what if the recordings themselves have their own opinions about which one should be sampled and how, in a given context? And if the effects also have opinions on which one should be used and how, on which sound? When the materials are able to speak their own opinions, the intricate work of editing might be better supported. (This idea is inspired by Latour, who claims that scientists act as spokesmen for formerly unrepresented objects [—, 1987, Section 2.A.2, Spokesmen and women]).

The keys to accessing those opinions are: descriptors of a sound, which give information on where a sound is in a perceptual space; and models of sound transformations, which point in the directions that each transformation is able to modify the source sounds in that space. When the composer is able to define a target in the perceptual space, the materials can offer clues on which sources to use, and which directions to move in the transformation space.

The following application, texture transfer, which was developed originally in image processing research, can provide a model setting in which this added information can be used in synthesis.

1.2 Texture Synthesis and Transfer

Texture in images is usually only implicitly defined, but texture mapping, the art of mapping an image source to a surface in 3D graphics, seems to have motivated the application known as texture synthesis [Heeger and Bergen, 1995, 1. Introduction], as image textures needed to be mapped to larger objects without repetition and to surface geometry without distortion. Their representation of image texture, the image pyramid, is essentially a bandpass filter that decomposes the energy at different scales and edges in different orientations into layer images, along with image histograms that summarize the color distributions at each layer. This representation allows solving the first problem, that of generating a larger non-repeating image from a smaller example, in the following way. The new image is synthesized by taking an input noise image, and equalizing it at different scales and orientations to match the source texture histograms.

The stochastic, noisy approach of Heeger and Bergen, in which texture is modeled explicitly as a parametric distribution, however, does not seem to be the most common approach. Rather, more common is sampling directly from the image sources, usually in such a way that respects the local statistics of the image texture. For example, the systems of [Efros and Leung, 1999; Ashikhmin, 2001], copy pixels progressively in order to match neighborhoods of each filled pixel; while the systems of [Efros and Freeman, 2001; Kwatra et al., 2003] copy contiguous patches of pixels that minimize artifacts at boundary edges.

Several systems take those or similar representations, and apply them to synthesis with the following additional objective: that the output image should also generally match a second target image. This application, texture transfer, produces a rendering that overall resembles the structure of the target image, while the local detail and pattern reflects the source texture. Examples of these systems based on copying patches or pixels include [Ashikhmin, 2001; Efros and Freeman, 2001; Ingram and Bhat, 2004].

By contrast, the recent image texture transfer system [Gatys et al., 2015] uses the following approach perhaps more akin to Heeger and Bergen: the source and target images are used to train convolutional neural networks, which encode visual features at different scales. Then, an input noise image is modified so it fits the target networks at the coarse scales, while fitting the source networks at the detail scales, as optimized with a gradient descent method.

The image and sound domains differ in many respects. One main difference is that images of objects occlude each other in the visual field; for each image pixel, there is generally only one object present except at edges. While physical occlusion also affects sounding objects, imparting directional and spectral characteristics, the signals of sounding objects in a scene readily mix and an occluded object will nonetheless be heard by the listener, though it might be masked by other objects in the time-frequency domain. Therefore, mixtures should be more important in the sound than the image domain.

In the survey of [Schwarz, 2011], a great number of sound texture synthesis methods are reviewed. To pick just one example, the systems of [Kersten and Purwins, 2010, 2012] decompose the source sounds (fire and water sounds) into multiscale signal atoms (Debauchies wavelets) and then train generative models in order to sample new sounds from the models.

Despite the fact that there are many documented systems of sound texture synthesis, it is harder to find references to sound texture transfer in the literature. Regardless, this general idea (explained further) served as inspiration in this thesis, both in the development of new synthesis methods (Chapter 6) and in their evaluation (Chapter 7). Mosaicing synthesis (overview in Chapter 2), in which sounds are selected and sampled according to a descriptor target, is one possible way of fulfilling the goal of sound texture transfer; it is also the strategy chosen in developing the methods of this thesis. This strategy in the sound domain seems analogous to the predominant texture transfer strategy in the image domain of non-parametric sampling, i.e. copying and assembling 2D patches of the source image.

The survey of Schwarz adopts the restrictive definition of sound texture of [Saint-Arnaud and Popat, 1998] which excludes speech and music (as they convey information and long-term structure). Going further, [Schwarz, 2011, Section 1.1.1. What Sound Texture is Not] excludes contact sounds (as fine structure properties do not remain constant) and soundscapes (as they contain informative events). By excluding sources that have structure at larger scales, these restrictions might help to create models that can generate sound texture without having to model and generate long-term information.

However, one need not restrict the use of these kind of sources in sound texture transfer. That is, given a target structure defined as a time-sequence of descriptors for harmony and timbre (and possibly others), the aforementioned sound types (music, speech, contact sounds, and soundscape events) can be used as source textures, adapting them to the musical and sonic context defined by that target. In this setting, as opposed to texture synthesis, no model of large-scale structure is needed, because that structure is provided rather by the target signal.

In the opinion of the author, the difference between structure and texture can be seen as simply a matter of scale. The mechanics of the image texture transfer systems seem to support this idea. For example, the system of Gatys et al. works explicitly by combining two different models at different scales, while systems based on copying neighborhoods [Ashikhmin, 2001; Efros and Freeman, 2001] trade-off implicitly between the coarse scale of the target and the local neighborhoods modeled in the source.

In a way, this idea of combining signals at different scales fit the limitations of sampling synthesis, because it is naturally constrained to working with the descriptor trajectories contained in the source material (local detail that could be seen as texture) as lamented by Wessel et al. [1998], rather than being able to modulate controls arbitrarily as in, e.g. additive synthesis.

The proposed algorithm of Chapter 6 takes the following approach to sound texture transfer: it searches for continuous sections of source sounds that when transformed and mixed together, best match with the target signal in the descriptor space. This strategy favors sampling longer continuous sections, and listening tests show (Chapter 7) that it was better at preserving the sense of the source than alternatives.

Limiting the synthesis to what can be directly sampled and transformed (as was done) could be considered a conservative approach, given that we have also just mentioned systems that model spectral-temporal content and generate new signals (as in systems of Kersten and Purwins). Alternative approaches, based on either modeling and generating new signals, or simply recombining spectral or atomic components, are certainly also interesting and viable (some of which will probably be bolstered by recent advances in recurrent neural networks); some systems of this type are also listed in Section 2.1.4 (p. 22).

But rather than using generative signal models or spectral/atomic recombination to generalize and adapt source signals, the approach of the framework of this dissertation was to use transformations for this generalization. This came from a belief that transformations, when appropriately controlled and limited, could still result in relatively "natural" sounds that continue to reflect the sampled source material.

In the next section, we discuss how sound transformations, although not an important part of the image texture transfer systems mentioned, can play a key role in sound texture transfer.

1.3 The Role of Transformations

As mentioned, the approach taken in this thesis is that of automatically selecting, transforming, and compositing samples, referred to as Mosaicing (further detailed in Chapter 2). In a survey of systems of this type, referred to by Schwarz as Concatenative corpus-based synthesis systems, a similar definition is given [Schwarz, 2007]:

> Concatenative corpus-based synthesis methods use a large database of source sounds segmented into units and a unit selection algo-

> rithm that finds the sequence of units that match best the sound or phrase to be synthesized, called the target.

The emphasis in this analysis seems to be on "large database". The role of transformations is acknowledged,

> The selected units can then be transformed to fully match the target specification and are concatenated.

i.e. it is assumed that the selected units can be transformed to completely match the target. Despite this pivotal role for transformations, they are given only minimal attention within the framework of the survey, and the feasibility of such transformations are not examined.

> However, if the database is sufficiently large, the probability is high that a matching unit will be found, so the need to apply transformations is reduced.

The survey seems to prefer this ideal scenario, i.e. when the database is large enough so that the need for transformation is mostly avoided.

However, the user of the system might not wish to use a large database, because they would have less choice in which material is used. Rather, they might want to apply samples from a specific sub-corpus purposefully, as short as minutes or seconds of audio, onto another musical or sonic context, as in the texture transfer application (previous section). As the purpose of texture synthesis is to extrapolate new material from a small sample, the purpose of texture transfer is to adapt a possibly small sample to a new setting.

Other sampling applications that seem to be appropriate for small or medium source databases can be found in Chapter 2, State of the Art.

In a smaller sample, the coverage of the source in descriptor space should be quite limited. To give an informal example (from my early experiments), take as source a short melodic excerpt in which the pitches (and pitch classes) are limited. Chances are, some of the pitches in the target will not be present in the source, and so transposing some source units will be necessary.

Figure 1.1 shows the similarity, here computed with the dot product of chroma vectors, for a given target frame with respect to the source frames (the distribution in blue) and all possible transpositions (the distribution in green). When transpositions are added via resampling, the given sources are multiplied in descriptor space, giving the wider similarity spectrum shown.

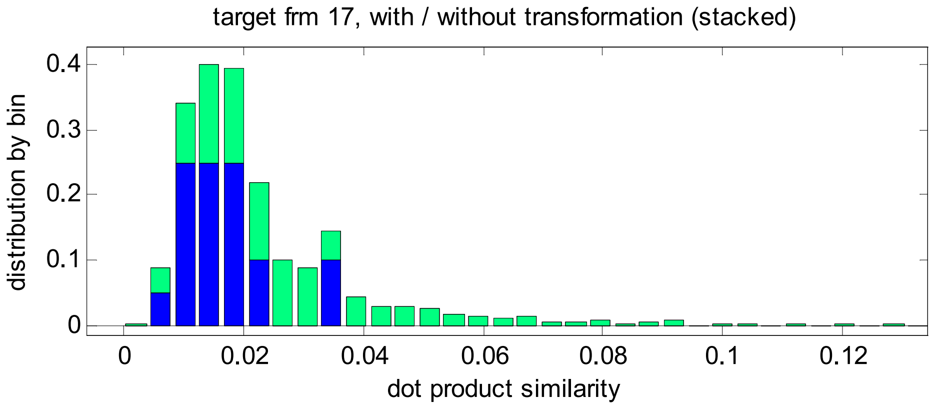

Figure 1.1: Distribution of similarity (dot product of chroma vectors) of small source database with target frame, without (blue) and with (green) transformation (transposition).

Of course, there is previous work applying transpositions in mosaicing synthesis [Simon et al., 2005], but the principle is general; transformations multiply the potential coverage of source units in descriptor space. Not necessarily to arbitrary points, that depends on each of the specific transformations, descriptors, and source units; but predictive models of transformations, not part of the previous work using transpositions, can be used to see which target points are feasible as well as predict side-effects to related descriptors.

To characterize how transformations affect the source units, strategies for modeling them are proposed (Chapter 3). For example, in the proof-of-concept study described in Chapter 4, a spectral model of two effects used together with many parameters is developed, and parameters are automatically chosen using the model. Chapter 5 is an early attempt to learn these models automatically.

Even if a large database is to be used, the typical distribution of audio in descriptor space is highly non-uniform. Haro Berois [2013, Chapter 3, Rank-frequency distribution of audio timbral descriptors] shows: in terms of codewords which are basis vectors of timbre representation (Bark and

MFCC coefficients), some codewords are much more common and the rest are progressively rarer in distribution in reasonably large databases.

That concerns individual components, but many features such as chords and timbres are judged in combination. One would guess that certain phenomena: e.g. certain synthetic chords, microtonal clusters, or instrumental timbre combinations, are only rarely encountered and with few close neighbors, no matter how large the database is. When the target is a combination of different aspects (harmony, timbre, spatial characteristics, etc...), finding a match in the source database should be vanishingly small.

Where the source material is sparse in the descriptor space, using transformations to add new neighbors around the existing sources is one tool for creating closer matches that do not exist in the original source database. Another tool is to allow the combination of original or transformed sources.

1.4 The Role of Mixtures

As mentioned in the last section, the source database might not have a full gamut of combination features, e.g. chords, that the user might like to synthesize/imitate. A natural strategy is to allow the combination of simpler features that may be present or more common in the sources.

To put this in somewhat abstract terms, imagine the goal is to synthesize an approximation to the sequence of descriptor vectors: $(y_1, ..., y_t)$ using the set of s available sources (also given in descriptor space): $\{\vec{a}_1, ..., \vec{a}_s\}$.

The approach taken by most previous work uses the path minimization approach, meaning to solve an optimization problem fitting of this basic form (although the distance function is a simplification):

$$\arg\min_{s[t],w_t} \underbrace{\sum_{t=1}^{T} \|y_t - w_t \vec{a}_{s[t]}\|^2}_{\text{distance to target}} + \underbrace{f(s[1], ..., s[t])}_{\text{addl. penalties}}, \tag{1.1}$$

where $s[t], t = 1...T$ is a sequence of source frame indices chosen for each target frame, w_t is a gain chosen for the t^{th} target frame, and f is some function of $s[t], t = 1...T$. In this way, a single stand-in (a source frame with some gain applied) is chosen for each target frame and the function f decides which sequences of sources are better or worse.

Instead of choosing just a single weighted source frame for each target frame, an alternate approach is to allow each target frame to be approximated by a weighted sum of all source frames. In this case, we choose a sequence of weight vectors $\vec{w}_t, t = 1...T$, and solve the following optimization problem:

$$\arg\min_{\vec{w}_t} \sum_{t=1}^{T} \|y_t - A\vec{w}_t\|^2 + f(\vec{w}_1, ..., \vec{w}_T), \tag{1.2}$$

where A is a $d \times n$ matrix of source descriptor vectors, and f is similar to the penalty function in the previous problem.

It is interesting to note that although the optimal solutions of both problems will depend on the target and source descriptors, the distance to the target (the collective distance to the target frames given by the sum of squared norms) of Eq. 1.2 will always be less than or equal to that of Eq. 1.1. This is because the approximation of Eq. 1.1 is a special case of the approximation of Eq. 1.2, i.e. the case when each $\vec{w}_t$ is a 1-sparse vector (consisting of only one non-zero value).

The second approach presumes that there is a meaningful way to synthesize sounds that are in-between a group of sources in descriptor space. Additionally, although the approximations of 1.2 contain all sources as terms, there are practical reasons for using fewer terms and fewer simultaneous sources in synthesis. Both issues are dealt with in Chapter 6.

In Chapter 2, previous work using mixtures is discussed, and in Chapter 6, new mosaicing algorithms using mixtures are introduced, and evaluated in Chapter 7.

1.5 Research Questions

In this section, motivating research questions are stated along with brief answers referencing the text.

1.5.1 How can general-purpose control of sound transformations be achieved?

This is addressed by the model system presented in Chapter 4. The strategy introduced is to model in some way the transformations in descriptor space; and to use appropriate numerical optimization techniques to select the transformation parameters.

1.5.2 How can adding sound transformations affect sampling synthesis?

Adding transformations increases the potential similarity with target material, even when using a simple transformation such as transposition. This is demonstrated in Chapter 6.

1.5.3 What kinds of models of sound transformations are possible?

This is addressed in Chapters 3, 4, 5. It is argued that predictive models (introduced in Chapter 3) are more flexible in the context of synthesis systems.

1.5.4 Of what use are models in control of sound transformations?

This is discussed in Section 2.2.1 of State of the Art. Models avoid the computational effort of having to transform and analyze new sources to find out what their descriptors are, during the parameter search process. This should be critical in making interactive systems.

1.5.5 Can models of sound transformations in descriptor space be learned from data?

Yes, with the reservation that posing these models as general smooth regression models leads to statistically very hard problems that will not generalize well in prediction. This is explored in the experiment of Chapter 5.

1.5.6 How can mixtures and transformations be incorporated into mosaicing algorithms?

In Chapter 6, a mosaicing system is presented which incorporates mixtures and transformations along with standard criteria and criteria especially relevant, such as continuity criteria, to sampling synthesis.

1.5.7 Do hybrid mixture-mosaicing algorithms produce results of sufficient quality?

The algorithms in Chapter 6 are explored technically in Chapter 6, and evaluated subjectively in Chapter 7.

1.5.8 Which dependencies arise when integrating sampling synthesis with descriptor-controlled transformation?

Continuity of source sampling (as in mosaicing) and continuity of transformation are two important concerns when integrating transformation with sampling synthesis. In some cases, these concerns interact (are interdependent), such as the case of using resampling as a transformation in mosaicing, in which the transposition factor affects the speed of source signal evolution (this is embodied in the $\pi_{\Delta\mathrm{src}}$ penalty in Section 6.4.1, p. 101).

Due to the fact that continuous solutions in time are sought, along with the consideration of a number of additional criteria, it is argued that predictive models of transformations are more flexible for this purpose than selective models (as defined and argued in Ch. 3).

1.6 Applications

The main application of this thesis is the mosaicing algorithm of Chapter 6 that implements a kind of texture transfer, by the following process. First, the target signal is analyzed and turned into a descriptor sequence of harmony and timbre descriptors. Then, an optimization process determines a score that should produce an output similar to those target descriptors. Finally, the mosaic is rendered by transforming the source signals by the indicated parameters and mixed as indicated by the score. Figure 1.3 shows a detail view of atoms from an example score.

Although there is no interactive implementation of this algorithm yet, the algorithm is causal, so it should be possible in the future. Figure 1.2 is an early conceptual sketch showing how the source texture could be switched interactively in real-time.

Besides the texture transfer application, it should be possible to use the framework of transformation models to implement “transformation by example” systems, meaning effects units that tune their parameters according to some target or reference sound. It should also be possible to add mixtures into this framework.

1.7 Contributions

This section sums up the research contributions.

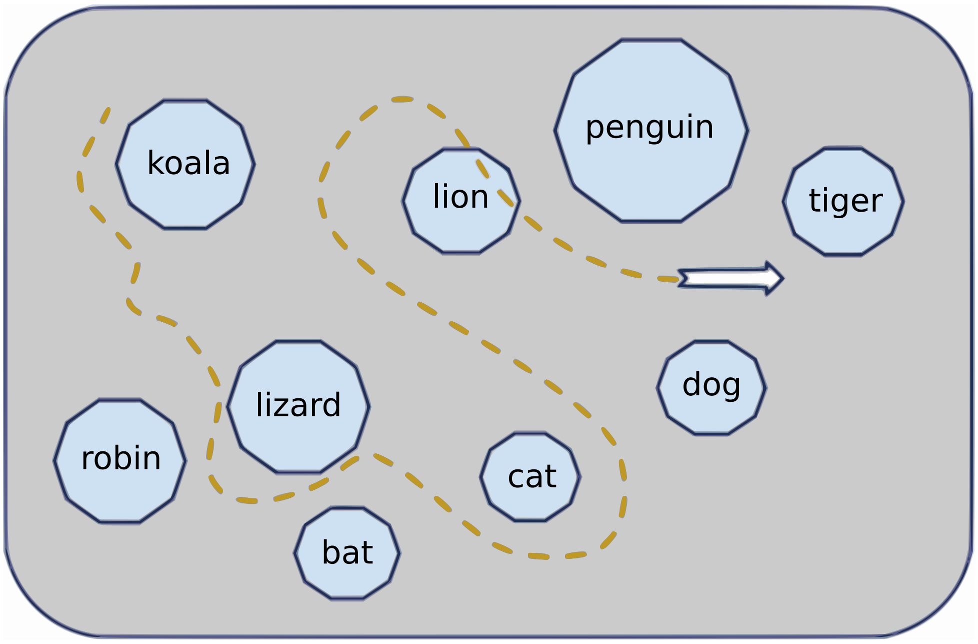

Figure 1.2: A conceptual sketch imagining interactive control of the sound sources used to render music by way of a pointing gesture.

1.7.1 A Case Study on Optimizing Analytic Models for Control of Transformations

In Chapter 4, analytic models of a cascade of two transformations are developed and optimized to find parameters given target descriptors.

1.7.2 A Design and Evaluation for Learning Predictive Models of Transformations

In Chapter 5, predictive models of a transformation are learned from data. They are evaluated by comparing their predictive performance with that of an analytic model. Additionally, failure modes of the statistical learning method's hyperparameters are illustrated.

1.7.3 Mixture Mosaicing Methods incorporating Sound Transformations

In Chapter 6, new hybrid mosaicing methods that promote both mixtures and continuity are introduced. These methods are causal, relatively efficient (compared to MP), and should be compatible with real-time implementation.

1.7.4 Design of Listening Tests for Mosaicing Methods

In Chapter 7, a series of listening tests, designed to evaluate audio texture transfer systems are presented. These tests include a test to identify the source of a sound, and comprise mainly a leading form of audio quality testing (MUSHRA) in order to distinguish quality factors resulting from different mosaicing algorithms.

1.7.5 Validation of Mixture Mosaicing for Musical Texture Transfer

In Chapter 7, the new mosaicing methods presented in Ch. 6 are evaluated along with classical mosaicing algorithms and signal oriented atomic decomposition methods (MP).

1.8 Dissertation Structure

This chapter introduced the motivation for the research and the main research questions, as well as mentioning some possible applications. Next, Chapter 2 outlines previous work regarding mosaicing synthesis, and synthesis and transformations controlled by target descriptors. Then, Chapter 3 gives some terminology and preliminaries concerning transformation models. Chapter 4 develops a model of a small set of transformations, and shows a proof-of-concept strategy for controlling these transformations by numerical optimization. Chapter 5 is an attempt to learn transformation models from data.

In Chapter 6, new mixture mosaicing methods are introduced, and evaluated in Chapter 7 with listening tests for the application of texture transfer. In Chapter 8, we summarize the results and discuss some future work.

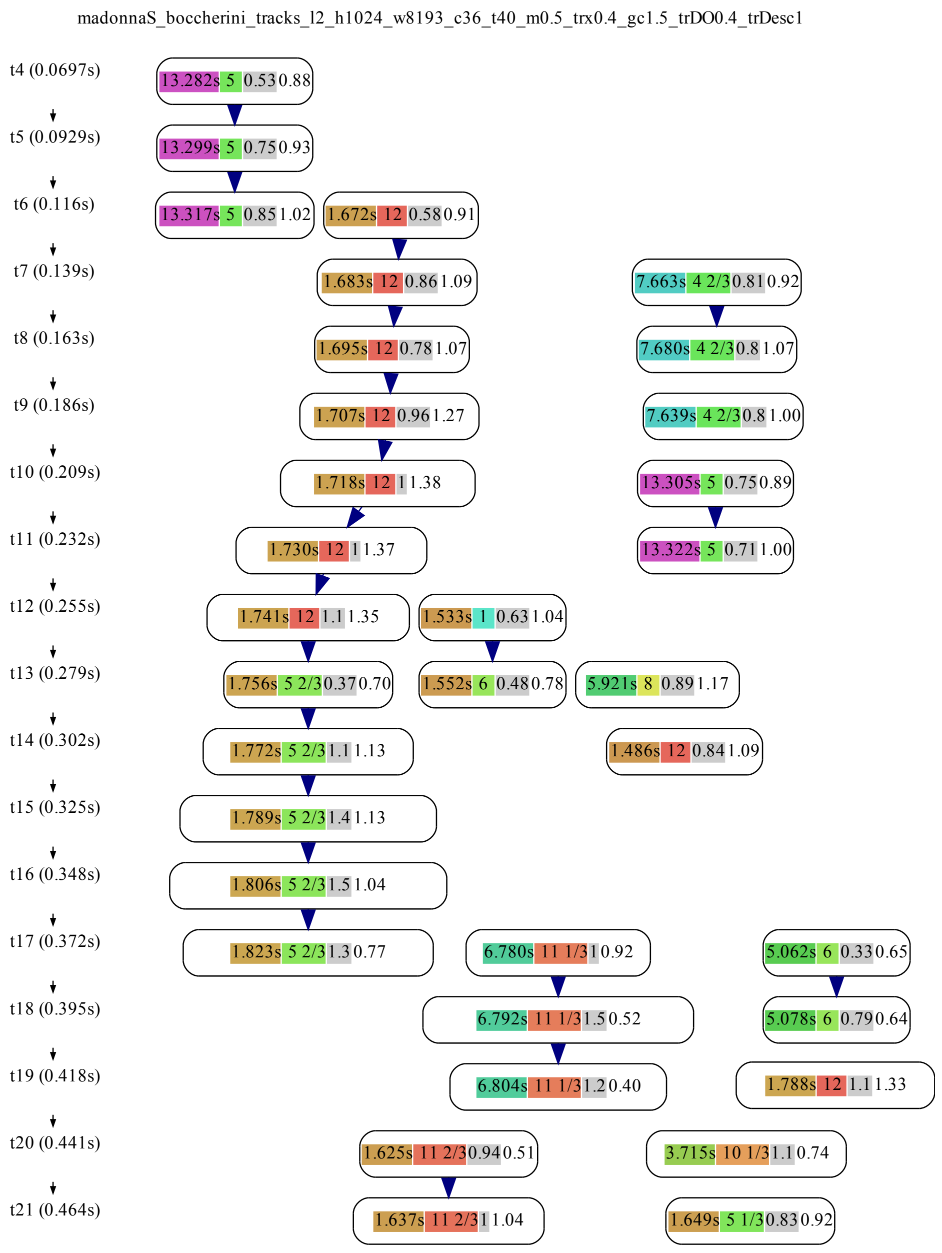

Figure 1.3: Roughly 400ms visualized from an example score. Bubbles depicting synthesis atoms detail the following fields: Source position in seconds, transposition in semitones, linear atom weight, and gain coefficient. The heavy arrows denote exact continuous links between atoms in subsequent frames. This example was generated at 44.1kHz sampling rate, 8193 sample window size and 1024 hopsize, 36 chroma bands and 40 mel-spaced bands with half weight to each, with a high cost on creating new tracks, gain, and change in transposition.

Chapter 2

State of the Art: Audio Mosaicing, Descriptor-controlled Synthesis and Transformation, and Sparse Decompositions of Audio

This chapter is a technical review of automatic audio mosaicing systems, and their capabilities, solution methods, and limitations.

However, mosaicing is only a particular instance in a class of systems, those that work towards targets specified in descriptor spaces, i.e. descriptor-controlled synthesis and transformation systems. Through this lens, mosaicing is simply sampling synthesis controlled by descriptors. Reviewing this larger class is useful because it opens up more insights and methods that are possibly applicable. Indeed, the proposed system described in Chapter 6, can be seen as an embedding of descriptor-controlled transformation within mosaicing.

Finally, we review the uses of and techniques for sparse decompositions of audio. This will shed light on what is unique, useful, and significant about our proposed methods.

2.1 Mosaicing synthesis

First, we focus on descriptor-controlled sampling synthesis, also known as audio mosaicing. It is referred to as such, because it creates audio mosaics:

sound compositions assembled from audio samples of potentially disparate sound sources to contribute an overall musical or sonic impression for the ears; in the same way that a physical mosaic is assembled from tiles of varying color and texture that contribute to form a larger image for the eyes.

Within sound synthesis, mosaicing belongs to a larger family of sampling synthesis. In sampling synthesis, sound is derived from existing samples rather than being generated numerically, for example, in the cases of physical modeling and abstract algorithm synthesis. Concerns in sampling synthesis include how to flexibly modify these samples (addressed by strategies such as spectral modeling), as well as how to select, and assemble in time, the samples from a database [Smith, 1991].

Mosaicing is thus a special case of sampling synthesis, in which an abstract target descriptor sequence determines which samples (from a wide variety of sound classes) are used and how they are modified and composed. This is in contrast to more specific types of sampling synthesis, for instance, sample-based singing synthesis, in which the descriptors might demand specific phonemes and notes.

In this review, the following definitions are used:

descriptors Data that describe the content of the audio samples, allowing for a measure of similarity between the target and source units.

source The audio samples to be assembled in synthesis.

target The synthesis goal, i.e. a time-sequence of descriptors.

units The smallest undivided audio samples to be selected, transformed, and assembled. These can either be of uniform or non-uniform duration.

In this review, the main comparison is limited to systems that deal mainly with sound data, and that synthesize the output with sampling synthesis, guided by time-sequences of descriptors representing a target signal; although other work is also mentioned in passing.

Perhaps the first audio mosaicing system driven by descriptors was the CATERPILLAR system [Schwarz, 2000]. In this system, target units, derived either from a score or analyzed audio samples, are matched up with source units. Both the target and source units are of non-uniform length, informed by rhythmic, onset, and envelope analysis of the input signals.

The guide for this matching process is a cost function, in which the total cost is divided up into two parts: first, the target costs, based on similarity between the matched target and source units; second, the concatenation costs, based on finding good sequences of concatenated source frames, e.g. that continue the sampling context, if possible.

The total cost can be conceived as the cost of a path over time, through a graph of source units. If the costs are limited to the two types mentioned above, target costs and concatenation costs, the optimal path can be found using the Viterbi path decoding algorithm [Viterbi, 1967]. This algorithm uses space and time $O(NT)$, where N is the number of source units and T is the length of the sequence of target units, rather than in time $O(N^T)$, the number of possible paths. In practice, this is still quite costly, so there is usually some pruning of the best local matches at each time step.

This approach, including much of the terminology, are inspired by sampling approaches to speech synthesis, i.e. Concatenative Speech Synthesis (CSS). The method of using the Viterbi algorithm to find the best sequence of samples seems to have originated with [Hunt and Black, 1996].

Further development of the system [Schwarz, 2003, 2004] brought many enhancements, including a wide variety of descriptors and specific modifications for synthesis of artistic speech [Beller et al., 2005].

In the MUSAICING system [Zils and Pachet, 2001], additional cost functions are proposed that regulate the overall representation of source sounds in the final mosaic, referred to as cardinality constraints. For example, two examples given are the "all-different" constraint, i.e. all samples used should be different, and that "80% of sounds should be percussive". In general, cost functions that have this global scope, as opposed to the strictly local costs of the target and concatenation costs, are incompatible with Viterbi path search.

In order to find approximately good mosaic solutions, the cost functions are formulated into a soft Constraint Satisfaction Problem (CSP), and heuristic search method known as Adaptive Search [Codognet and Diaz, 2001] is applied. The authors claim this gives good results on large databases. This method was also later adopted in CATERPILLAR [Schwarz, 2003].

Not all systems need to use complicated sets of cost functions and intricate search methods to get interesting results; simple matching based on nearest neighbors may be sufficient. Soundmosaic [Hazel, 2001] and MATConcat [Sturm, 2006] are two examples of systems that work based on matching uniform duration units based on similarity to the target units alone.

Several proposals have been made to improve the relevance and descriptor information associated with the source units. In a framework for "event synchronous music analysis-synthesis" (referred to here as ESMAS) [Jehan, 2004] units at the onset and beat levels are provided. The units are segmented with a technique based on the auditory spectrogram, which is meant to extract perceptually relevant units from polyphonic source audio. Similar in character is the LoopMash system [Bachmann et al., 2012, p. 161], which uses target replacement of beat units, one of the few interactive real-time systems that target-based mosaicing.

In the Audio Analogies system [Simon et al., 2005], pitch information from MIDI scores is used to segment a monophonic audio source. Then, a new score can be synthesized with transpositions of the original note units, using with an optimal path approach. The system of [Dannenberg, 2006] (referred to as CSUSAT) uses a similar premise, except it uses polyphonic source units aligned to a score, segmented into maximally long sets of concurrent pitches.

In the Guidage system [Cont et al., 2007], more intended for audio retrieval than for resynthesis, a new descriptor over entire audio sequences called the Audio Oracle is proposed. This descriptor is a Markov model (a graph) predicting how the descriptors evolve and repeat in time. When searching for similar audio sequences, the system also builds a reconstruction of the target as a side-effect of the search. This system seems to be unique in considering the dynamics of the source signals, as represented by a finite state machine; how they might repeat in cyclic patterns or terminate. Eargram mentioned in Section 2.1.5 uses a similar strategy.

To the present date, few systems have extended mosaicing beyond the concatenative dimension (sequences of units) to the mixture dimension, in which several sounds are mixed to create a better approximation of the target. The Bayesian Spectral Matching (BSM) system [Hoffman et al., 2009] was the first mosaicing system to synthesize simultaneous mixtures of time-overlapping segments to get closer matches. In this system the units, fixed-length subsequences of source frames, are mixed with different gains to create a spectral approximation to the target. The model for this system is a temporal Bayes net in which the relative strengths of each shifted source are treated as latent or hidden variables. The posterior distribution over sources and shifts is estimated using Gibbs sampling and the highest probability assignment to the hidden variables (maximum a posteriori, MAP) is sonified to produce the mosaic.

Another mosaicing system supporting mixtures, an early version of the proposed system described in Chapter 6 (here referred to as AugMos), was documented [Coleman et al., 2010]. Like BSM, it supports approximation by a weighted mixture of fixed-length source segments; single frames, in the case of AugMos. It uses a different sparse projection technology, basis pursuit (BP) [Chen et al., 2001]. In addition, it supports tonal transpositions of the source units, facilitated by models that predict the descriptors of a given transposition. The approximation itself is carried out in the reduced dimensional perceptual space consisting of filter banks representing harmony (PCP, or chroma features), and mel-spaced filter banks (described in Section 6.3, p. 92).

The sparse projection approach used in AugMos is described in more detail in Sections 6.7 and 6.7.1, starting page 118.

2.1.1 Unit Transformations in Mosaicing Systems

Let's take a closer look at transformations enabled in Mosaicing and Concatenative Synthesis systems thus far.

First we examine Caterpillar [Schwarz, 2004, p. 155]:

> The transformations applied in Caterpillar are only the adaptation of the mean loudness of the selected units to the target units, and the shortening of selected units that are longer than the target units.

Two reasons are given for limiting the transformations:

> The reason for limiting the transformations to loudness change and shortening is that these two are the only ones that do not degrade the sound quality. Moreover, they do not necessitate a sound representation other than the sampled signal.

In the Musaicing system, no transformations of the units are mentioned [Zils and Pachet, 2001]; likewise with the ESMAS system [Jehan, 2004]; and the Guidage system [Cont et al., 2007].

In MATconcat two transformations of source units are supported, that of reversing the source segments, and that of convolving the selected segments with the target segments. In Soundmosaic, the volumes of the segments are scaled [Hazel, 2001].

In the Audio Analogies system, a combination of resampling (to change pitch) and synchronized overlap-and-add (SOLA) is used to match audio segments to the midi pitch and length of the target [Simon et al., 2005]. By contrast, CSUSAT did not use pitch transpositions [Dannenberg, 2006]. This is perhaps because SOLA might not be appropriate for the polyphonic source units in that system. If CSUSAT were to alternately use resampling, as in the proposed system of Chapter 6, the system would have to account for the change in length of the tonally transposed synthesis units.

In the BSM system, mosaics are produced by applying gains derived from the MAP-estimated mixtures of the input frames, no other transformations are applied to the source units [Hoffman et al., 2009, Section 3.2, "Sonifying the MAP estimate"].

In the AugMos system, two transformations, tonal transposition (implemented by bandlimited resampling) and filtering, were supported [Coleman et al., 2010]. The model predicted transpositions are first selected from the dictionary by the sparse projection algorithm. Then, after the sparse representation is chosen, the filter parameters are chosen based on the synthesis model, based on quadratic smoothing of the filter gain parameters in time-frequency.

In the LoopMash system, many effects can be applied manually, i.e. by toggling selected units occurring on specific beats: such as reverse, staccato, scratch, stutter, and others [Bachmann et al., 2012, p. 167, Applying Slice Modifiers and Slice Effects]. There are also some target driven effects available, such that the volume, temporal envelope, spectral envelope, and the sample length (through time-scaling) of the units can modified to be closer to the target units [Bachmann et al., 2012, p. 171, Audio Parameters].

2.1.2 Unit Transformations in Other Concatenative Synthesis Systems

Outside of mosaicing systems, other concatenative synthesis systems such as speech synthesizers also use unit transformations enabled by signal processing. Three such examples are: achieving target prosody through pitch modification and spectral envelope modifications based on the source-filter decomposition [Valbret et al., 1992], as well as spectral smoothing (filtering) designed to smooth spectral discontinuities at concatenative unit boundaries [Plumpe et al., 1998].

In a singing voice synthesis system [Bonada and Serra, 2007], both time-varying filters and pitch changes are applied as transformations to the units, using a technique referred to as Voice Pulse Modeling (VPM). These transformations model the following aspects of the target and source units: pitch and prosody, evolution of speech formants, vibrato, and other expressive characteristics.

2.1.3 Summary of Target-based Sound Mosaicing

Continuity constraints, which maximize the length of original subsequences from the source material, and minimize the number of necessary new concatenations, have been used in several systems: Caterpillar, Musaicing, and Audio Analogies. One system, Guidage, takes this even further, by detecting which unit transitions are common within the source material itself.

This type of constraint is more generally a transition cost. Although the Audio Analogies and LoopMash systems include non-trivial target-based transformations, they do not include transition costs on the evolution of the transformation in time, even though this is possible in a path minimization framework.

Perhaps because those systems use note and beat length units, it is not necessary to consider the transformation beyond the length of the unit. Still, it may be useful in systems which have shorter, more granular units, or that wish to have continuous evolution of transformation parameters, to also consider transition costs concerning their evolution in time. Such a transition penalty is used in the proposed system, given in Equation 6.20 (p. 103).

Advantages of the path minimization approach are clear: the ability to find variable-length subsequences of original source material, and the minimization of general continuity between sequenced units. On the other hand, the sparse mixture approach allows closer spectral approximations of the target material.

However, the two approaches are difficult to reconcile, as there is no existing algorithm known to the author that combines transition costs within a framework supporting sparse mixtures. That is, neither the formulation of path minimization (e.g. in Caterpillar) nor the formulation of a constraint satisfaction problem (in Musaicing) are able to consider a search domain including spectral mixtures. Furthermore, the sparse projection method employed in BSM only considers fixed-length subsequences from the source database (in AugMos just single frames). Neither considers general continuity, as expressed by transition costs, of variable-length unit sequences.

The proposed system in Chapter 6 is an attempt to combine the two frameworks of path minimization and sparse mixtures.

2.1.4 Cross-synthesis based on Atomic Decompositions and Non-negative Matrix Factorization (NMF)

In this section, some relatively recent work in cross-synthesis techniques based in atomic and NMF decompositions is discussed. These approaches are relatively "granular", as they are built by rearranging or modifying units representing frames of the input.

Matching pursuit (MP) [Mallat and Zhang, 1993] (further explained in Section 6.5.1, p. 104) is a sparse approximation method that is often used for time-domain decompositions of audio. In [Collins and Sturm, 2011], two cross-synthesis processes based on decomposing the source and target with a Gabor dictionary were proposed. But rather than resynthesizing the target with source units directly, the atom weights of the target are modulated or mixed with the source weights. These modifications can operate differentially according to temporal or spectral characteristics of the Gabor atoms, in a way similar to the "molecular" transformations of [Sturm et al., 2008a; Sturm, 2009].

Alternately, in another method proposed [Collins, 2012, Section 8.1, Sparse Concatenation], a dictionary consisting of short segments of the source signal was used to reconstruct the target signal with MP. This method is essentially the same as the *mp* method (as described in Section 7.3.2, p. 130), only without the tonal transpositions used for *mp*. Chapter 7 gives a subjective evaluation comparing the quality characteristics of mosaics generated from *mp* and other methods.

Non-negative Matrix Factorization (NMF) [Lee and Seung, 1999] is a numerical technique that can be used to separate non-negative signals into parts. It is widely used for audio source separation [Smaragdis et al., 2014]. In the audio setting, it is most common to factor the spectrogram matrix (consisting of each frame of the power spectrum in time) into two matrices. One matrix contains a set of single-frame spectral components, and the other matrix contains a set of sparse gains for the activation of each spectral component in time.

In the FactorSynth system [Burred, 2014], cross-synthesis is implemented by taking the target spectral components, and substituting them with the source spectral components, where these two sets are matched according to a timbre similarity measure based on MFCC descriptors. To resynthesize the output signal from the spectrogram, which requires estimating the phases of the output mixtures of power spectral components, a new scheme based on Wiener filtering was proposed.

The HarmonyMixer system [Fukayama and Goto, 2014] has the goal of taking source and target signals, and imposing the relative target chord qualities on the source, rather than reconstructing the target using the source units (using our terminology, in which the source units are sampled). In this system, a similar decomposition and matching process occurs, except the NMF decomposition is performed on a transposed form of the chromagram matrix, and chroma components representing chords are swapped or interpolated.

The modified chromagram is resynthesized by first estimating two sequences of transposed source frames with a dynamic programming procedure. The first sequence accounts for positive changes to the original chromagram, and is synthesized by adding the transposed source frames back to the original source frames. The second sequence accounts for negative changes, and is synthesized by linear filtering of the original source frames.

One recent and promising work using NMF for target signal imitation is the LetItBee system [Driedger et al., 2015]. In this system, the target spectrogram is factored using the source spectrogram using NMF, resulting in a sparse activation matrix allowing resynthesis of the target from weighted combinations of source frames, as in BSM and AugMos.

However, the system encourages several desirable solution qualities by applying a sequence of functions, linear and nonlinear, to the activation matrix before a final NMF update. These qualities are: that repeated source frames are avoided, that only few sources activated at once (sparsity at a time instant), and that longer sequences of time-activations are preferred. These same qualities are also promoted in the proposed system, although with a different mechanism (based on penalty functions), and in a more complicated setting allowing transformation of source units.

2.1.5 Interactive Alternatives to Fixed Target Mosaicing

Several systems have proposed to bring something similar to mosaicing into an interactive and real-time context. Rather than reconstructing a fixed target signal, they provide ways to interact, arrange, and perform with the source units in a database, usually by navigating in a descriptor space.

In Mosievius [Lazier and Cook, 2003], an alternate model of the "sound sieve" is developed, based on user-defined ranges in descriptor space. These regions are used to segment units, to include or reject them from presentation, even to transform units lying in one part of the feature space to another.

In the CataRT system [Schwarz et al., 2006], a database of analyzed and segmented sound units is represented as an interactive scatter plot. In this plot, multidimensional scaling is used to project the high-dimensional descriptors into a lower dimensional projection that organizes the sound units by similarity. The mouse is used to point to nearby units in descriptor space, and different triggering modalities are available for performance.

The Mused system [Coleman, 2007], designed for selecting sound units for sample-based sound and music composition, also uses an interactive scatter plot to navigate a database of segmented units. To aid the selection of specific sound units, the user can select ranges of the descriptor space (similar to the "sound sieve" concept from Mosievius) and the scatter plot rapidly updates as units are included or excluded. This technique of giving immediate visual feedback of a query result, "dynamic queries" [Ahlberg et al., 1992], is previous work in the field of information visualization.

The recent Memory Mosaic [Mital, 2015] is a mobile sampling instrument that uses automatic segmentation and an interactive scatter plot, similar to CataRT and Mused.

The Eargram system [Bernardes et al., 2013] intended as an improvisational and compositional tool, is based on multiple views of the database that include an interactive scatter plot. Eargram provides several interesting real-time triggering strategies, including one modeling the dynamics of sources in descriptor space (akin to the representation of Guidage).

Another branch of systems, e.g. SoundSpotter [Casey and Grierson, 2007], focuses on triggering similar sounds from a database in response to live input. Rather than trying to transform and arrange the source sounds to be as perceptually similar to the target sound as possible, the focus is on creating a compositional feedback loop between the performer and the system [Casey, 2011].

The prototype system of [Janer et al., 2009] is a beat-based loop system (similar to LoopMash). But instead of target units that are matched by similarity, a symbolic score is used, in which the user places marks in a step sequencer. The marks represent different sound categories (percussive and noisy) which are triggered. Sound units belonging to each category are identified by automatic sound classification.

None of the systems cited in this section use advance planning by minimizing transition costs (in contrast to some of the systems of the main Section 2.1, i.e. Caterpillar, Musaicing, or Audio Analogies). This is likely due to its high computational cost. Instead, more immediate selection methods are used. [Schwarz et al., 2006] explains:

> Because of the real-time orientation of CataRT, we cannot use the globally optimal path-search style unit selection based on a Viterbi algorithm as in Caterpillar, neither do we consider concatenation quality, for the moment. Instead, the selection is based on finding the units closest to the current position x in the descriptor space, in a geometric sense...

But it is still possible to use some representation of dynamics. For example, it is computationally simple to employ triggering based on Markov dynamics, as in Eargram. The key is that there are no target costs in this setting to balance with the transition costs, which typically requires dynamic programming.

Dynamic programming can be adapted to some limited real-time settings. In the work of [Costello et al., 2013], a phase vocoder is used to time-warp a live source input so that it is temporally more similar to a target sound file. This certainly seems in the spirit of mosaicing, even if it doesn't allow interaction with a multitude of different sounds at once.

Another real-time system that incorporates dynamics in various forms is Improvasher [Davies et al., 2014]. Beat tracking is used to trigger beat units with the audio input, and the chord sequence history (represented with chroma vectors) is used to predict which source units will harmonically match the input.

2.1.6 Mosaicing with Other Data besides just Sound

Other systems present novel ways to describe audio content with respect to synthesis and resynthesis. The Sound-by-Numbers system [Cardle et al., 2003] uses linked motion paths to describe the audio and uses a target motion path to generate a new soundtrack using the correspondence between motions in the target and source paths.

Sven König's Scrambled Hackz system [Van Buskirk, 2006] is another mosaicing system that uses target audio to assemble beat units from source clips consisting of musical video.

2.2 Descriptor-controlled Synthesis and Transformation

Next, we look at systems for descriptor-based control of other synthesis methods and transformations. This will provide additional inspiration about design decisions involved in selecting transformation parameters for units.

In general, these systems can be divided into two categories: those that iteratively synthesize and analyze the result in descriptor space, and those that use models of the sounds in descriptor space to obviate the need for iterative synthesis.

2.2.1 Synthesis controlled by Descriptors

Additive synthesis is one class of synthesizers for which control by descriptors has been quite well explored. In the system of [Wessel et al., 1998], referred to as Time-Axis, two input/target descriptors, fundamental frequency (F0) and loudness, are used to control the amplitudes and frequency parameters for an additive synthesizer based on data of real instrumental timbres.

That sampling synthesis preserves the continuity of phrases is cited as an inflexibility, which is an impetus for their system:

> Analysis-synthesis methods have for the most part privileged time warping and pitch shifting. Musical signals analyzed by such methods as the phase vocoder and sinusoidal modeling allow composers to stretch and shrink the time axis independent

> of the pitch and to alter the pitch without altering the duration. These time and pitch modifications have been put to practical and creative use, but the fact that the time-stretched sounds preserve the order of the evolution of the sound's spectral features greatly constrains the nature of the potential transformations. The data from such analysis methods does not afford the construction of new phrases, this is to say, new sequences of pitches and amplitudes.

The title of that paper, "Removing the Time Axis from Spectral Model Analysis-Based Additive Synthesis" thus refers to removing the time-dependency inherent in sampling synthesis.

The control is accomplished by using two different machine learning methods, one parametric and the other non-parametric, in order to perform the mapping between descriptors and synthesizer parameters. The system of [Jehan, 2001; Jehan and Schoner, 2001], referred to here as Audio-Driven, follows the same pattern but adds innovations such as an additional control descriptor, brightness; as well as adding a synthesizer component for the residual noise spectrum (as in spectral modeling synthesis). Another system, Perceptsynth [Le Groux and Verschure, 2008], is essentially similar to Time-Axis; although, it simplifies the machine learning problem by first reducing the additive synthesis parameters using principal components analysis (PCA). Ssynth [Verfaille et al., 2006a] is yet another example of an additive synthesizer controlled by interpolating data from additive analysis of instrument recordings.

Those systems used analysis data from specific instrument recordings to learn models that map high-level controls to synthesis parameters. But it is also possible to control synthesis using models that go in the other direction: that take as input synthesis parameters, and that output a prediction in descriptor space of the output. With this second type of model, it is possible to abstract away from any specific instrument timbre. These distinct model classes will be further discussed in Chapters 4 and 5.

One such system is the Timbral Synthesizer [Mintz, 2007]. It uses a predictive model of MPEG-7 instrumental descriptors (covering both spectral and temporal envelope characteristics) to drive an additive synthesizer. Instead of being learned from data, the model is derived from spectral theorems. The target descriptors are converted into linear "synthesis equations" (satisfying constraints for target descriptors) comprising a linear program (LP), a type of convex optimization problem.

Rather than using relatively generic high-level descriptors to control to an additive synthesizer, the SINGTERFACE system [Janer, 2008] has as its goal using the voice as synthesizer controller. To this end, methods were developed that track the voice formants, segment the input into notes, and map the result to a singing voice concatenative synthesizer. The mapping from descriptors to concatenative units, which additionally aligns the input signal to the song text, is done with dynamic programming.

2.2.1.1 Systems using Iterative Synthesis

The VOCALISTENER system [Nakano and Goto, 2009] has a similar goal to that of the previous system: to adapt a singing synthesizer score (including timing, pitch, and dynamics parameters) to be sufficiently similar to a recorded performance. This is accomplished in roughly the following way: the initial score is synthesized, and aligned with the target signal. Then, the score parameters are iteratively updated, and the process is repeated until the score is suffiently close to the target.

Like VOCALISTENER, and in contrast to the previously mentioned model-based systems, this class of systems instead use an iterative synthesis loop, meaning the following: first, some starting parameters are synthesized. Then, the generated audio is analyzed, and new parameters can be proposed, repeating until stopping conditions are met. In this strategy, a model linking the synthesis parameters with the output/target descriptors is not necessarily required, and potentially any synthesis technique may be used. Perhaps that is why this strategy is also used by the following two systems.

FEATSYNTH [Hoffman and Cook, 2007] is a framework that uses genetic search to find parameters that match a target descriptor sequence. For example, an accompanying tutorial [Hoffman, 2007], mentions this example: the mixture of two sinusoidal generators with a white noise generator, all being controlled by target descriptors consisting of spectral centroid and spectral rolloff.

SYNTHBOT [Yee-King and Roth, 2008] is another system that also uses genetic algorithm search in an iterative synthesis loop to tune parameters of a completely generic, black box synthesizer; in this case, of VST synthesizers to match a target sequence of mel-frequency cepstral coefficients (MFCCs).

However, in some cases when the synthesis methods (or transformation methods) are known in advance, this extra computational effort in parameter search can be avoided by using models.

2.2.2 Transformation controlled by Descriptors

A common and general strategy related to descriptor-controlled transformations is Analysis-Synthesis. In this strategy, there is an analysis method that finds parameters that exactly or almost exactly reproduce an input sound with a given synthesis method. Those parameters can then be interpolated or modified with parameters corresponding to target descriptors so that the output more closely resembles the target.

Many classic transformation techniques use this approach, such as pitch and time stretch modifications in the phase vocoder [Flanagan et al., 1965; Dolson, 1986], harmonic models such as SMS [Serra, 1989], and other source-filter based modifications [Verfaille and Depalle, 2004]. The parameters available in these synthesis methods are frequently low-level (short-time spectra, harmonic additive synthesis, and filters) and do not map directly to higher level perceptual descriptors of interest; although the Analysis-Synthesis strategy has also been applied to the setting of singing voice synthesis in the SINGTERFACE system.

In the Feature Modulation Synthesis (FMS) approach, target descriptors of interest (temporal, spectral, or harmonic) are paired with transformations that attempt to modify them directly with minimal side-effects to other descriptors [Park et al., 2007, 2008; Park and Li, 2009]. This promising paradigm could be paired with sampling synthesis, by applying cross synthesis to the sources using target features.

However, this is not sufficiently general to work for all sampling applications. For example, in the case of mixture synthesis, where rather than applying the same target descriptors to each of the mixture sources, one might prefer to apply different transformation parameters to each of the sources, such that they produce a mixture that is close to the target descriptors (the approach taken in this thesis). This scenario is more general (based on optimization of parameters) and possibly allows higher quality solutions, given the continuity criteria of the transformations.

Whereas the input of a synthesis model consists of target descriptors only, the input of a transformation model must also take into account the descriptors of the input sound, leading to a higher dimensional function. In the case of learned models, this may lead to a harder estimation problem (see Appendix C.1).

Barring an easy relationship between transformation parameters and descriptors, an alternative strategy consists of formulating a model that predicts how the transformed sound's descriptors change with differing parameters, akin to the predictive model of Timbral (Section 2.2.1).

The system of [Coleman and Bonada, 2008] (also documented in Chapter 4) used essentially that approach. The goal of the system was to control a set of two transformations, resampling and equalization, using standard moments of the power spectral distribution, (mean, standard deviation, skew, kurtosis, etc) as target descriptors. First, based on Fourier theorems [Smith, 2007], a model was derived that predicted the change in descriptors of interest. Then, a gradient-based heuristic search was used to find optimal transformation parameters for each descriptor target.

In a related work of [Caetano and Rodet, 2009], the goal was to find parameters for a filter that was intermediate in a perceptual space, for the purposes of sound morphing. The descriptors were essentially the same as the previous study, and genetic algorithms were used to find the filter parameters. This approach can also used to choose mixing parameters, in automatic mixing applications, such as one that chooses gains according to desired monitor levels using local search [Terrell and Reiss, 2009].

Certain types of transformations might be easier to optimize. One such class consists of transformations that are linear in their inputs, leading to the optimization problems being convex. The work of [Olivero et al., 2013] examines linear transformations that are literally multipliers to the inputs, such as a mask applied to a Gabor representation of the sound in order to find smooth interpolations in timbre space. The optimization problem is solved efficiently and leads to a unique optimum, as the problem is convex.

Spectral theorems might not be available for every possible sound transformation, in which case being able to learn changes in descriptors of transformed sounds from data would be useful. As far as learning general predictive models of the evolution of descriptors under sound transformations, there has been little work so far. One such attempt [Coleman and Villavicencio, 2010], to learn changes in a filter bank descriptors under resampling, is further documented in Chapter 5.

2.2.3 Summary of Descriptor-controlled Synthesis and Transformation

On one hand, there seem to be clear examples of synthesis systems that can be controlled by high-level descriptor controls, including examples of systems that learn from real instrumental data. On the other hand, there seem to be fewer systems for controlling sound transformations with descriptor controls, and even fewer that learn models of how transformations change a sound in descriptor space. Perhaps this discrepancy is due to a difference in the problem difficulty between synthesis and transformation.

2.3 Additional Applications of Sparse Decompositions of Audio

Finally, as an important topic in this dissertation, we would like to highlight some interesting uses of sparse decompositions of audio. For example, as shown by [Sturm et al., 2008a; Sturm, 2009], atomic decompositions have structure in time and frequency that can be used for sound transformations. In addition, this structure can also be used for denoising [Siedenburg and Dörfler, 2013], separating harmonic from transients [Siedenburg and Dörfler, 2011], filling in corrupted or clipped samples in an application known as "audio inpainting" [Adler et al., 2012; Siedenburg et al., 2014], detecting harmonic notes [Gribonval and Bacry, 2003]; not to mention audio source separation [Smaragdis et al., 2014], sound localization [Gretsistas and Plumbley, 2010], and identifying birds when there are several of them singing at the same time [Stowell et al., 2013].

Chapter 3

Design of Transformation Models for Sampling Synthesis

This chapter introduces some terminology for discussing models of transformations in the abstract, with a view to applying them in sample-based synthesis applications controlled by target descriptors. Whether the models are derived from knowledge of the transformation, or learned from examples consisting of inputs, outputs, and parameters, there seem to be several distinct approaches possible.

Another purpose for this chapter is to justify the choice of focus, regarding this thesis and regarding the synthesizer of Ch. 6, on one type of model, the predictive model, rather than another type of model, the selective model, based on the main problems faced in sampling synthesis.

3.1 Preliminaries

Whether one wishes to choose parameters for a simple transformation, or a network composed of many transformations with their own parameters, each transformation itself can be viewed as a black box that transforms input audio samples into output audio samples, with the result varying according to which parameters are used.

However, at the modeling level, it is not necessary to use audio samples; rather, a lower-dimensional descriptor representation is used. This allows the user to express a synthesis goal, the target, in some perceptually relevant space. Along with the target, the input and output samples are also represented in descriptor spaces.

For the target descriptors, they should reflect the qualities of interest relating to the transformation or synthesis goal. The output vector should contain at least the same set of descriptors in the target vector, since a main goal is matching the target vector as well as possible according to some distance function.

The input vector, in turn, should be a set of descriptors that predicts, sufficiently well, the output descriptors relevant to the target. This might be a bigger set than the target descriptors, because some signal information that is not relevant to the perceptual target might still help in better predicting the output descriptors.

It is assumed that we have extractors that turn samples into descriptors.

In total, this gives three descriptor vectors: the target $\vec{d}_\mathrm{T}$, input $\vec{d}_\mathrm{in}$, and output $\vec{d}_\mathrm{tr}$ vectors, and a parameter vector $\vec{p}_\mathrm{tr}$ containing the numerical controls of the transformation.

3.2 No Model

The black box transformation, denoted as $f_\blacksquare$, has the following form:

$$f_\blacksquare(\vec{d}_\mathrm{in}, \vec{p}_\mathrm{tr}) = \vec{d}_\mathrm{tr}, \tag{3.1}$$

or in the case of synthesis, with no corresponding input signal,

$$f_\blacksquare(\vec{p}_\mathrm{syn}) = \vec{d}_\mathrm{syn}. \tag{3.2}$$

That is, by observing black box transformations, one can record tuples (triplets) of the form $(\vec{d}_\mathrm{in}, \vec{p}_\mathrm{tr}, \vec{d}_\mathrm{tr})$. With executable access, one can also produce new transformations by free choice of input sound (which determines $\vec{d}_\mathrm{in}$), and parameter vector $\vec{p}_\mathrm{tr}$.

Although the functional description of a black box transformation is identical to the simplest form of the following model type, a pragmatic distinction can still be made between systems that search for feasible or optimal parameters by repeated application of a black box synthesizer or transformation (referred to in Section 2.2.1.1, p. 28 as Iterative Synthesis), and systems that use models to avoid the additional computational effort that repeated application in search entails.

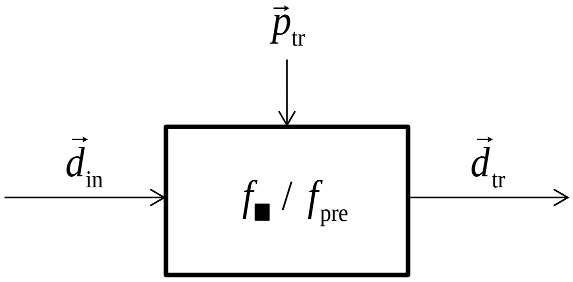

Figure 3.1: Graphical view of a black box transformation, or a predictive model.

3.3 Predictive Models

These forms of Eqs. 3.1 and 3.2 also correspond directly to the first type of model, predictive models. The simplest type of predictive model, f_{pre}, has the form:

$$f_{\text{pre}}(\vec{d}_{\text{in}}, \vec{p}_{\text{tr}}) \approx \vec{d}_{\text{tr}}, \qquad f_{\text{pre}}(\vec{p}_{\text{syn}}) \approx \vec{d}_{\text{syn}}. \tag{3.3}$$

That is, predictive models are those that, through either analysis or machine learning, predict the output descriptors, without respect to the target signal. The triplets of Eq. 3.1 can be used to learn models from data using the supervised machine learning paradigm.

An active learning paradigm (in which the learning algorithm chooses the input patterns using some clever criterion that maximizes new information, in order to learn the concept with relatively fewer examples; see [Settles, 2009] for overview) could also be used. However, an important motivation for active learning, the high cost of producing labels (output patterns) in domains such as speech processing (in which human effort may be required), is not as important in this case, given that black box transformations can be called without human input.

It is the class of predictive models that form the basis of the optimization experiment of Chapter 4 and the learning experiment of Chapter 5; and indeed of the models used in the mosaicing algorithms of Chapter 6. However, some alternative model classes are possible (further sections).

In order to model cascades (series) of transformations, predictive models may be combined with function composition, as shown in the system design of Chapter 4.

Note that the input space to these functions is larger in the transformation case than in the synthesis case, if the parameter vector has otherwise the same dimension. This increase in dimensionality can, in some cases with certain types of models, cause the statistical problem of learning the model to become more difficult (see Appendix C.1).

3.4 Choosing Parameters

Given a predictive model (or a black box transformation), simply evaluating the model does not alone fulfill the synthesis goal. In order to achieve the goal of bringing the output to the target specified in descriptor space, an additional search process is necessary.

For the immediate problem of choosing transformation parameters $\vec{p}_{\mathrm{tr}}$ when at least some target descriptors $\vec{d}_{\mathrm{T}}$ are specified, this search process can take the form of a regularized optimization (explained in Appendix C.3):

$$\arg\min_{\vec{p}_{\mathrm{tr}}} \left[m\left(\vec{d}_{\mathrm{T}}, f_{\mathrm{pre}}(\vec{d}_{\mathrm{in}}, \vec{p}_{\mathrm{tr}}) \right) + f_{\mathrm{pen}}(\vec{p}_{\mathrm{tr}}) \right] = \vec{p}_{\mathrm{opt}}, \tag{3.4}$$

where m is some kind of metric, a distance function in the descriptor space, and f_{pen} is some penalty function that expresses some preferences for certain parameters over others. In general, this flexibility in choosing certain solutions over others, based on additional criteria, could be considered an advantage of the predictive model approach.

The penalties expressed by f_{pen} could include: a preference for smaller amounts of transformation (in order to minimize distortion of the input sounds), and if multiple time frames are considered, a preference for continuous changes in parameters (again, to minimize distortions resulting from temporal discontinuities in parameters). These two concerns are indeed represented in the penalty functions of Section 6.4.1 (p. 101).

However, the problem of selecting parameters (given in Eq. 3.4), is only a subproblem with regard to concatenative synthesis, for example, in which source sound units must also be selected and arranged (as in Eqs. 1.1, 1.2, p. 8). Rather, the full problem is a joint optimization, in which both source sound units and transformation parameters must be selected.

3.4.1 Complexity of Parameter Search

The difficulty of the search problems will vary with the individual effects and the number of effects involved, but in general the optimization problems will not necessarily be convex, which means that local search alone is not enough to find a global optimum. For an example of this, see the relatively simple joint optimization of a resampling parameter and equalization parameters, found in the system of Chapter 4. In these cases, either exhaustive search approaches such as grid search or global optimization, which can require effort exponential in the size of the problem, or heuristic search approaches, which are not guaranteed to find the best optimum, must be used.

However, in many cases, sets of gain parameters form linear representations (such as in filter bank equalization, or in modification of spectral or wavelet coefficients), leading to convex subproblems that can be solved efficiently with local search. For examples of this, see the system of [Olivero et al., 2013] focusing on time-frequency multipliers, or the use of quadratic smoothing to find filter gains as in [Coleman et al., 2010, Section 4. Unit and Parameter Selection].

3.5 Selective Models

As the goal is to find parameters that get the transformation output closest to the target (more precisely expressed by Eq. 3.4), one can also imagine a model that does this directly. In fact, this is possible, and in some cases fits the wider application goal.

So, the functional form of these models would be:

$$f_{\mathrm{sel}}(\vec{d}_{\mathrm{in}}, \vec{d}_{\mathrm{T}}) \approx \vec{p}_{\mathrm{opt}}, \qquad f_{\mathrm{sel}}(\vec{d}_{\mathrm{T}}) \approx \vec{p}_{\mathrm{opt}}, \tag{3.5}$$

for the transformation and synthesis cases respectively.

The algorithmic result of this model type, $\vec{p}_{\mathrm{opt}}$ is not necessarily the result of an optimization process, although it could be produced by a process as described in Eq. 3.4. Alternately, such a model could be learned by generalizing the same tuples from the black box transformation Eq. 3.1, by substituting the output vector $\vec{d}_{\mathrm{tr}}$ for the target vector $\vec{d}_{\mathrm{T}}$ in the model.

This latter approach, that of generalizing the black box tuples, is exactly the approach used for learning the selective models in the trio of additive synthesis systems controlled by pitch and loudness descriptors (Time-Axis, Perceptsynth, Audio-Driven; the latter including a brightness descriptor) as described in the previous chapter in Section 2.2.1.

This approach seems, at first glance, equivalent to the approach of predictive models described in Section 3.3. After all, the relation comprising the observed tuples could be the same for both model types, predictive and selective.

Even though it seems they are closely related, from several perspectives these forms are not necessarily equivalent. From a mathematical perspective, just the assumption that one of the models is a function (a many-to-one relation) would be different in the two model types. Smoothness assumptions, as well, used to learn models from data, would also differ given the different domains.

Finally, from a statistical learning perspective, the different input dimensionalities could affect the number of examples necessary to generalize with certain models (see Appendix C.1).

3.6 Why predictive models were used

For our applications in sample-based synthesis, there is a more practical objection to simple selective models. For many such applications, joint selection of a context of several units is required, whether simply favoring continuous changes in parameters to preserve continuity in synthesis (as mentioned in Section 3.4), or the joint selection over concatenative source units and unit transformation parameters as is done in the mosaicing algorithms of Chapter 6.

The regularized estimation of Eq. 3.4 (also having the same structure as the mosaicing unit selection Eqs. 1.1 and 1.2, p. 8) explicitly includes the form of the predictive model. The estimation's scope can be expanded by adding more variables to the argument, and new concerns can be added by adding new terms to the penalty function. In each case, predictive models can query each point in the parameter space to see how well it fits the context.

On the other hand, consider a selective model, which answers the question: "given an input $\vec{d}_{\mathrm{in}}$ and target $\vec{d}_{\mathrm{T}}$, what should the parameter vector $\vec{p}$ be?". If it is a mathematical function, it will always return the same $\vec{p}$ answer for any given $\vec{d}_{\mathrm{in}}$ and $\vec{d}_{\mathrm{T}}$. Thus, with the form of Eq. 3.5, it is not straightforward to search among variations in sets of $\vec{p}$ parameters, for example, that are continuous in time, because the selective model gives you just a single $\vec{p}$ answer for each query.

Perhaps it is possible to rehabilitate the selective model for this purpose, e.g. by adding an additional neighborhood parameter $\vec{p}_0$, as in:

$$f_{\mathrm{sel+}}(\vec{d}_{\mathrm{in}}, \vec{d}_{\mathrm{T}}, \vec{p}_0) \approx \vec{p}_{\mathrm{opt}}. \tag{3.6}$$

In that case, in solving those problems, it would still be necessary to search in parameter space; whereas the idea of selective models was to be able to avoid explicit parameter seach.

As the subproblem of parameter selection is to solve regularized problems of the form given by Eq. 3.4, predictive models were used in this thesis.

3.7 Conditional models

One criticism of the previous model types described is that they are pointwise estimates, i.e. in both cases, they return a single point in descriptor or parameter space, but do not reflect the uncertainty of the model estimate.

Although this type of model is not treated substantially in this thesis, the following is a brief sketch of what they could look like.

For example, in the case of predictive models, these models assume that if the input and parameters are known, the output is exactly known in descriptor space. However, perhaps the uncertainty of the estimate is greater in some descriptors rather than others, or even greater in some part of the input and parameter spaces than others.

To express the uncertainty in the output descriptor space, a conditional probabilistic model could be used, perhaps of this form:

$$f_{\vec{D}_{\mathrm{tr}}}(\vec{d}_{\mathrm{tr}}|\vec{D}_{\mathrm{in}} = \vec{d}_{\mathrm{in}}, \vec{P}_{\mathrm{tr}} = \vec{p}_{\mathrm{tr}}). \tag{3.7}$$

Using this type of model, the parameter or unit selection processes could take into account the estimated uncertainty in the predicted output descriptors, and perhaps manage this risk by avoiding the riskier areas of the search spaces. This type of model could also account for accumulated uncertainty in cascades of transformations, something that is not done with pointwise predictive models.

3.8 Error Measures in Model-based Design

As model-based systems (models of transformations, models of mixtures) rely on estimates for controlling the synthesis outcome, it is useful to make a distinction between different types of error encountered by the system (e.g. for debugging purposes).

That is, descriptor controlled synthesis and transformation can be viewed as a progressive projection of the ideal descriptor target into the real synthesis space in which error is accumulated in the multiple steps.

One clear distinction that can be made, is the distinction between, on one hand, the deviation in the model-predicted optimum of the search and the target, versus the deviation between that model-predicted optimum and the output created by synthesis.

From the three vectors that can be measured: $\vec{d}_{\mathrm{T}}$, the target vector, $\vec{d}_{\mathrm{opt}} = f_{\mathrm{pre}}(\vec{d}_{\mathrm{in}}, \vec{p}_{\mathrm{opt}})$, the optimum from parameter selection in descriptor space, and $\vec{d}_{\mathrm{tr}}$, the actual output in descriptor space; the following three error measures can be derived.

By comparing estimates of the first two error vectors, one can see which causes more error in the system: either distance in the model class (e.g. with different transformations or unit databases), or inaccuracies in the model itself.

3.8.1 Model Error

The model error is defined as the vector from the target to the optimum in the model space.

$$\mathrm{ME}(\vec{d}_{\mathrm{T}}, \vec{d}_{\mathrm{opt}}) = \vec{d}_{\mathrm{T}} - \vec{d}_{\mathrm{opt}}. \tag{3.8}$$

This optimum could either be a true optimum, or if using approximate solutions this would also incorporate error contributed by the search process. Another error type could also be added to account for this search error, but here it is omitted for simplicity.

3.8.2 Empirical Error

The empirical error is defined as a vector from the expected result from the model to the real synthesized result:

$$\mathrm{EE}(\vec{d}_{\mathrm{opt}}, \vec{d}_{\mathrm{tr}}) = \vec{d}_{\mathrm{opt}} - \vec{d}_{\mathrm{tr}}. \tag{3.9}$$

3.8.3 Total Error

The total error is defined as a vector from the target to the real synthesized result:

$$\mathrm{TE}(\vec{d}_{\mathrm{T}}, \vec{d}_{\mathrm{tr}}) = \vec{d}_{\mathrm{T}} - \vec{d}_{\mathrm{tr}}. \tag{3.10}$$

By the triangle inequality (if using a distance metric like the Euclidean norm):

$$\|\mathrm{TE}\| \leq \|\mathrm{ME}\| + \|\mathrm{EE}\|, \tag{3.11}$$

i.e. the total error is upper bounded by the sum of "component" errors, and one can compare the relative amounts of error from model and empirical errors.

A summary of the error measures is depicted in Figure 3.2.

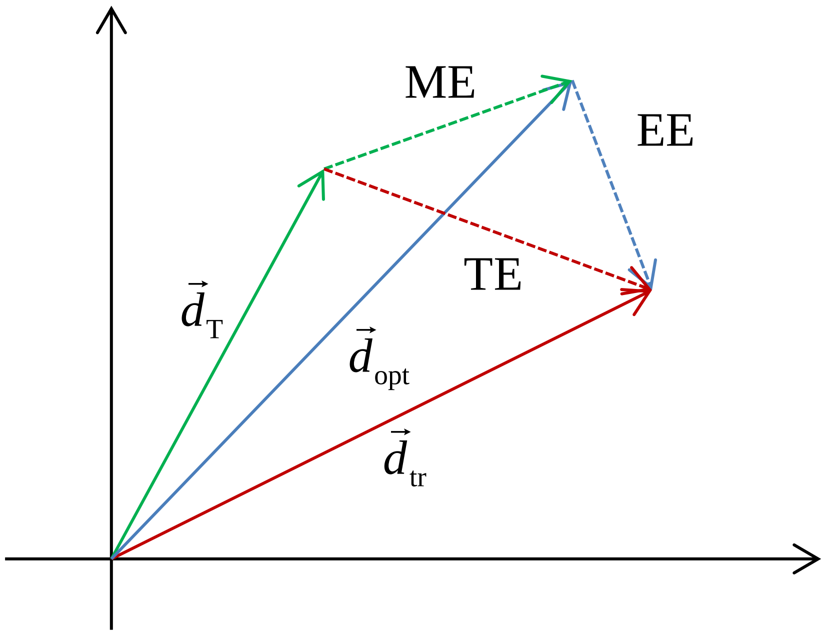

Figure 3.2: An illustration of the measurable descriptor vectors (solid lines) along with error vectors (dotted lines).

Chapter 4

Control of Sound Transformations by Target Descriptors using gradient-based search methods and Predictive Models

This chapter covers the system design and experiment reported previously in "Sound Transformation by Descriptor using an Analytic Domain" [Coleman and Bonada, 2008]. The previous paper is revised and somewhat expanded.

4.1 Abstract

In many applications of sound transformation, such as sound design, mixing, mastering, and composition the user interactively searches for appropriate parameters. However, automatic applications of sound transformation, such as mosaicing, may require choosing parameters without user intervention. When the target can be specified by its synthesis context, or by example (from descriptors of the example), "adaptive effects" can provide such control. But there exist few general strategies for building adaptive effects from arbitrary sets of transformations and descriptor targets.

In this study, the usually direct (procedural) link between analysis and transformation in adaptive effects is decoupled. This link is instead replaced by a coupling of predictive models of effects (overview in Ch. 3) with search methods used to find optimal transformation parameters within those

models.

This is meant to allow more diverse sets of transformations and descriptors in the field of adaptive effects, besides those for which direct mappings between effects parameters and descriptors are already known. However, this approach has an added cost in computational effort, mainly that incurred in the parameter search (see Section 3.4 in the previous chapter).

Analytic predictive models are developed for two simple effects, resampling and equalization, and combined to model a cascade of those effects. This model is then searched to find optimal transformation parameters given a target, and the numerical accuracy of the control is examined.

4.2 Introduction

Sound transformations (commonly: effects) are practically used by sound and music producers in a variety of contexts: mixing, mastering, synthesis, composition, sound design for varying media. Effects are typically modeled as mathematical functions transforming one or more input audio signals into output signals according to a set of numerical parameters. These parameters usually are tuned either interactively or according to some knowledge of the transformation domain.

Because this process can be immediate and interactive, it is usually fast and effective for a user to find parameters which correspond to their target sound goals for the input sounds in question. However, the assumption that parameters can be effectively manually tuned could break down under several conditions: if the parameter space is too large (there are too many parameters), if that space is too complex (e.g. nonlinear) to be interactively searched, or if the desired result needs to be synchronized or finely articulated in time.

For example, consider an automatic mosaicing system that selects "source" sound samples from a database to match input "target" samples, then composites them into a score (such as those discussed in Section 2.1). To improve the match quality, the retrieved source sounds could be transformed to be more similar to their targets. But for the system to be automatic, the transformation parameters should be selected without human input.

Adaptive effects "in which controls are derived from sound features" [Verfaille and Depalle, 2004] (the terms descriptor and feature are used here synonymously) are often implemented directly via an analysis-synthesis

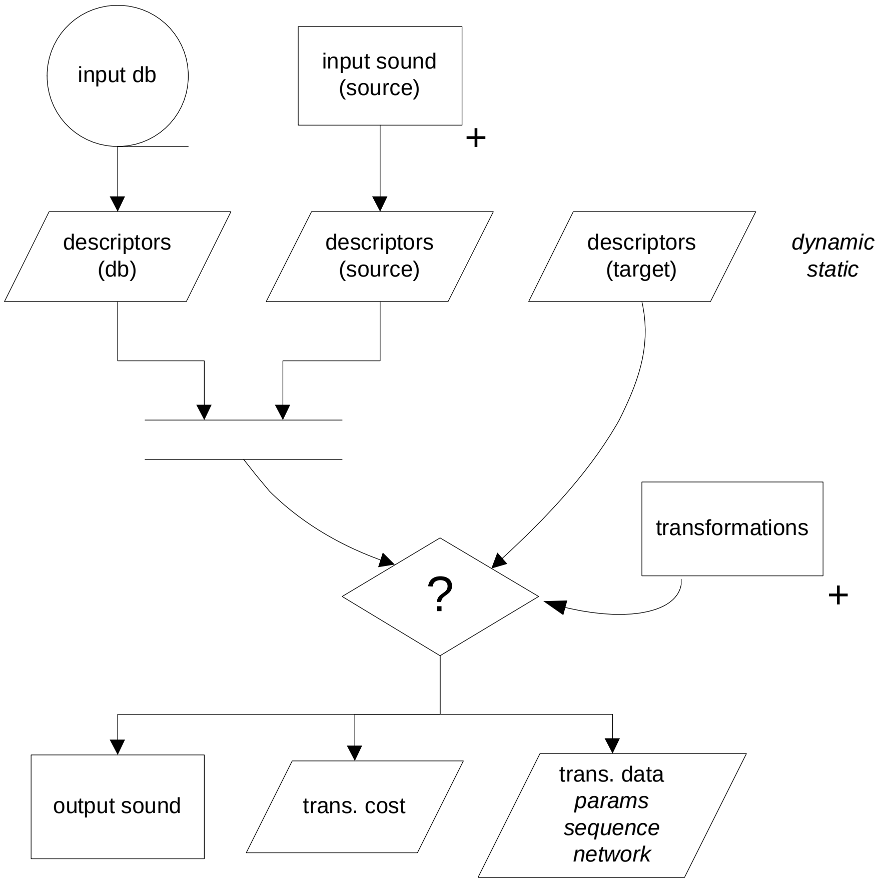

Figure 4.1: An ideal transformation by descriptor system, which uses available transformations to bring input sounds close to a target, and also discriminates between candidate source sounds in a database.

paradigm, with a modification of the analysis representation used to steer the result of the synthesis towards the target. This offers a direct route to control effects by descriptors, usually by exploiting mathematical properties of transforms such as the short-time Fourier transform (STFT) or the source-filter model, in such a way that allows independent algorithmic modification of desired properties of the sound. In these systems, the analysis of the target is coupled directly with the transformation of the input material. However, this approach requires the development of a procedural model that links the fixed transformation with effective, independent changes in a fixed target descriptor set; hence with many sets of transformations and descriptors, this may not be possible.

An alternative approach would allow consideration of any target descriptors of interest, using the set of transformations that are available. By breaking the link between analysis and transformation, we intend to allow both wider and more complex sets of effects and descriptors, but also wider sets of criteria to be considered.

Instead of using a transformation domain that allows direct modification according to a target, an alternative strategy is to build predictive models of sets of transformations in descriptor spaces (in previous version of this article: a "transformation-descriptor" (TD) space).

By first determining the relationship between the input sound, the transformation parameters, and the output descriptors, one can provide a map of the space. Then, numerical optimization techniques can be used to find transformation parameters that best meet the target (as well as other objectives). Thus, this numerical search provides an additional link in the middle of this chain, after the analysis and before transformation/synthesis, where procedural mappings were in the previous work.

4.3 Related Works

Several previous works [Arfib and Verfaille, 2001; Verfaille and Arfib, 2002; Verfaille and Depalle, 2004; Amatriain et al., 2003; Verfaille et al., 2006b,c]; [Verfaille, 2003 as cited by Verfaille et al., 2006c]) have as their subject adaptive transformations. As defined in [Verfaille et al., 2006c], these effects are controlled by "a time-varying control derived from sound features transformed into valid control values using specific mapping functions". For example, in [Verfaille and Depalle, 2004], the STFT and source filter model form a basis over which some aspects of the sound (at once descriptors and parameters) can be independently modified.

As arbitrary or more complex sets of descriptors are included, the chance of extending or discovering such "specific mapping functions" lessens. As an antidote to the specificity of these mappings, we propose an alternative that is somewhat generic. First, that the domain of parametric transformations can be modeled with respect to the target descriptors; second, that numerical search (the generic part) fills in the procedural gap.

Concatenative synthesis synthesizers, ranging from a those dealing with a single-instrument (like the singing voice, as in [Bonada and Serra, 2007]) to audio mosaicing (which deals with diverse sound material, as in [Schwarz, 2007]) use similar synthesis techniques. These systems generate sequences with different target descriptor sequences from a limited sample database, and could likewise be extended by transforming the input sound closer to its intended target. In this application context, we would also like to be able to select source samples that are most easily transformed to the targets. Figure 4.1 illustrates this concept of "Transformation by Descriptor" in a sampling synthesis context.

Perhaps the closest work, in the area of generic control of sound synthesis, is that of [Hoffman and Cook, 2007] (previously mentioned in Section 2.2.1). Similar to the work described in this chapter, it uses indirection between analysis and synthesis, using numerical optimization to control parametric synthesizers from frame-based audio features, and thus explores a similar technique. The main difference between this and that work, is that this work uses models whereas that work does not, and this work deals with transforming sounds (the output sound is generated by the input sound samples) whereas that work uses synthesis models controlled by target descriptors.

4.4 Predicting Descriptors under Transformation

By modeling our transformation space with predictive models, we intend to guide the search process towards its intended target. Each transformation is thus modeled as a function mapping an input sound (represented by its descriptors) and a parameter vector, to an output sound again represented by descriptors (as proposed in Section 3.3, although the notation here differs slightly from the previous chapter).

Feature vector of input sound : $\vec{d_{\text{in}}}$ (length determined by context)
Hopefully something that predicts $\vec{d_{\text{tr}}}$ well.

Feature vector of transformed sound : $\vec{d_{\text{tr}}}$ of length D
There are D descriptors of interest, need not match $\vec{d_{\text{in}}}$.

Vector of transformation parameters : $\vec{p}$

The transformation function : $\vec{t}(\vec{d_{\text{in}}}, \vec{p}) = \vec{d_{\text{tr}}}$

Model of transformation : $\hat{t}(\vec{d_{\text{in}}}, \vec{p}) \approx \vec{d_{\text{tr}}}$
In other words, an approximation of $\vec{t}$, the real transformation.

Target feature vector : $\vec{T}$ of length D (matches $\vec{d_{\text{tr}}}$ in dimension).

4.4.1 Descriptors: Spectral Magnitudes and Moments

The input descriptors used are the spectral magnitudes, i.e. $|X_k|$ where k is an index of a positive frequency bin.

The descriptors of interest (predicted output descriptors and target descriptors) are statistical moments of the spectrum (such as those in [Peeters, 2004, Section 6.1, Spectral Shape descriptors]). In general, statistical moments describe global (as opposed to local, as in limited to certain frequencies) shape characteristics of a distribution.

The resampling model in this section applies to arbitrary and higher order moments, such as skewness and kurtosis; although the automatic control experiments reported in Section 4.7.3 used only spectral centroid and spectral spread (standard deviation).

The spectral moments are defined as functions (specifically, expectations) of the normalized magnitude spectrum NMS, where the kth normalized magnitude is defined as:

$$\textsc{Nms}_k(x) = \frac{|X_k|}{\sum |X_i|}. \tag{4.1}$$

The first two spectral moments, the spectral centroid and spectral std. deviation, are defined as follows:

$$d_{\text{mean}} = \mu = \frac{\sum k|X_k|}{\sum |X_k|}, \qquad d_{\text{std}} = \sigma = \sqrt{\frac{\sum (k-\mu)^2 |X_k|}{\sum |X_k|}}. \tag{4.2}$$

Skewness and kurtosis are central moments (with the mean subtracted) normalized by powers of the standard deviation:

$$d_{\text{skew}} = \frac{\sum (k-\mu)^3 |X_k|}{\sigma^3 \sum |X_k|}, \qquad d_{\text{kurt}} = \frac{\sum (k-\mu)^4 |X_k|}{\sigma^4 \sum |X_k|}. \tag{4.3}$$

As a shorthand, these moments and higher moments (except d_{std}) can be represented in the following way. Let m be a function with arguments k, μ and σ. Then, each moment can be represented as a function composition of an individual m function with a function M of a single variable z:

$$M(z) = \mathbb{E}_{|X|}[z] = \frac{\sum z|X_k|}{\sum |X_k|}, \qquad d_m = m \circ M = \frac{\sum m(k,\mu,\sigma)|X_k|}{\sum |X_k|}, \tag{4.4}$$

with $m_{\text{mean}}(k) = k$ for the spectral centroid, $m_{\text{var}}(k) = (k-\mu)^2$ for the spectral variance, and $m_{\text{cm}n}(k) = (k-\mu)^n$ for higher central moments.

4.4.2 Transformations Overview

In these experiments, two simple transformations are modeled. These transformations are bandlimited interpolation (resampling) and linearly spaced bandpass equalization. In this case, the transformation parameters are the resampling factor L, and the vector $\vec{h}$ of log-gains (see Section 4.4.4) for each of the B equalization bands:

$$\vec{p} = (L, \vec{h}) = (L, h_1, h_2, ..., h_B). \tag{4.5}$$

A model is introduced that predicts the coordinated action of both transformations in the spectrum. Two phenomena are omitted from consideration and could be considered potential sources of model error. One potential source of model error is the bandlimited interpolation (which relies on a filter bank to reconstruct the ideal sinc function), and the other potential source of model error comes from using a reduced version of the input spectrum (also a filter bank) to approximate the full spectrum.

4.4.3 Resampling

Resampling (consisting of upsampling and downsampling) is used to change the pitch and duration of sounds dependently, with a single parameter L determining the ratio of the output duration to the input duration.

Changing the length of a signal in the time domain (by, e.g. stretching) has an inverse effect in the frequency domain (compressing), and vice versa. Smith [2007, Continuous Fourier Theorems, Scaling Theorem] states the following, which applies to a continuous signal $x(t)$ with Fourier transform:

$$\textsc{Scale}_\alpha(x) \longleftrightarrow |\alpha| \, \textsc{Scale}_{(1/\alpha)}(X), \qquad \textsc{Scale}_{\alpha,t}(x) \triangleq x\left(\frac{t}{\alpha}\right), \tag{4.6}$$

where α (corresponding to parameter L above) is a nonzero real number.

However, the previous result applies to continuous signals, not to sampled signals. There are additional complications that apply to stretching or compressing of sampled signals.

Uniformly sampled signals can only represent a limited frequency range, that between $[-f_s/2, f_s/2]$, where f_s is the sampling frequency. That is, exact reconstruction of a continuous signal is only possible if the original signal was bandlimited to that interval (meaning no energy was present at other frequencies) [Smith, 2007, Sampling Theory, Sampling Theorem]. $f_s/2$, the upper frequency limit, is referred to as the Nyquist frequency.

When an input signal contains energy outside of this frequency range, that energy is irrevocably mixed with energy at certain lower frequencies, in a process (usually unwanted) referred to as aliasing [Smith, 2007, Signal Operators, Alias Operator]. (This problem is normally dealt with by low-pass filtering signals before sampling them). However, aliasing is relevant not only to sampling continuous signals, but also to resampling (changing the effective sampling rate).

When stretching a signal in time (upsampling, $L > 1$), the (previously bandlimited) frequencies are scaled down, so there is no risk of aliasing and no need for further bandlimiting. On the contrary, when compressing a signal in time (downsampling, $L < 1$), additional bandlimiting is necessary, which occurs due to the action of a low-pass filter integrated into the resampling process [Smith, 2007, Interpolation Theorems, Bandlimited Interpolation of Time-Limited Signals].

In practice, bandlimited interpolation methods use finite approximations to an ideal sinc filter [Smith, 2002, Theory of Operation, Theory of Ideal Bandlimited Interpolation], which adds error to the resampled signal. However, this is not taken into account into the following model.

The bandlimited normalized spectrum, BlNms, is modeled with a "brick-wall" rectangular mask to simulate bandlimiting. This is done by substituting for $|X_k|$ in the expression for Nms (Eq. 4.6):

$$\textsc{BlNms}_{L,k}(x) = \frac{\textsc{BlB}\left(\frac{k}{KL}\right)|X_k|}{\sum \textsc{BlB}\left(\frac{i}{KL}\right)|X_i|}, \tag{4.7}$$

where $\frac{k}{KL}$ is the scaled normalized frequency, and the binary bandlimiting mask, BlB, is derived from the Heaviside step function H:

$$\textsc{BlB}(x) = 1 - H(x-1) = \begin{cases} 1 & x \leq 1 \\ 0 & \text{otherwise.} \end{cases} \tag{4.8}$$

Substituting BlNms in $M(z)$ (Eq. 4.4), and scaling the frequency argument (as indicated by Eq. 4.6) yields a model of resampling for spectral moments:

$$\hat{t}_r(X, L, z) = \frac{\sum \frac{z}{L} \cdot \textsc{BlB}\left(\frac{k}{KL}\right)|X_k|}{\sum \textsc{BlB}\left(\frac{k}{KL}\right)|X_k|}. \tag{4.9}$$

4.4.3.1 Sigmoids

However, the binary mask BLB based on the step function is discontinuous; its derivative is zero everywhere except at the discontinuity. Thus it is less useful in a local search technique, in which derivatives are used to inform the search method of the local behavior of the function.

Instead, a surrogate composed with the sigmoid function is used:

$$\begin{aligned} \textsc{Bls}_\alpha(x) &= 1 - S(\alpha(x-1)) && (4.10) \\ &= 1 - \frac{1}{1+e^{\alpha(x-1)}}, && (4.11) \end{aligned}$$

where the sigmoid function is defined as:

$$S(x) = \frac{1}{1+e^{-x}} \tag{4.12}$$

having the derivative:

$$\frac{dS}{dx} = S(1-S) = \frac{e^x}{(1+e^x)^2}. \tag{4.13}$$

Using a sigmoid, the resulting model is a smooth function, which is necessary for using gradient descent based optimization methods. (Indeed, in this manner sigmoids are used to facilitate back propagation, a gradient descent method used to learn neural network parameters).

This does not completely solve the problem of the local minima created by the resampling, it just allows the search to find the local minima easier by giving them directional cues.

4.4.4 Equalization

Equalization is modeled as a bank of rectangular filters that partition the spectrum into B linearly-spaced bands, apply non-negative gains $\vec{g}$, and then add the scaled signals. In practice, the filters used will overlap and contain small amounts of energy from other bands.

Thus, each transformed magnitude is modeled as being scaled by the appropriate, non-negative gain:

$$\hat{t}_{\text{eq}}(|X_k|, \vec{g}) = g_{j(k)} \cdot |X_k| \tag{4.14}$$

where $j(k)$ indicates the filter bank that corresponds to the spectral bin k.

One way of constraining the gains to be non-negative is to use log-domain (exponents) $\vec{h}$ to control the gains, such as $g_j = 2^{h_j}$, giving us:

$$\hat{t}_{\text{eq}}(|X_k|, \vec{p}) = 2^{h_{j(k)}} \cdot |X_k|, \tag{4.15}$$

which are equivalent (up to a constant) to specifying gains in decibels.

4.4.5 Composition

In order to model a cascade of transformations, the predictive models can be combined using function composition. For example, to model the series of (equalization, resampling), the composed model is as follows:

$$\hat{t}_{\text{eq,r}}(X, \vec{p}) = \hat{t}_r \circ \hat{t}_{eq} = \hat{t}_r(\hat{t}_{eq}(X, \vec{d}), L). \tag{4.16}$$

In this case, that particular sequence was chosen for mathematical convenience. Conceptually, independent sets or bands of Fourier coefficients are scaled by the equalization, which are then shifted and possibly bandlimited by the resampling. If composed into the opposite sequence, membership in a given equalization band would vary with resampling parameter L.

The combined predictive model is as follows:

$$\hat{t}_{\text{eq,r}}(X, \vec{p}, z) = \frac{\displaystyle\sum_{j=1}^{B} 2^{h_j} \cdot \textsc{Bls}\left(\frac{j}{BL}\right) \cdot \sum_{k \in k(j)} \frac{z}{L} \cdot |X_k|}{\displaystyle\sum_{j=1}^{B} 2^{h_j} \cdot \textsc{Bls}\left(\frac{j}{BL}\right) \cdot \sum_{k \in k(j)} |X_k|} \tag{4.17}$$

where $k \in k(j)$ are the Fourier indices that fall in a given equalization band. (The above model makes the additional approximation of using frequency masks over filter bands, rather than individual bins, as in Eq. 4.9).

4.4.6 Behavior of Transformation / Model

To give an intuition for the shape of the space we wish to model and search, several plots are examined. The first, shown in Figure 4.2, shows the effect of resampling on a set of 10 sounds. The plot shows the ratio of spectral centroids (output to input) when different values of the resampling parameter L are used. On the right side of the function ($L \geq 1$), the scaling rule

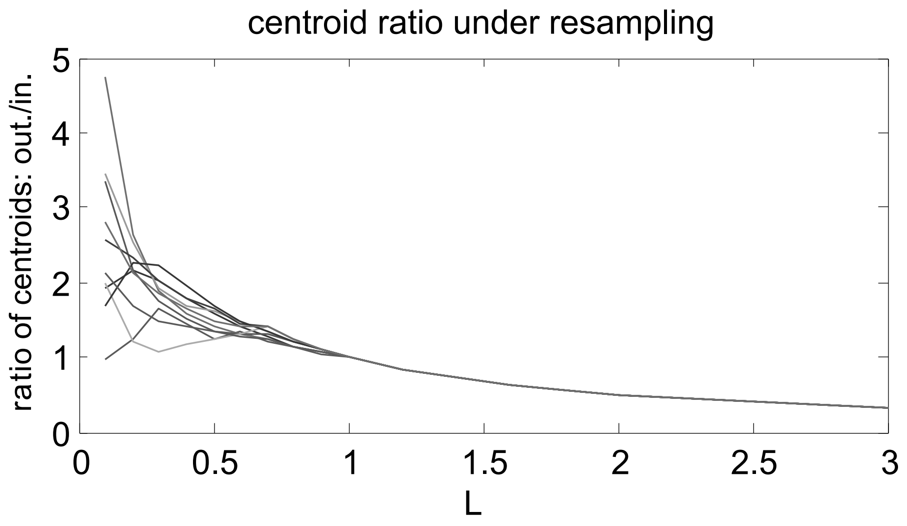

Figure 4.2: Centroid ratios (output centroid to input centroid) diverge when $L < 1$ due to bandlimiting.

functions perfectly as a rule for predicting the output centroid. In strong contrast, on the left side with $L < 1$, the centroid ratios depend strongly on the individual spectra, as different pockets of energy are bandlimited away at different values of L.

The next plot, Figure 4.3 shows a grid sampling of a cross-section of the transformation space of resampling and equalization for the standard deviation as a descriptor. By looking at slices of constant h_3, it can be seen that 1) the d_{std} is non-monotonic with respect to varying L, and 2) these curves are smooth, perhaps due to the smoothness of the bandlimiting.

By contrast, the moments seem to be monotonic with respect to equalization. Looking at Eq. 4.17, we see that the $2^{h_j} \cdot \textsc{Bls}\left(\frac{j}{BL}\right)$ subexpression merely determines the non-negative weight applied to each scaled z in the spectrum, leading to limiting cases forming two poles. When one amplifies a band far above the rest of the bands, or far below, it either predominates or becomes insignificant in the summation, giving it the slow limiting effect, as seen in slices of constant L and varying h_3.

This gives us hope that in the error space will be relatively smooth and

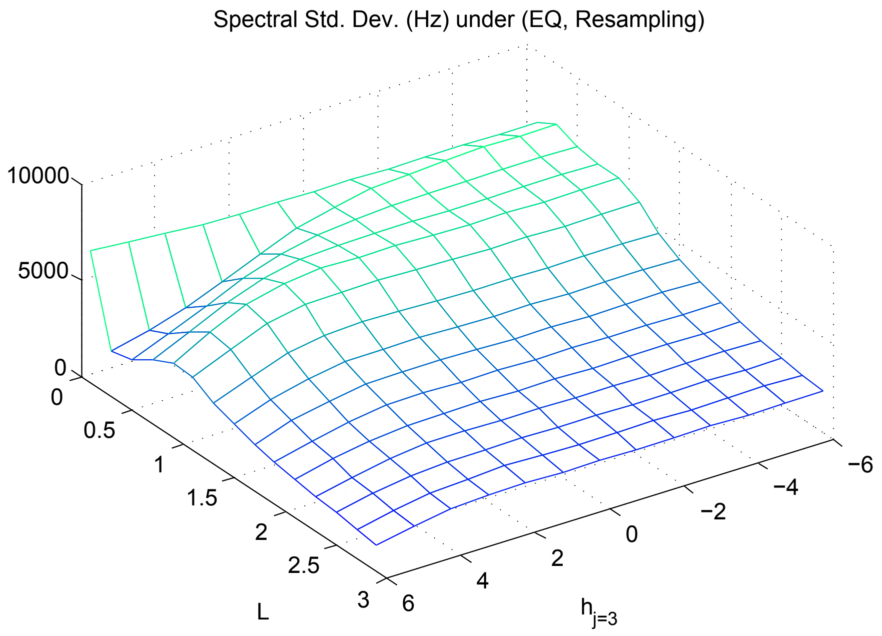

Figure 4.3: Cross-section of actual transformation space for one variable band (out of 16) and a resampling parameter.

easy to follow the gradient in these dimensions, with the non-monotonicity being confined to the resampling dimension.

4.5 Parameter Selection by Smooth Optimization

For each sound to be transformed, there are input descriptors $\vec{d_{\text{in}}}$, a predictive model $\hat{t}(\vec{d_{\text{in}}}, \vec{p})$, and a target descriptor vector $\vec{T}$.

When we use a model to approximate the effect of the transformation, we replace $\vec{t}$ with $\hat{t}$. and an error term e. The residual is defined as the difference vector between the target and the predictive model (similar to the Model Error, Section 3.8):

$$\vec{r}(d_{in}, \vec{p}) = \vec{T} - \hat{t}(\vec{d_{\text{in}}}, \vec{p}) + e. \tag{4.18}$$

To optimize the transformation parameters, some function of the residual is minimized. For mathematical convenience, the sum of squares of the individual terms (least-squares) is often chosen [Nocedal and Wright, 1999]:

$$f(\vec{d_{\text{in}}}, \vec{p}) = \frac{1}{2} \sum_{j=1}^{m} r_j^2(\vec{d_{\text{in}}}, \vec{p}). \tag{4.19}$$

4.5.1 Partial Derivatives

Partial derivatives of the objective function f are a basic ingredient of local search techniques (such as gradient descent). The gradient is defined as the vector of partial derivatives with respect to each of the parameters:

$$\nabla f(\vec{p}) = \left(\frac{\partial f}{\partial L}, \frac{\partial f}{\partial h_1}, \dots, \frac{\partial f}{\partial h_B} \right), \tag{4.20}$$

Using the least-squares objective function, the gradient reduces to:

$$\nabla f(\vec{p}) = \sum_{j=1}^{m} r_j(\vec{p}) \nabla r_j(\vec{p}) = J(\vec{p})^T \vec{r}(\vec{p}), \tag{4.21}$$

where J, the Jacobian matrix, is defined as the partial derivative of the residual in each parameter:

$$J(\vec{p}) = \begin{bmatrix} \frac{\partial r_1}{\partial L} & \frac{\partial r_1}{\partial h_1} & \cdots & \frac{\partial r_1}{\partial h_B} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r_D}{\partial L} & \frac{\partial r_D}{\partial h_1} & \cdots & \frac{\partial r_D}{\partial h_B} \end{bmatrix}. \tag{4.22}$$

As the target $\vec{T}$ is constant and the error term e can be assumed to be independent, the partial derivatives of the residual are simply the partial derivatives of the transformation model $\hat{t}$, and have the following form (derivative of a quotient):

$$\frac{\partial \vec{r}}{\partial \vec{p}} = \frac{\partial \hat{t}}{\partial \vec{p}} = \frac{\frac{\partial \text{top}}{\partial \vec{p}} \text{bot} - \text{top} \frac{\partial \text{bot}}{\partial \vec{p}}}{\text{bot}^2}, \tag{4.23}$$

given subexpressions for the top and bottom:

$$\text{top} = \sum_{j=1}^{B} 2^{h_j} \cdot \textsc{Bls}\left(\frac{j}{BL}\right) \sum_{k \in k(j)} \frac{z}{L} \cdot |X_k|, \tag{4.24}$$

$$\text{bot} = \sum_{j=1}^{B} 2^{h_j} \cdot \textsc{Bls}\left(\frac{j}{BL}\right) \sum_{k \in k(j)} |X_k|. \tag{4.25}$$

A subexpression is defined for the summation over each band:

$$\text{band}_j = \sum_{k \in k(j)} \frac{z}{L} \cdot |X_k|. \tag{4.26}$$

This gives the following partial derivatives for L:

$$\frac{\partial \text{band}_j}{\partial L} = \sum_{k \in k(j)} \left(\frac{\partial z}{\partial L} \cdot \frac{1}{L} - \frac{z}{L^2} \right) \cdot |X_k|, \tag{4.27}$$

$$\begin{aligned} \frac{\partial \text{top}}{\partial L} = \sum_{j=1}^{B} 2^{h_j} \cdot \Bigg[&\textsc{Bls}' \left(\frac{j}{BL} \right) \cdot \frac{-j}{BL^2} \cdot \text{band}_j + \\ &\textsc{Bls} \left(\frac{j}{BL} \right) \cdot \frac{\partial \text{band}_j}{\partial L} \Bigg], \end{aligned} \tag{4.28}$$

where:

$$\textsc{Bls}'_\alpha(x) = -\alpha \cdot S(\alpha(x-1)) \cdot (1 - S(\alpha(x-1))). \tag{4.29}$$

The bottom expression is similar to the top, but simpler (since there is no scaling term in L):

$$\frac{\partial \text{bot}}{\partial L} = \sum_{j=1}^{B} 2^{h_j} \cdot \textsc{Bls}' \left(\frac{j}{BL} \right) \cdot \frac{-j}{BL^2} \cdot \text{band}_j, \tag{4.30}$$

and the following partial derivatives in terms of the log-gains h_j:

$$\frac{\partial \text{top}}{\partial h_j} = 2^{h_j} \ln 2 \cdot \textsc{Bls} \left(\frac{j}{BL} \right) \sum_{k \in k(j)} \left(\frac{\partial z}{\partial h_j} \cdot \frac{1}{L} \right) \cdot |X_k|, \tag{4.31}$$

$$\frac{\partial \text{bot}}{\partial h_j} = 2^{h_j} \ln 2 \cdot \textsc{Bls} \left(\frac{j}{BL} \right) \sum_{k \in k(j)} |X_k|. \tag{4.32}$$

Some additional complications are necessary if the partial derivatives are to be computed exactly. For example, the dependencies introduced by partial derivatives of z ($\frac{\partial z}{\partial L}$ in Eq. 4.27, $\frac{\partial z}{\partial h_j}$ in Eq. 4.31) are complex indeed (due to the recursive nature of the spectral moments).

The partial derivatives for general central moments of the form:

$$m_{\text{cent}}(k, \mu, \sigma) = \frac{(k-\mu)^a}{\sigma^n}, \tag{4.33}$$

including variance, skewness, and kurtosis, with respect to a parameter p are as follows:

$$\frac{\partial m_{\text{cent}}}{\partial p} = \frac{1}{\sigma^{2n}} \left[-a(k-\mu)^{a-1}\sigma^n \frac{\partial \mu}{\partial p} - n\sigma^{n-1}(k-\mu)^a \frac{\partial \sigma}{\partial p} \right]. \tag{4.34}$$

The partial derivatives of μ and σ are the same as must be computed for the top-level partial derivatives (Eq. 4.23):

$$\frac{\partial \mu}{\partial p} = \frac{\partial \hat{t}_{\text{mean}}}{\partial p}, \qquad \frac{\partial \sigma}{\partial p} = \frac{\partial \hat{t}_{\text{std}}}{\partial p}. \tag{4.35}$$

For the partial derivative with respect to the standard deviation, a square root must be added:

$$\frac{\partial \hat{t}_{\text{std}}}{\partial p} = \frac{\partial \sqrt{t_{\text{var}}}}{\partial p} = \frac{\partial \hat{t}_{\text{var}}}{\partial p} \cdot \frac{1}{2\sqrt{\hat{t}_{\text{var}}}}. \tag{4.36}$$

4.6 Numerical Search

Many optimization methods assume f is smooth, and use its derivatives to navigate around the space. Once a search direction is chosen, several function evaluations are done at different distances in a process known as line search. In most cases the search direction should be a descent direction, i.e. a direction in which the function is decreasing.

For simplicity of development, we have used the normalized gradient descent with backtracking, with the normalized gradient descent itself as our search direction.

As our error surface is likely non-convex (and thus has local minima) the iterated line search will only return one of several local minima. To get around this problem, we can start the search in different places to sample different local minima, known as randomized gradient descent.

4.6.1 Penalty Terms (Regularization)

To encourage less extreme parameter transformations when possible, penalty terms were added to the objective function f (Eq. 4.19). For the log-gains, the following penalty was used (penalizing gains away from unity):

$$f_{\text{eq}}(\vec{h}) = \kappa_{\text{eq}} \sum_{j=1}^{B} h_j^2, \qquad \frac{\partial f_{\text{eq}}}{\partial h_j} = 2 \cdot \kappa_{\text{eq}} \sum_{j=1}^{B} h_j, \tag{4.37}$$

and for the resampling parameter L, the following penalty was used:

$$f_{\mathrm{r}}(L) = \kappa_{\mathrm{r}}[\max(L, 1/L)]^2, \qquad \frac{\partial f_{\mathrm{r}}}{\partial L} = 2 \cdot \kappa_{\mathrm{r}} \max(L, 1/L). \tag{4.38}$$

The ability to favor certain areas of the parameter space, is an advantage in flexibility afforded by control using predictive models.

4.7 Basic Experiments

4.7.1 Development Database

A small database of 10 sounds of different types of audio signals was assembled, in order to test the predictors and basic optimization tests. The sounds were collected from Freesound [Music Technology Group, 2005] and included speech (adult, baby), sounds (dishes, mouth pop), musical instruments (harmonica, gong), several synthetic electronic beats, and environmental noise.

A minority of the sounds were less than one second long, and longer sounds were truncated to that duration for the experiments. The input sounds were analyzed with a number of spectral moments, and some miscellaneous descriptors including zero crossing and power related descriptors.

4.7.2 Predictors

In the first experiment we attempted to predict a variety of simple time and Fourier domain descriptors under either resampling or bandpass filtering. Results are described in Figure 4.4.

4.7.3 Transformation by Descriptor

To test prediction and target-based optimization end-to-end, we used each of the sounds as an input sound, and likewise each of the sounds as a target, using the extracted spectral centroid and standard deviation, for a total of 10x10 trials, including the identity trials (serving as a sanity check). Free parameters for the experiment included constants for the parameter penalty terms, choice of sigmoid steepness α, and parameters of randomized line search (trials, starting distribution, step size, contraction rate, etc), all of which were chosen by hand.

Once parameters are chosen for each set of targets, the input sounds were transformed according to those parameters. Then, the descriptors of the

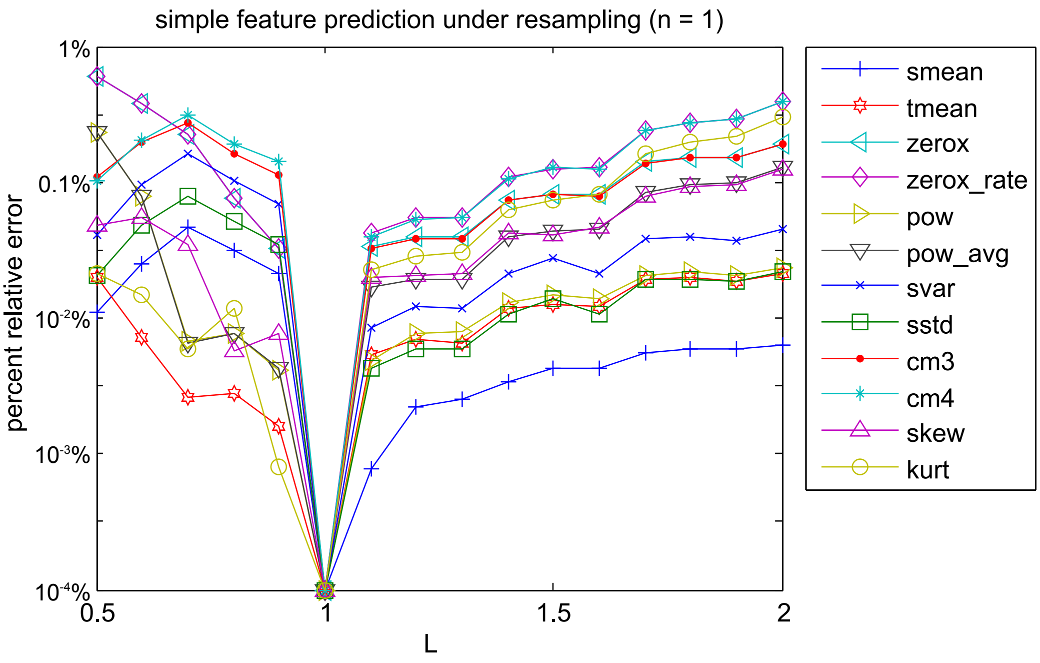

Figure 4.4: A sound is resampled at different rates, and descriptors such as spectral mean, variance, skewness, kurtosis, power, temporal centroid, etc are predicted given the input descriptors and the resampling rate L. Error is given as relative error (deviation from actual value / actual value) in the log domain. At $L = 1$ we have the identity transformation, thus no error in prediction.

transformed sound were measured, and compared with the original target descriptors. Figure 4.5 shows a set of input trials against a particular target.

As introduced in the previous chapter (Section 3.8, p. 39) this experiment yields at least two forms of error, the model error, consisting of the vector from the target after model optimization, which can explain the difficulty of search in the model error space, or the adequacy of a particular search method; and the empirical error, the vector from the model to the transformed sound, which can explain potential errors of the model in describing the real transformation in descriptor space.

The targets of the control experiment consisted of spectral centroid and std. deviation, for which numerical results are reported below. Shortly after the results below, using formulations in 4.5.1, a larger set of spectral moments were also used as target in a more general spectral moment pursuit.

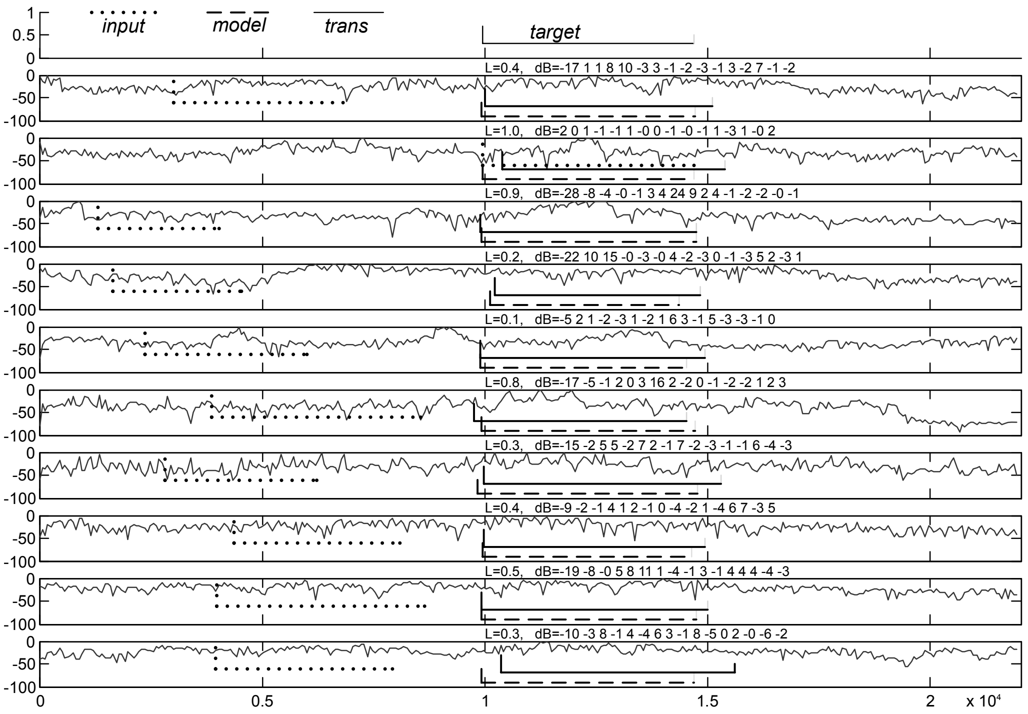

Figure 4.5: One of the targets from the preliminary transformation experiment. 10 input sounds are transformed to match a target, shown in the first row. Each bar represents a descriptor vector, where the left notch is the centroid in Hz, and the length is the std. dev. Descriptors for input sound are shown in the dotted bars, optimized model descriptors in the dashed bars, and descriptors after transformation in solid bars, against the transformed magnitude spectra in dB, with the transformation parameters L and h_j (in dB) overlaid on the sound.

4.8 Preliminary Results

In developing our models of resampling and equalization, the accuracy of the predictive models were tested, starting over one sound, then generalizing to our small 10 sound database. A range of parameters for the resampling model was tested over a set of descriptors to fair accuracy (around 10%), as shown by Figure 4.4.

On the same set of sounds used as targets to each of the input sounds, we optimized the transformation parameters to an average of ±45 Hz (deviation in spectral centroid and spectral std. deviation) in terms of the model.

When these parameters were actually used for transformation and testing, we got an average of about ±190 Hz in spectral centroid and ±140 Hz in standard deviation. (This is a strictly numerical error and should probably be supplemented by perceptual measures in the future).

4.8.1 Optimization Efficiency

Simple penalty terms added to the model made the search take longer, but returned solutions with less extreme parameters. These added two more free parameters to the optimization, effectively creating a trade-off between squared residual error, eq. sharpness, and potential resampling rates.

4.8.2 Qualitative Analysis

After the optimization experiment, the investigator listened to the groups of transformed sounds to qualitatively evaluate the basic rendered result.

By listening within an input sound group, one gets an idea of the range of transformation, as a sound is transformed to hit different targets; and within a target group, one sees how different input sounds are transformed to hit the same target. An impression of similarity within the same target was present but not predominant. This would be due to many variations in the sounds not described by the two dimensions of spectral centroid and standard deviation, which only give the rough shape of the spectral distribution of a sound.

One hopes that by adding other descriptors to the target, other spectral shape coefficients, temporal shape coefficients (along with transformations that affect them), and particularly descriptors that are strongly perceptually grounded, that the future synthesis results will be stronger for within-target similarity.

What can be confirmed from listening is this: that combining transformations along with penalty terms (Sec. 4.6.1) can produce cooperation among them in reaching a target. Using either resampling or equalization, there are more direct and efficient ways to make an input sound more like a target. For example, by computing the spectral envelope of a target, and then adjusting the gains of the input directly to have the same envelope. But this solution can be characterized by its severity of transformation, and its brittleness to subsequent transformations, that may destroy the correspondence with the direct target.

4.9 Further Discussion

Numerical optimization as a solution for this type of problem, when we wish to learn a function from descriptors to parameters or vice versa, has a disadvantage in terms of increased computational effort, as well as sensitivity to the algorithm parameters and the shape of the objective function. The complexity of formulating and computing derivatives is an additional complication in the particular approach of gradient-based optimization.

However, in general, the use of predictive models for control of sound transformations enables a potentially freer choice of effects and descriptors to be coordinated together, versus control based on specific parameter mappings.

4.10 Future Work

The experiments of this chapter have shown that effective, coordinated control of multiple effects is possible, as long as there exist predictive models linking the input descriptors with the output and target descriptors.

In order to build maximally flexible transformation systems, predictive models linking a gamut of general interest effects and descriptors are needed. Building analytic predictive models, as explored in this chapter, is one approach.

Another approach, learning predictive models from data, is perhaps even more attractive, mainly because it does not require the mathematical analysis that the first approach requires. This second approach is explored in the following chapter.

4.11 Acknowledgments

Thanks to MarC Vinyes for suggesting the use of sigmoids used for optimizing the transformation model. Thank you to both MarC and Mike for proofing and suggestions on the notation, and to Mark and Alex for additional comments.

Chapter 5

Learning Predictive Models of Sound Transformations

using Support Vector Regression

This chapter covers the learning experiment and evaluation reported previously in "Predicting Transformed Audio Descriptors: A System Design and Evaluation" [Coleman and Villavicencio, 2010]. The previous paper is revised and expanded, and an additional section about new proposed methods is added.

5.1 Abstract

Predictive models (of the form of Sec. 3.3), which predict changes to perceptually relevant properties of transformed sounds, can be used to enable descriptor-driven control of sound transformations (as shown in the previous chapter). But rather than deriving these models mathematically, it should be possible to learn these models from data. This chapter documents such an attempt. In this study, spectral descriptors of a limited class of sounds under the resampling transformation were modeled with Support Vector Regression (SVR). The accuracy of the predictions is reported, with an emphasis on performance as a function of model hyperparameters. On a cross validation set of resampled inputs, the statistical model predicted an output filter bank to some degree, yet was less accurate than a comparable analytically derived model.

5.2 Introduction

Descriptor-driven control is a strategy that can be applied to sound synthesis as well as to transformations (sound effects), wherein parameters for the effects are selected that steer the output towards desired target descriptors.

Several works [Hoffman and Cook, 2007; Yee-King and Roth, 2008] (also mentioned in Sec. 2.2.1.1) in control of parametric synthesis by descriptors have focused on the optimization of parameters, using a trial and error approach. This involves an audio synthesis/transformation followed by analysis of the result directly in the search loop, which may require considerable computational resources.

By modeling relationships between parameters and descriptors, the need to synthesize/transform candidate parameters can be avoided. A type of model that we refer to as "predictive" predicts output descriptors with respect to input parameters. For example, the system of [Caetano and Rodet, 2009] uses evolutionary search to find a path of spectral envelopes that is smooth in terms of spectral moments, and can be seen as employing predictive models. A second type of "selective" model maps target descriptors directly to control parameters. For example, in the system of [Groux and Verschure, 2008] target descriptors fundamental frequency and loudness are mapped to control parameters of an additive synthesizer using SVR machines. This work deals with predictive rather than selective models (for explanations of model types, see Ch. 3).

As shown in Chapter 4, given a predictive model of a transformation, numerical programming techniques can be used to find acceptable parameters for descriptor based control. If an analytical predictive model can be derived for our transformation, then the model can be used for such a purpose. However, if such a model would be too complex to derive, another solution would be to learn a statistical model from examples, which is the focus of the experiment of this chapter.

Therefore, it is proposed to predict descriptors of the outputs of audio transformations as a function of their input descriptors and control parameters, or $f(\vec{d_{in}}, \vec{p}) = \vec{d_{tr}}$, with $\vec{d_{in}}$, the input descriptor vector, $\vec{p}$ the parameter vector, and $\vec{d_{tr}}$ the output descriptor vector (as discussed in Sec. 3.3).

In this work, the resampling transformation was modeled as a statistical black box, from input and output descriptors and the transformation parameter. To this end, a database of signals with certain characteristics (noisy, stationary, completely described by spectral features) was generated, and modeled with Support Vector Regression, after which the accuracy of the predictions was measured.

This chapter is structured as follows; in Section 5.3, the objects being modeled and the modeling tools are presented. In Section 5.4, the specific experimental procedure is described. In Section 5.5, the results are reported and discussed, examining the effects of model hyperparameters on the predictions and their accuracies. In Section 5.6, conclusions for this study are given, and in Section 5.7, an alternate approach to learning predictive models is proposed for future experiments.

5.3 SVR Prediction of Descriptors

5.3.1 Average Magnitude of Bands (AMB)

As descriptors, a rough measure of the spectrum obtained by averaging the magnitude values from a set of DFT bins is used. This simple measure was used, partly to be able to compare learned models with an analytic model, based on the known changes to the resampled spectrum (see next section).

The descriptor for the j^{th} frequency band is as follows:

$$\text{AvgMagBand}(j) = \frac{1}{\sharp K_j} \sum_{k \in K_j} |X_k|, \tag{5.1}$$

where K_j is the set of DFT bins for band j, $\sharp K_j$ is the number of such bins, defined over frequency bands $j = 1 ... B$.

These descriptors are linear with respect to gain (which facilitates the analytic model of Sec. 5.3.2.1), and should be adequate for describing noisy signals. Like the spectral moment descriptors (used in Ch. 4), they are sums of spectral magnitudes, but they are not normalized as moments are, and are local, rather than global spectral features.

In this chapter, they are reported in units of nAMB, normalized by the maximum band value (7.30×10^{-3} AMB) in the input database.

5.3.2 Resampling Transformation

Resampling is a frequently used audio transformation in which the signal speed is adjusted by a length factor L (relative to a fixed sampling rate). This also scales the frequency content, changing the pitch by $-\log_2 L$ octaves.

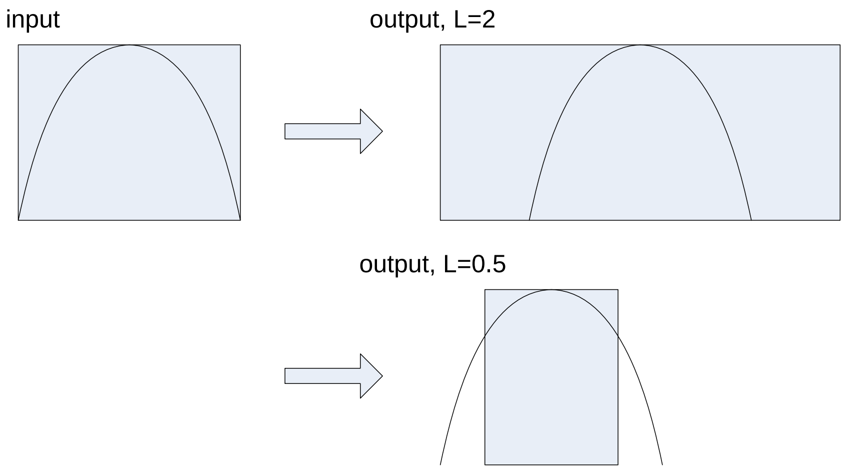

Figure 5.1: Predicting the descriptors of single resampled frames.

Changes brought about by resampling are deterministic and known [Smith, 2002, 2007], but are nonlinear in terms of the parameter with respect to a filter bank representation, making resampling an interesting test case for statistical learning.

A typical descriptor extraction process divides an input signal into uniform-size analysis frames (overlapping or non-overlapping). Resampling changes the length of its input, and as such, either the number or spacing of frames in analysis must change, i.e. under a fixed analysis frame rate, there will be different numbers of input and output frames. Hence, input and output frames can no longer be directly compared.

One approach to predicting descriptors under varying length would be to restrict the input to single centered frames, and the resampling factor L from 0.5 to 2. In that case, part of the input frame content might not be present in the output frame, or vice versa (as shown in Figure 5.1). Another approach is to predict descriptors of frames in aggregate (e.g. mean) rather than for each frame (shown in Figure 5.2). This is the approach chosen for this study; input sounds are stationary and mean output descriptors are predicted from mean input descriptors.

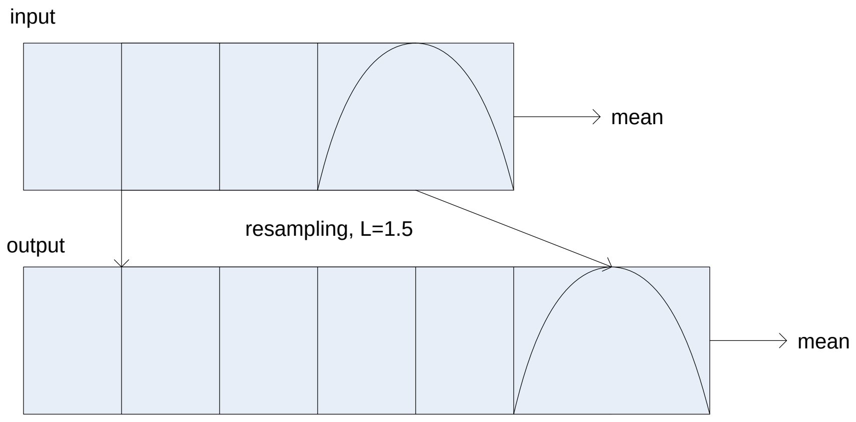

Figure 5.2: Predicting the mean descriptors of resampled frames.

5.3.2.1 Analytic Model

An analytic model for resampling (for comparing with learned models) can be derived based on the observation that resampling is equivalent to spectral interpolation [Smith, 2007]:

$$f(f\vec{b}_{in}, L) = L^{\alpha} \cdot \text{INTERP}(f\vec{b}_{in}, L) \tag{5.2}$$

and gains per band are scaled according to the change in length, and α is a constant that minimizes error on the training database ($\alpha \approx 0.5$). However, the ratio of mean output to mean input seems to vary quite a bit by L and instance (between 0.8 and 1.4), so this could be improved.

5.3.3 SVRs for Descriptor Prediction

Support Vector Machines are a supervised learning technique frequently used for classification, and their extension to learning real-valued functions is known as Support Vector Regression (SVR) [Vapnik et al., 1996; Smola and Schölkopf, 2003]. Both techniques rely on a formulation of linear regression generalized by a mapping into a higher dimensional space (the kernel trick), which allows for nonlinear decision boundaries and regression functions. A model is selected by solving a quadratic programming problem intended to minimize the complexity of the model, in order to counteract overfitting.

The approach of this study was to pick one class of general models and tune it as well as possible, in order to estimate the potential benefit of such an approach in applications. SVR machines were picked because they performed well in initial informal tests on the database.

SVR models have the following functional form [—, 2003]:

$$f(x) = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) k(x_i, x) + b, \tag{5.3}$$

where x is the query point, x_i is the i^{th} input pattern, k is the kernel function providing the potential nonlinearity to the model, α_i and α_i^* are positive and negative weights for that input pattern, and b is a bias term computed once the weights are already established.

In order to train a given model, a subset of the input patterns, known as support vectors, are selected and weighted by an optimization process. This process has two objectives, one being that all the predicted output labels should be within a certain error threshold ϵ of the true output labels (if possible), the other being that the function be as simple (in a particular sense) as possible.

In the simplified, explanatory case in which models are linear functions of the form $f(x) = \langle w, x \rangle + b$, this optimization has the following form [—, 2003]:

$$\begin{aligned} &\min_{w} \quad \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{l} (\xi_i + \xi_i^*), \\ &\text{such that} \quad \begin{cases} y_i - \langle w, x_i \rangle - b & \leq \epsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i & \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq 0 \end{cases} \end{aligned} \tag{5.4}$$

in which ξ_i and ξ_i^* are “slack variables” that account for deviations of the prediction outside of the error threshold, and C is a complexity parameter that measures the trade-off between those deviations and the measure of complexity $\|w\|^2$ of the function.

For the nonlinear case, there are additional complexities, e.g. the optimization problem is not typically solved in the form above, but in some other form, the “dual form”, characterized by an implicit representation of the regression function (as a sum of kernels) and its optimization in terms of the input pattern weights (α_i and α_i^*, as seen in Eq. 5.3), rather than a parameteric representation of the function itself. For a full explanation, see [Smola and Schölkopf, 2003].

As suggested in a practical guide to SVMs [Hsu et al., 2003], the Radial Basis Function (RBF) kernel was used. This kernel maps examples to a space corresponding to their distance from the support vectors, having the following form [—, 2003]:

$$k(x,y) = e^{-\gamma\|x-y\|^2}. \tag{5.5}$$

Finding an optimal model with SVRs typically consists in training models on the training data, and cross-validating them on other unseen data, in search for the best hyperparameters (parameters of the class of models), e.g. using a grid search. Besides the training size, which is an implicit (and important) hyperparameter, there are three hyperparameters that can affect the model performance. C, the complexity parameter, determines the tradeoff between models that fit the data closely and flatter models (hyperplanes closer to zero). ϵ (precision), determines the width of the insensitive-tube in the error function, and informs the model which scale details are significant to model (although some regression machines use other formulations of precision). The optimal value of this parameter should be related to the inherent noise level in the data [Smola et al., 1998] (and thus sensitive to the scale of the labels).

Finally, any kernel hyperparameters must be optimized as well; in this case γ, the radius of the RBF kernel (a spherical Gaussian function), which determines the selectivity of the model; how many of the support vectors are used to determine the label of the query instance.

Predicting vectors of descriptors (as should the predictive models of this study) entails predicting output descriptors for all filter bands. There exist formulations of SVRs for the vector-valued case [Brudnak, 2006], but implementations are not widespread. Fortunately, aggregating single SVRs for each output are equivalent under some criteria and have been reported to give similar performance [Brudnak, 2006].

One more thing should be clear about the approach: it involved concatentating the two arguments, $\vec{d}_{\text{in}}$, the input descriptor vector, and $\vec{p}$, the parameter vector (in this case, just a single parameter) in order to comprise input patterns for the learning algorithm. This could be described as a homogeneous approach, as parameters $\vec{p}$ and input descriptors $\vec{d}_{\text{in}}$ are treated in the same manner by the learning procedure (even though there are many pragmatic differences in audio processing, between descriptions of input signals and control signals). Perhaps it was a natural approach to learning generic functions with statistical learning methods, but in an upcoming section (Sec. 5.7), potential disadvantages to the homogeneous scheme are examined, and alternate approaches proposed.

5.4 Experimental Procedure

First, a database of 400 input sounds was generated by taking uniform white noise samples of 0.5 seconds each (at 44.1k sampling rate) and filtering them by a spectral distribution (see Figure 5.3) described by a (bell-shaped) Hann window surrounded by stop regions a) covering a certain width of frequency from 5% to 90% (20 values) b) with the non-stop section shifted to different positions, from hard left to hard right (20 values). Once generated, input sounds were randomly partitioned into two sets, a training set and a cross validation set (cross) with an 80% training split.

The training and cross sets are separated by input sounds, in order that queries are over new sounds (other than sounds in the training set) in the test condition. In this way, generalization over new input sounds is being tested by this evaluation.

5.4.1 Learning Database

Next, each input sound was resampled under different resampling factors L, according to a uniformly-spaced grid of parameters with 41 values of $log_2 L$ i.e. $(-1, -.95, .., 0, .., .95, 1)$ or between 0.5 and 2 in terms of L. From each of the input and output sounds the magnitude spectrum of all frames was averaged, dropping silent frames at the edges, using an Hann window of 2048 points, an overlap factor of 2, and a zero-padding factor of 4. Then, from each time-averaged spectrum, a magnitude average was taken over 16 uniformly-spaced (in Hz) frequency bands. These features were exported as training and test databases to the Weka ARFF format [Garner, 1995] for the learning tasks: to build and to evaluate LibSVM models [Chang and Lin, 2011].

To summarize, there were two databases, train and cross, which consisted of 320 and 80 input sounds. Each input sound had 41 corresponding output sounds, each of which described by a transformation parameter $-\log_2 L$, an input envelope of 16 bands, and an output envelope of 16 bands. In total, the data consisted of 541,200 real numbers, which should sum to 2 MB when represented in 32-bit floating point precision, although the Matlab data file (containing additional features such as MFCCs) was 89.9 MB, and

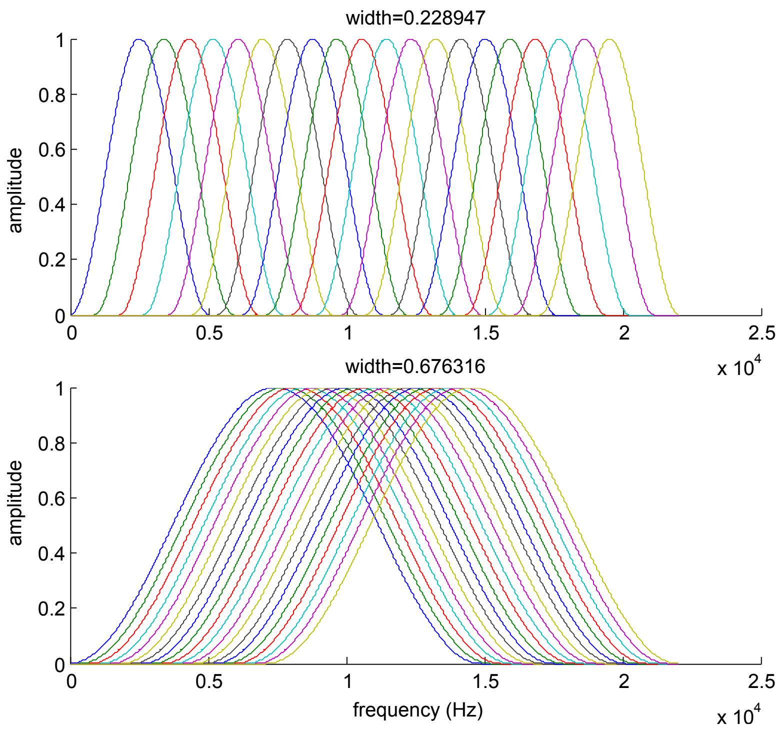

Figure 5.3: Generated noise envelopes for input sounds for two different spectral widths.

the ARFF format database (containing just the filter bank representation) was 15.5 MB due to the text format.

5.4.2 Selection of Model Hyperparameters

Model hyperparameters were evaluated by exhaustive search over a coarse grid, by testing the trained model against the cross validation set. For simplicity, all features were used as inputs, and a global set of model hyperparameters was used, although it might give better performance to different model parameters for each band.

Due to the nature of the resampling function, in which input energy ends up in different output bands depending on the parameter L, it was thought that eliminating certain input descriptors would be counterproductive; although perhaps feature transformation would have helped.

A grid of parameters (such as in Figure 5.4) for training size, C, ϵ, and

γ was iterated over. For each set of model parameters, the corresponding predictions on the cross validation examples, and a subset of the training examples were stored. Once generated, the predictions were compared with the actual transformed descriptors in order to generate error statistics. The performance of models was described, and the selection between models made, in terms of the mean of absolute residual ($\overline{|r|}$) over all bands.

5.5 Results

The results here reported were not on a true test set, but rather on a simple cross-validation, as they result from using the cross validation set to select a model (by tuning hyperparameters). For complete generalization, the test set would not be used for model selection, but would be a new third dataset.

For the model with best performance, the transformed descriptors of the cross validation set were predicted with a mean residual per band of 0.0326 nAMB which is around 16.2% of the mean output value (0.206 nAMB). This performance was achieved using a training set of 12k input-param-output tuples, complexity C of 3.162, kernel γ of 4.642, and a model precision ϵ of $1{\times}10^{-6}$. By contrast, the analytic model in Section 5.3.2.1 achieves an average residual per band of 0.0103 nAMB, which is around 5.14% of the mean output value.

At first, the importance of the ϵ parameter was overlooked, causing the model to fail to attend to the finer details of the regression. Using typical values of the other parameters we found a more suitable value that improved the performance by an order of magnitude (effect shown in Figure 5.8). (Normalizing the output data, as is often recommended, would have also had the same effect as fixing this parameter).

Next, a broad search for the other parameters with a smaller-than-maximum training size was conducted. As SVM training time is seemed to be quadratic with regards to training size, this allowed searching a large hyperparameter grid in a reasonable amount of time, after which a small local grid search was performed on the larger training size.

Figure 5.4 depicts the major trends in the coarse grid search. A roughly paraboloid shape can be seen, as well as a trend pointing towards greater γ and lesser C in on the test set. For this training size, it appears that the best solutions could be found in $\log_{10}\gamma$: [0, 2], $\log_{10}C$: [-3, 0], or overlapping with the corner facing us in the figure (region partially shown).

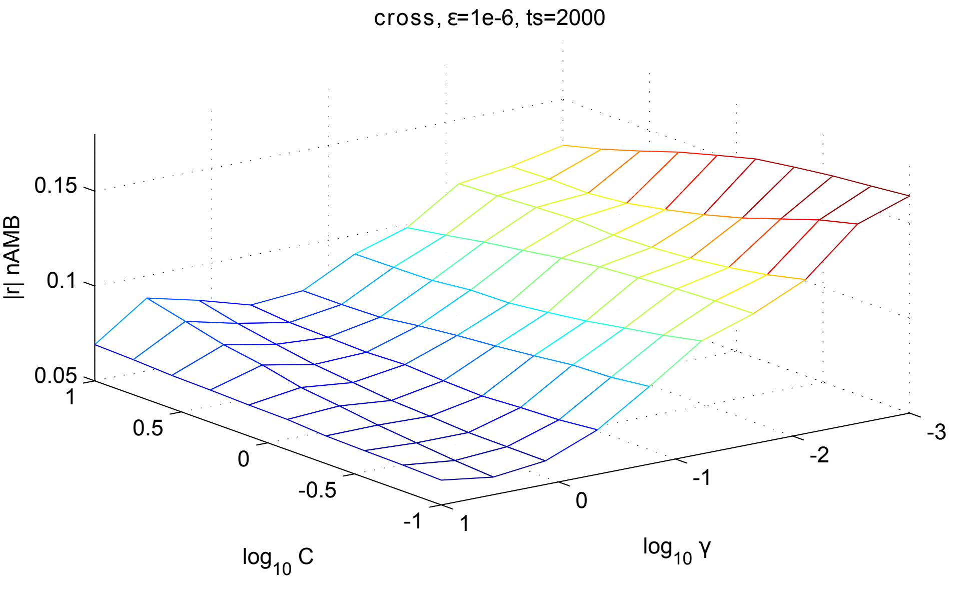

Figure 5.4: Coarse grid search (small training size) over γ and C hyperparameters

Along an axis of constant C, we see individual minima with respect to γ that shift slightly according to C. In the formulation of the RBF kernel used (Eq. 5.5), γ is inverse to the radial distance of the kernel, so that smaller γ means a less exclusive kernel that covers many of the training examples and that larger γ means a more exclusive kernel that covers only a few close training examples.

By contrast, the trend with respect to C is more subtle and more sensitive to the other parameters such as γ. Nonetheless, when C is less than the optimal value it corresponds to underfitting the data, and when C is greater than the optimal value it correspond to overfitting the data. C seems to be sensitive to the training size, while the optimal value of γ seems less so.

Looking at optimal hyperparameter values for different training sizes (in Table 5.1), it can be seen that optimal value of γ is relatively stable with changing training size. By contrast, the optimal value of C increases with the increase in training size (more data allows a more complex and less smooth model).

To see if there was sufficient data for learning to converge, the performance on the cross validation and training sets with respect to the amount of

trainsize	best ($\overline{\lvert r\rvert}$ nAMB)	best γ	best C
12000	0.0326	4.6	3.16
8000	0.0369	4.6	1
4000	0.0460	4.6	0.316
2000	0.0594	3.59	-1*

Table 5.1: Optimal values of γ and C for different training sizes. *Minimum for coarse grid was on edge, hence may not be optimum.

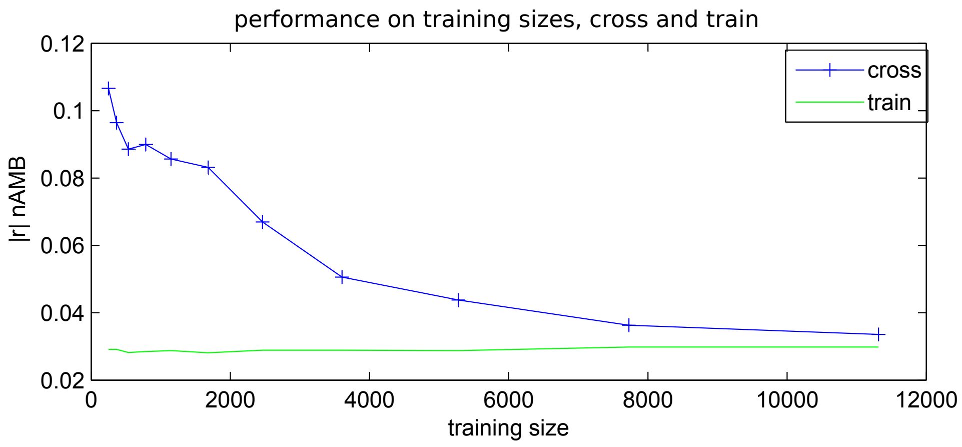

Figure 5.5: System performance with respect to training size, on training data (green) and cross-validation data (blue).

training data was examined, given in Figure 5.5. The performance on the cross-validation set does not completely flatten out, but as training data increases, cross performance approaches training performance quite closely.

5.5.1 On the Model Hyperparameters

In the following section, the effect of each model parameters on the vector-valued prediction result is illustrated, showing how changes in those parameters affect the predictions produced. This is illustrated with triads of plots: the true values after transformation, a predictive model with suboptimal parameters (we attend to this to see the effect of the parameters), and a model with "good" parameters. In each plot, the colored lines show the descriptor vector (filter bank) at different values of the resampling parame-

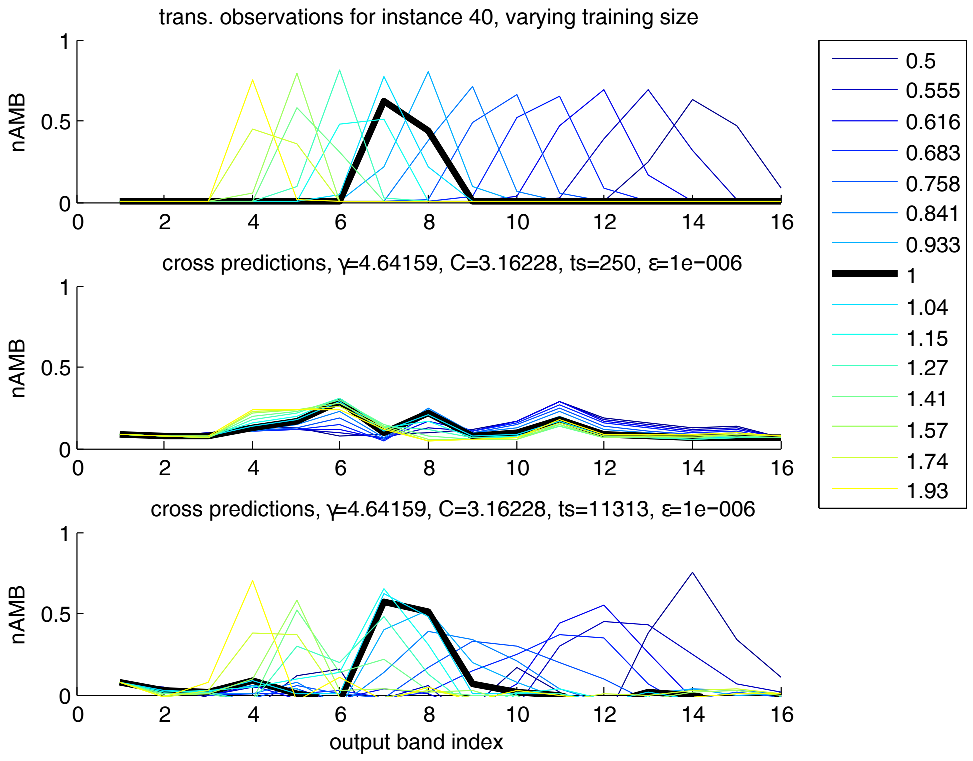

Figure 5.6: Comparison of instance predictions with regard to training size; cross-validation observations of a particular input sound (top) at different resampling rates (by color), versus predictions using a small training size (middle), and with the largest training size (bottom).

ter between 0.5 and 2. The bold line is the transformation with resampling factor L=1, also equivalent to the input.

Training Size For an insufficient training size, it seems that test examples don't converge, i.e. there are insufficient training patterns within the neighborhood of the RBF kernel for a particular query point, so the regression does not reach the level of the true function. See Figure 5.6 (middle plot shows small training size).

γ (Gaussian kernel inverse width) As stated earlier, small γ means large promiscuous kernel (many training patterns are active for a query pattern) and large γ means a small and selective kernel (few training patterns are active for a query pattern). As such, overly small γ will result in overly smooth approximations. By contrast, an overly large and selective γ will result in not enough training patterns for a query, so some examples will fail to converge. See Figure 5.7 (middle plot shows overly smooth approximation caused by too small γ).

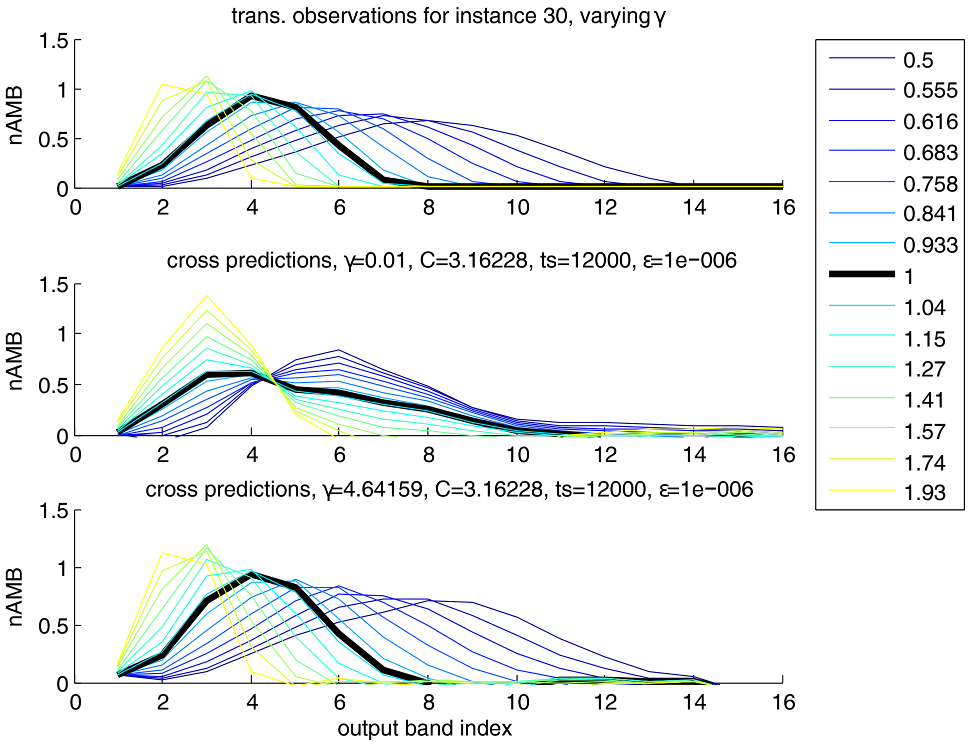

Figure 5.7: Comparison of instance predictions with regard to γ; cross-validation observations of a particular input sound (top) at different resampling rates (by color), versus predictions using non-optimal γ (middle), and with the optimal γ (bottom).

ϵ (Precision) controls the width of insensitive-tube for the error function. Thus, if it is too large, the regression will ignore the smaller details in the target function. By contrast, if it is too small, it will expend lots of computational effort overfitting to noise. See Figure 5.8 (bottom plot shows predictions with overly large ϵ, which fails to attend to details of the target function).

C (Complexity) controls the trade-off between overall fitness of the regression and flatness of the model. In the example given, the regression doesn't

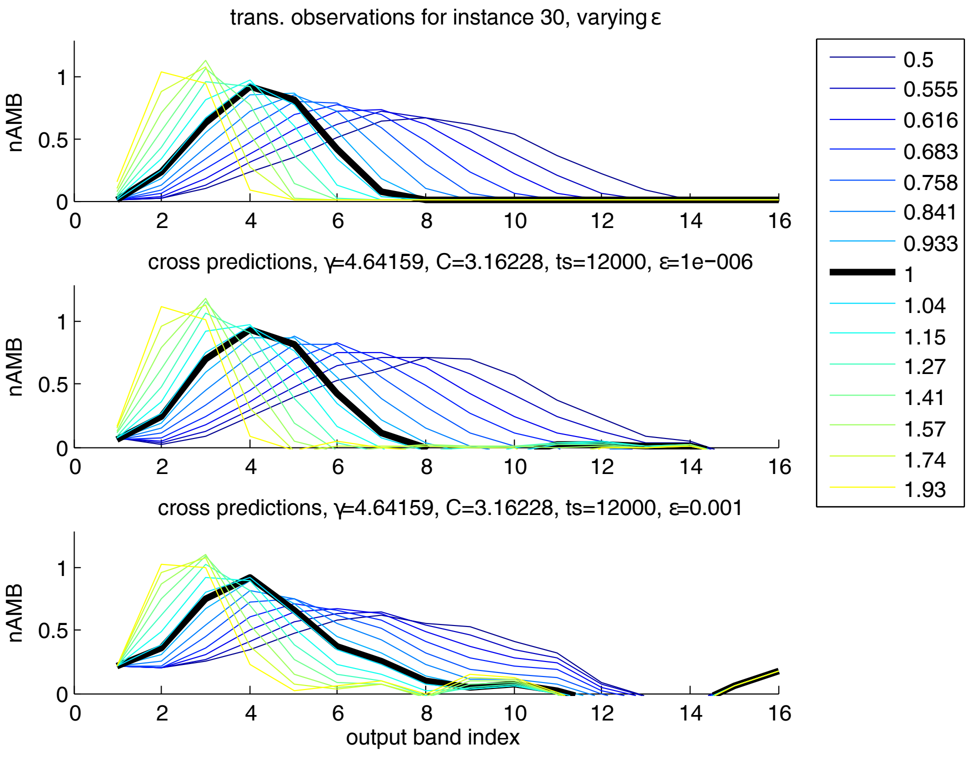

Figure 5.8: Comparison of instance predictions with regard to ϵ; cross-validation observations of a particular input sound (top) at different resampling rates (by color), versus predictions using optimal ϵ (middle), and with a non-optimal ϵ (bottom).

completely converge, but we see that when C is less than optimal, the regression stays lower than the target function. See Figure 5.9 (middle plot shows predictions with too small C).

5.5.2 Distribution of Test Error

By Input Sample In the discussions of model parameters in the previous section, (Section 5.5.1), it can be seen that despite hyperparameter optimization, some of the cross-validation instances did not seem to "converge". By averaging performance for each of the test input sounds, it can be shown that different inputs have vastly different average performance. Examining the sorted performance curve (see Figure 5.10) shows two sub-

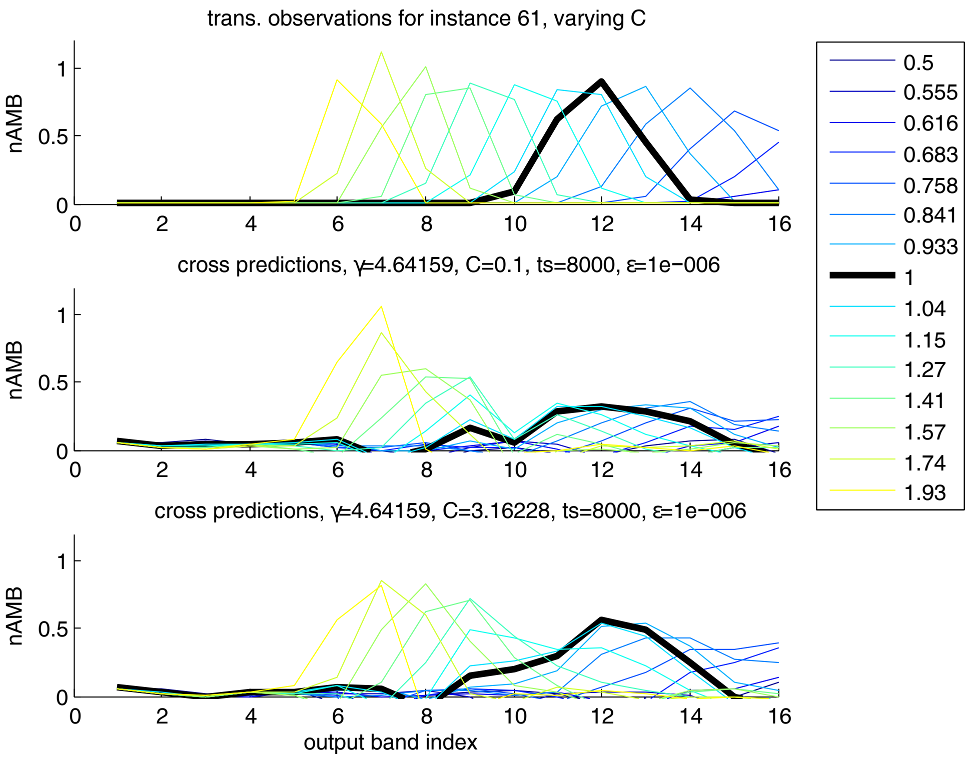

Figure 5.9: Comparison of instance predictions with regard to C; cross-validation observations of a particular input sound (top) at different resampling rates (by color), versus predictions using non-optimal C (middle), and with a optimal C (bottom).

sets of cross-validation input sounds with respect to performance divided by an inflection point: examples 1-25 (the first 30%) contribute more average error and error variance, while a broad section of about 70% of examples has lower and relatively uniform errors. These examples seem not to converge (not yet known why), and contribute a proportionally higher amount of error to the approximation (that perhaps merits further investigation).

By Transformation Parameter (L) The average performance by resampling parameter value is examined (see Figure 5.11), and is relatively flat (varies between 0.027 and 0.041 nAMB) compared to the difference among cross input sounds as above. This could be because the training database had an equal amount of examples for each parameter value (cross distribu-

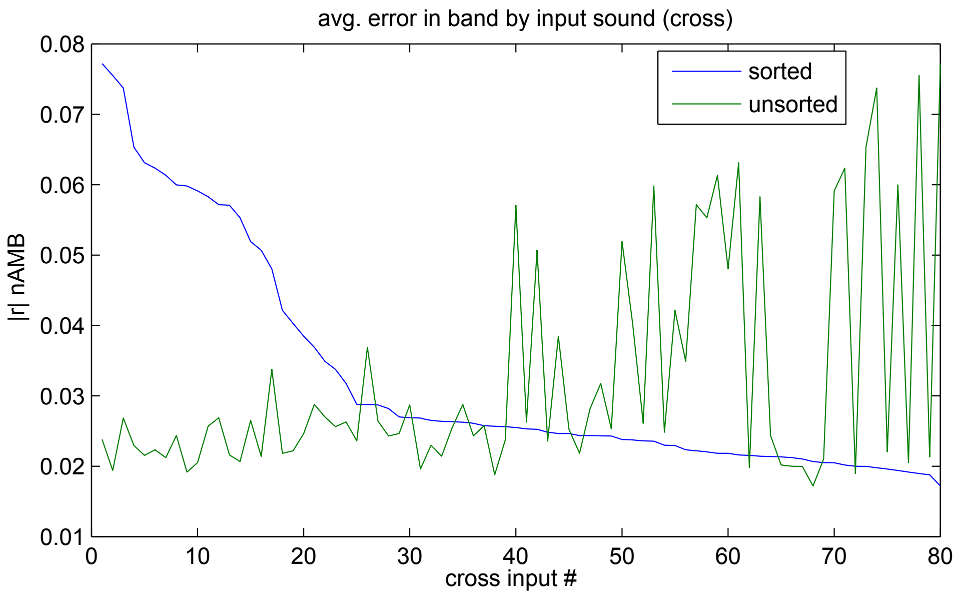

Figure 5.10: Average error per band of cross-validation input sounds, unsorted (green) and sorted by error (blue).

tion equal to training distribution).

5.6 Conclusions

In the experiments of this chapter, a statistical model learned from data produced by the transformation (as a black box), was able to achieve an error rate within a factor 3 of an analytic model, informed by knowledge of the transformation. This result suggests, at least as a proof-of-concept, that statistical models may be used to build predictive models of transformations without resorting to the compexity of deriving those models analytically (as in the previous chapter).

The model selection procedure, mainly tuning of the hyperparameters, was documented, allowing common failure modes of SVR models in terms of hyperparameters to be examined. Hopefully, such experience will assist in developing future statistical predictive models.

The accuracy of the statistical model compared to the simple analytic model

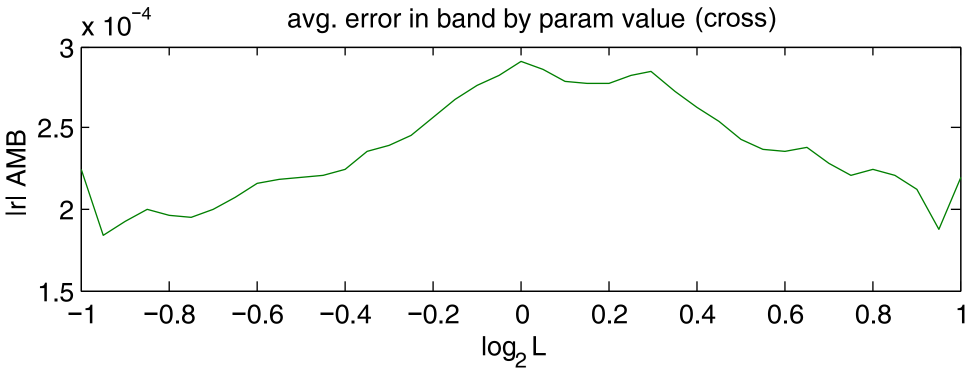

Figure 5.11: Average cross-validation error by transformation parameter (unnormalized units).

was somewhat disappointing, given several factors: the size of the database and the time taken to train and tune the larger models (around 1 day on the personal computer used), and the additional effort needed to pool the output predictions from the scalar-valued models.

The final models, which used a significant number of the input patterns/ support vectors, and had to duplicate input patterns, due to the pooling of scalar-valued model outputs, were unwieldy. Their hypothetical cost would also include that in parameter optimization in computing model gradients, as in the previous chapter, if these models were to be used for automatic control.

Given that a maximally general approach was used, based on smooth regression (approaches that are typically subject to the curse of dimensionality, see Appendix C.1), perhaps the dimensionality of the input space makes the problem too difficult. As the curse of dimensionality predicts that the number of examples needed scales exponentially with the number of input dimensions, perhaps there were not enough training examples to guarantee better than the lackluster performance seen.

The author believes that, perhaps by a more specific approach, incorporating more assumptions relevant to audio transformations (but still applying to subclasses of effects), that more accurate models should hopefully be attainable with smaller training sets and small model sizes.

5.7 Learning Sparse Linear Models of Transformations: a proposed approach

To recap, in the learning experiment described in this chapter, predictive models of a sound transformation, resampling (corresponding to pitch and speed change), were learned using a general nonlinear statistical regression method, Support Vector Regression (SVR) using Radial Basis Function (RBF) kernels. To produce input labels for the statistical learning, the input descriptors $\vec{d}_{\text{in}}$ and transformation parameter L were concatenated, in what we call a homogeneous approach. A nonlinear model was needed because the output descriptors, consisting of frequency filter banks, cannot be described as a linear function with respect to the parameter L.

This SVR model was able to learn the concept to some degree, but the accuracy of the model was significantly less than the derived model, despite training on 12k examples and needing around a day to train. These are high practical costs and relatively middling predictive power, stemming from the use of this maximally general modeling approach. Alternatively, perhaps it should be possible to use some assumptions relevant to audio effects in order to reduce the complexity of the problem, and thereby learn models with higher accuracy and less training examples.

Two properties of some audio transformations were not taken into account, that might admit less complex models, and that might also be applicable to a wide array of transformations, for example, spectral transformations.

The first property is linearity: there are transformations that are linear in terms of the spectrum and power spectrum. For example, resampling (the perennial effect of this thesis) can be shown to be linear with respect to the input signal, by applying the Scaling Theorem for continuous signals (an idealized model of resampling, earlier stated as Eq. 4.6, p. 49), along with the definition of SCALE, which yields:

$$\textsc{Scale}_{\lambda}(\alpha x + \beta y) \longleftrightarrow \alpha|\lambda|\,\textsc{Scale}_{(1/\lambda)}(X) + \beta|\lambda|\,\textsc{Scale}_{(1/\lambda)}(Y), \quad (5.6)$$

where λ is the scaling parameter.

That is, given a set of input signals and output signals, resampled with a fixed parameter λ, one could simulate a resampling transformation on any new input signal that was a weighted sum of the input signals, simply by weighting and summing the corresponding output signals using the same input weights.

Linearity with respect to input signals (for fixed transformation parameters) should apply not only to resampling and frequency warping, but also to other effects classes, such as reverbs and linear filters.

The linearity above applying to ideal scaling is exact, but an approximate linearity with regard to a power spectral representations (in which phase is discarded, similar to forthcoming in Section 6.3.3) could also be used.

This property alone should be enough to reduce the complexity of the statistical learning task, because, instead of having approximate functions having D_{in} (number of input descriptors) + P (number of parameters) dimensions, it should be possible to decompose the problem into a number of subproblems (one for each input $\times$ output descriptor) where the input dimension is only $1 + P$. That is, with an effect that is linear in the input, it should be possible to approximate independently, the individual responses to some set of basis signals like Fourier basis, that can be used to describe any input signal of interest.

The second property is sparsity: when a single input bin is active, the number of active output bins is few, so that in the matrix describing the function (in the case of a linear function) there are few non-zero entries.

Several examples can be shown where these properties could allow learning models with low-complexity: two functions that are linear in the input, and one that is non-linear in the input. In each case we illustrate a kind of linear response (power response) to the various transformations, by showing the output energy generated by a sinusoid at each input frequency.

Figure 5.12 shows this power response, computed for resampling with different parameters L, and composited. Figure 5.13 shows the high frequency power response for $L = 1.2$, indicating some aliasing in the implementation. This is a case in which a learned model could incorporate behavior not seen in the simple analytical model of resampling, and thus be more accurate.

Figure 5.14 shows the power response for single-side band (SSB) modulation (a modulation effect that shifts frequency content uniformly with respect to a frequency shift parameter f; see [Dutilleux and Zölzer, 2002a] for details) at different frequency shifts.

Since these effects are just shifting around energy from the input in some way, the power response here will be linear. The lines in the output spectrum are wider than a single bin, probably due to spectral leakage in the frequency analysis, not due to the transformations.

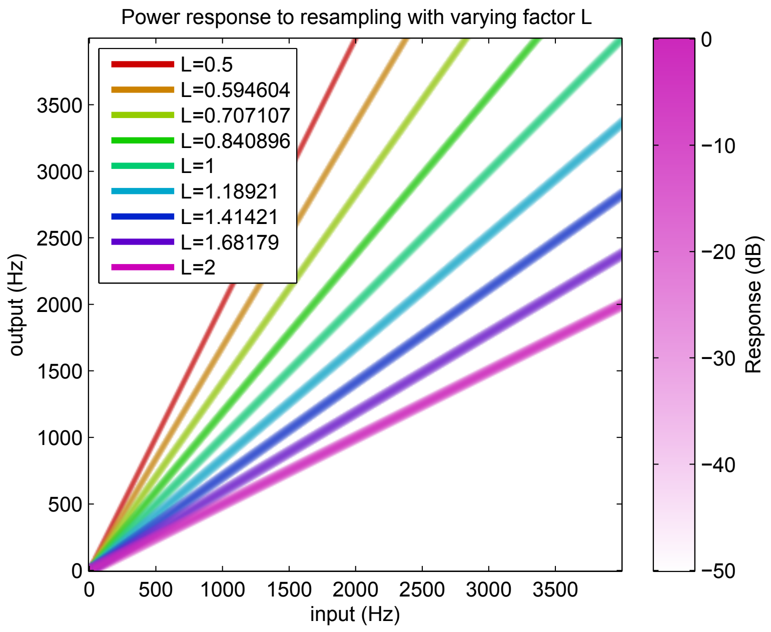

Figure 5.12: Power response to a sinusoid with different resampling factors.

Next, an effect known to be non-linear in the input power spectrum can be examined: using an exponential distortion effect [Bendiksen, 1997 as cited by Dutilleux and Zölzer, 2002b]:

$$f(x) = \frac{x}{|x|}\left(1 - e^{x^2/|x|}\right). \tag{5.7}$$

Figure 5.15 shows the distortion effect with gain 1, showing just two prominent odd harmonics. Figure 5.16 shows the distortion effect with gain 50, showing many odd harmonics, and high frequency noise that looks harmonically related due to the linear structures that seem to be present.

In the case of this distortion effect, the power response representation will not capture second-order effects (when energy in two frequencies combine nonlinearly), but perhaps it will be a good first approximation.

Of course, the representations shown are discrete representations, input bin by output bin, but it should be possible to recover the (sparse and linear) continuous structure present in those plots.

Perhaps the following two-stage approach, to learning models of transformations that are linear or non-linear with respect to their input power spectra,

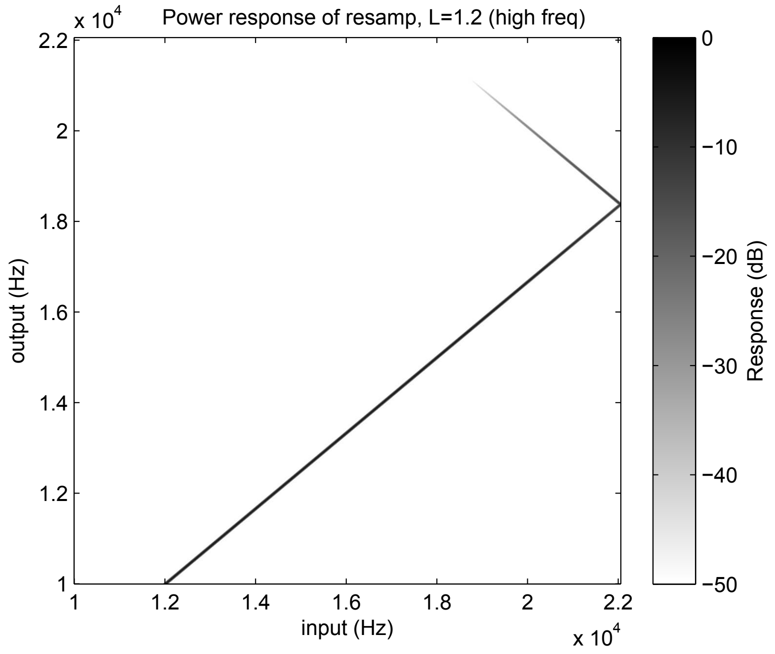

Figure 5.13: Power response for one resampling factor at higher frequencies, showing aliasing.

can be used. In the first stage, the power spectral response is collected as above, at all nonlinear parameters, including the input level, if the transformation is nonlinear in the input.

In the second stage, this variation in power response with respect to the non-linear parameters (and/or level) is modeled. At least two approaches would be possible. One approach would be to recover the linear structures that seem to be present, and build a sparse model consisting of weighted lines. To detect the lines, one could interpret the power spectral response as an image, and use the Hough transform [Duda and Hart, 1972]. Or, one could threshold the data, use a clustering algorithm that detects separate line segments [Thomas, 2011], and fit lines to the clusters as usual. Once the lines are detected, the gain along the lines could be fit parametrically, e.g. a piecewise linear function in input and output frequency.

Another approach would be to learn smooth nonlinear functions with two frequency parameters (input and output) and the small set of nonlinear parameters (including level), e.g. with general nonlinear models, i.e. as before,

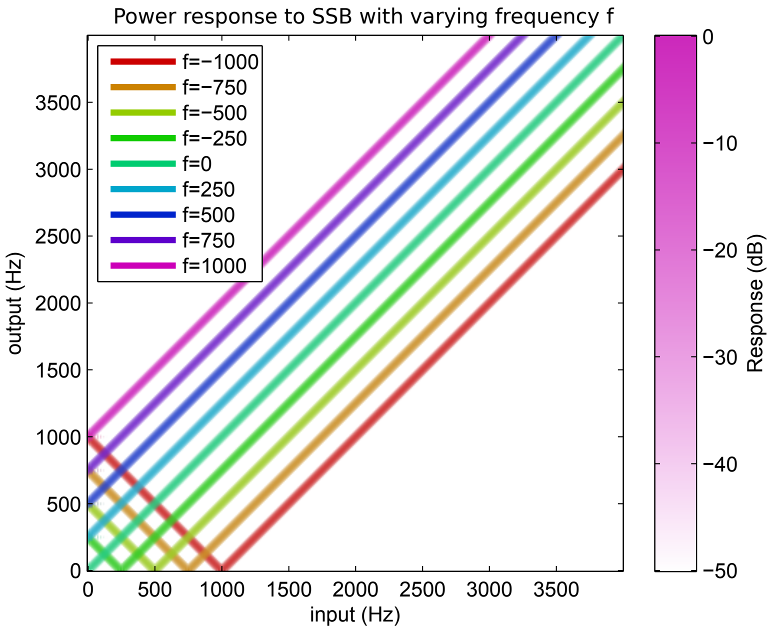

Figure 5.14: Power response to a sinusoid with different single-sideband modulation frequencies.

but with a much smaller input dimension.

Perhaps a third heterogeneous approach would be one with two stages. For each different value of the fixed nonlinear parameters, a linear model can be inferred. Then, a nonlinear mapping between the nonlinear parameters and the linear models (matrices) can be found.

A general form of the heterogeneous approach, where linearity is expected in the input parameters, could be written as follows:

$$f_{\text{het}}(\vec{d}_{\text{in}}, \vec{p}_{\text{tr}}) = M(\vec{p}_{\text{tr}})\vec{d}_{\text{in}} \approx \vec{d}_{\text{tr}}. \tag{5.8}$$

By choosing a specific nonlinear model for the matrix function M, for instance a parametric model (rational functions) or a non-parametric model (a sum of radial basis functions, or a hidden layer neural network), the matrix function M could be trained with stochastic gradient descent, the leading approach for training deep neural nets.

Yet another, more specific property that was not exploited in the case of resampling, due to symmetry with respect to the transformation parameter

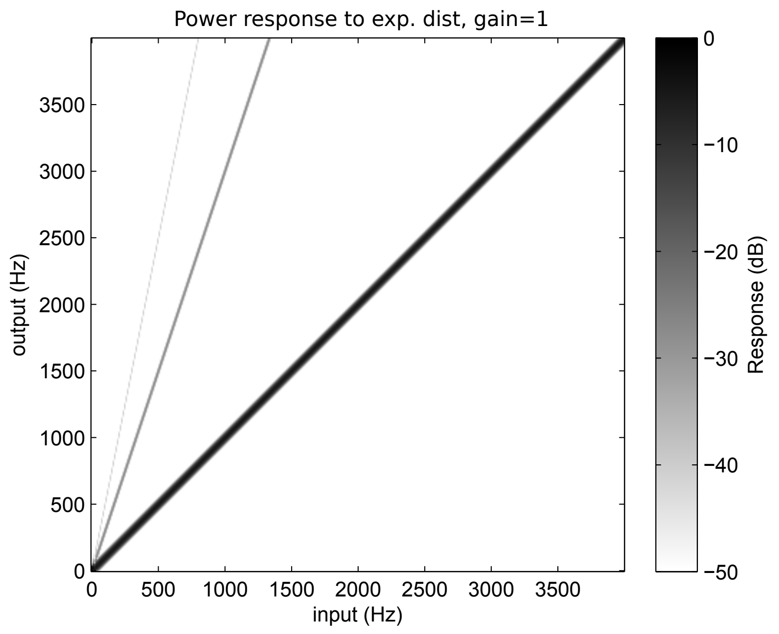

Figure 5.15: Power response to exponential distortion at a low gain.

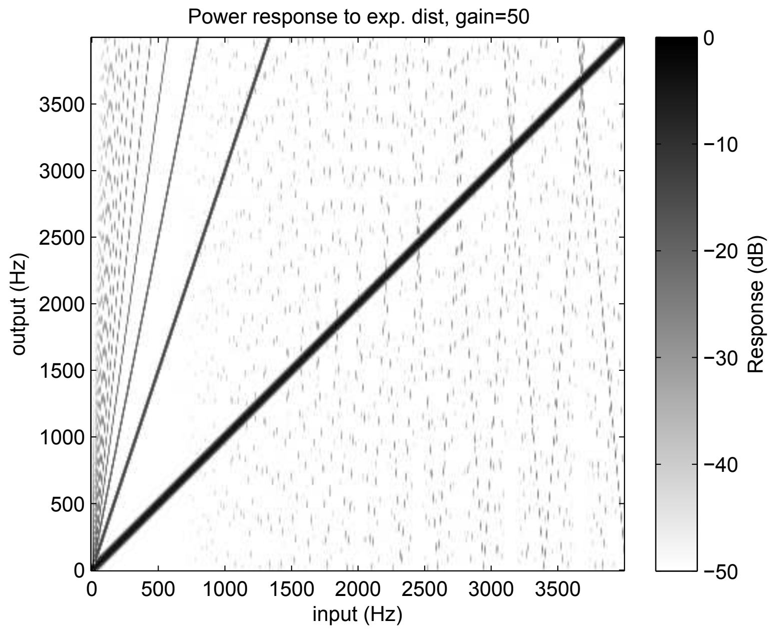

Figure 5.16: Power response to exponential distortion at a high gain.

L, is that the effect of transforming with larger magnitude parameters can be broken down into iteration of smaller magnitude parameters. In the case of continuous signals, this follows from the definition of the SCALE function (Eq. 4.6):

$$\textsc{Scale}_{\alpha\cdot\beta,t}(x) = \textsc{Scale}_{\alpha,t}\left(\textsc{Scale}_{\beta,t}(x)\right) = x\left(\frac{t}{\alpha\cdot\beta}\right). \qquad (5.9)$$

This symmetry could be used to extend a predictive model trained on a limited parameter set to a wider parameter set, or possibly to estimate parameters for a predictive model on the limited from the wider.

Digital signals, however (as previously stated), represent a limited frequency range, and bandlimited interpolation involves low-pass anti-aliasing filters. Therefore, an iterative representation of resampling might overestimate the low-pass effect of iterated anti-aliasing filters, for example. Even so, this might be an additional way to improve the accuracy of the model relative to the number of examples needed, for effects which can be represented iteratively.

In any case, these approaches should decrease the number of input and model parameters, which should theoretically allow for more accurate models with lower numbers of training examples. This is done by replacing the raw input and output features, in which there is an inherent but uninformative level variation, with power responses. This should allow the relevant variation of the transformations, with respect to the input signals and nonlinear parameters, to be captured with fewer examples.

Hopefully, future experiments can prove the utility of such approaches.

5.8 Acknowledgements

Thanks to Esteban Maestre and Ferdinand Fuhrmann for their helpful suggestions on the earlier version of this chapter.

Chapter 6

Creating Audio Mosaics with Mixtures and Transformations

6.1 Abstract

In this chapter, I describe the main engineering artifact of this dissertation: software that produces imitations of a target audio signal by transforming and mixing source audio signals. These imitations are produced by generating controls for a sampling synthesizer, based on a joint analysis of the target and source signals in descriptor space that takes into account the transformation possibilities of the sources.

This joint analysis is computed using several new methods proposed in this chapter. In each method, the output imitates the target signal so that it is maximally similar to the target signal's energy in perceptually motivated filter banks over time, balanced against other criteria such as amount of transformation and different aspects of continuity. Rather than choosing a single transformed source unit to approximate each target unit, the methods of this chapter use mixtures, i.e. sums of pitch-shifted transpositions of the source signals at different sampling positions, in order to better approximate the harmony and timbre (spectral characteristics represented by filter banks) of the target signals. Further refinements to these methods improve the preservation of source character, by favoring sampling of longer continuous source extracts.

Some aspects of earlier versions of this system were previously described in [Coleman et al., 2010] and [Coleman et al., 2011], but the solution methods

have changed (moving to a greedy sparse approximation paradigm), leading to improvements in efficiency and enabling (in a straightforward way) longer continuous source extracts, as mentioned above. This chapter tries to make the system description as complete and up-to-date as possible.

6.2 Introduction

Mosaicing synthesis, in which source sounds are transformed and composited to form an imitation of a target signal, is one possible approach to the problem of audio texture transfer, i.e. how to map the impression of timbre from a sound source or instrument onto an existing sonic or musical context. With mosaicing, we can produce these target imitations, while creatively recycling the source signals of our choice. In Chapter 2, State of the Art, a detailed look at previous work in mosaicing is given.

Classical mosaicing methods generally worked by selecting sequences of individual source segments that best matched the target (along with other criteria such as continuity and diversity), e.g. [Zils and Pachet, 2001]. However, in general, there might not be close matches for all target segments, especially when the set of source segments is small. This might be especially true for an audio texture transfer scenario with the following characteristics: a large descriptor space, covering aspects of harmony and timbre; a relatively small sample of source segments, covering only a small part of the descriptor space. For example, if the source samples were instrumental, they might only cover a certain tonality (certain scales and only a few chords) and certain timbres. The target will most probably fall in areas of the descriptor space where there is no support in the sources.

To make an analogy between notated pitches and harmony descriptors, consider the following scenario: the target is a segment with a simultaneous A minor (a) triad, but source segments are single notes in a pentatonic scale in B major. As given, no segments match well one-to-one from source to target (see Figure 6.1).

There are two natural strategies to multiply the possibilities of expression for a limited set of source segments. One is to support transforming segments (in this case, transposing them) so they can cover a wider set of the descriptor space. The other is to allow mixtures of two or more segments. From the analogy above, one sees how these strategies could work well together: we can take notes from the B major scale, and transpose them

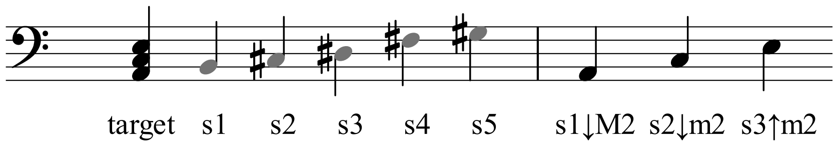

Figure 6.1: A musical example illustrating transposition and mixtures. In the first bar we have the target, a chord, and source notes s1-5. Although there are no direct matches for the target chord, source notes s1-3 can be transposed by small amounts (minor seconds, m2, or major seconds, M2; shown in second bar) and mixed for a close match.

individually to nearby notes in the A minor triad, and mix them to make the simultaneous triad.

If one allows mixtures, a sparse representation is preferred (i.e. using only few active segments), rather than mixing most of all possible segments simultaneously with different weights. This is certainly possible, and even partly works in the case of spectral methods like ours (creating an amusing chorus effect). However, in the chorus of many segments, individual source characteristics are lost. This is the main reason to prefer sparsity, i.e. for clarity. Additionally, when large number of segments are mixed, many of them will be similar; mixing similar segments also creates particular distortions, and violates our synthesis model (see Section 6.3.3). Finally, the computational cost of transforming and sampling many segments at the same time could be very expensive. Sparsity thus has a large practical advantage, especially for interactive and real time applications.

We model a mosaic as a polyphonic sampling process: the mixed signal output of a collection of sampling processes, called tracks in our application, each track having an independent set of control parameters consisting of: source position, tape speed, and gain. (The source material being sampled might be monophonic, like a recording of a single violin, or polyphonic, like that of an ensemble). At each instant, individual tracks can be created, and existing tracks can be continued or destroyed; so that the duration of individual tracks varies. In order to compute the sparse mosaics, we adapt methods from a family of sparse approximation algorithms distinguished for efficiency, the so-called "greedy" algorithms; and furthermore adopt additional heuristics to extend the tracks in time.

Because the track representation consists of groups of signal units (linear chains of units in time belonging to a single track) rather than independent units, it could be considered an example of what is called "structured sparsity" (e.g. [Bach et al., 2012]).

The proposed system is the first to integrate the following elements: a) appropriate perceptual descriptors describing short-time harmony and timbre, b) transforming source sounds, c) mixing those transformed sounds, with d) variable-length sampling tracks that maintain the continuity of sampling and transformation. This combination of elements (a-c) leads to a flexibility of synthesis in the approximating the target, while preserving the character of the source sounds (in particular, d). Processed mosaics produced by the system, when compared to mosaics from other alternate mosaicing algorithms, were competitive in quality, as judged by subjective listening tests (see Chapter 7, Subjective Evaluation of Sound Mosaics using Listening Tests).

This chapter proceeds with the following structure: first, we detail the descriptors used to describe short-time sound segments in Section 6.3, Perceptual Descriptors. In the next section, 6.4, An Objective Function for Mixture Mosaics, we express our preferences for mosaics as a tradeoff between desired qualities such as similarity to the target signal, use of less extreme transformation parameters, or the sparsity, in terms of using less sampling tracks. These preferences take the form of an objective function. Then, the methods are detailed in Section 6.5, Mosaicing Methods based on Greedy Sparse Decompositions. After that, we can see properties of the generated mosaics in Section 6.6. Finally, we will go over some alternate approaches in Section 6.7, and conclude shortly thereafter.

6.3 Perceptual Descriptors

For controlling the proposed synthesis algorithms, we use certain descriptors dealing with harmonic and timbric aspects of short time-slices of sound, hereafter referred to as frames. In general, these descriptors are filter banks; meaning that they separate an input signal into multiple components, in this case, components of overlapping frequency ranges in the spectrum. But rather than explicitly computing these signal components, we estimate the energy of specific filter bands and collect them into descriptor vectors.

Of the two classes of filter banks we use, both are linear functions of the short-time power spectrum, i.e. $f(X) = M\left(|X_k|^2\right)$. That is, if x is a windowed time-domain signal and X is its Fourier transform vector, we form the power spectrum, $\left(|X_k|^2\right)$, by collecting the squared absolute value entries of X at bins k corresponding to positive frequencies. Then each filter bank can be computed by multiplying some matrix M with $\left(|X_k|^2\right)$. In the case of both filter banks, M is non-negative and sparse: the entries of M are all ≥ 0, and for a given filter band, the contribution from most of the frequency indices is zero.

The significance of using linear functions of the power spectrum will be discussed in Section 6.3.3, following two subsections about the specific filter banks.

6.3.1 Chroma Filter Bank

Chroma vectors, also known as Pitch Class Profiles (PCP), are a common descriptor used for estimating musical harmony, first proposed by Fujishima as a component of a chord recognition system [Fujishima, 1999]. A chroma vector gives a rough picture of which musical pitches (without regard to which octave) are present in a given segment. This idea is based on the music theoretic idea of pitch classes; that musical notes of the chromatic scale can be divided into equivalence classes of notes related to each other by octaves. These descriptors are used in many retrieval systems; for example, a music thumbnailing system that identifies song choruses [Bartsch and Wakefield, 2001], a cover song identification system [Serrà et al., 2008], and an interactive browser for selecting short-duration sounds to sample [Coleman, 2007].

Several variants have been proposed. For example, the Harmonic Pitch Class profile (HPCP) descriptor, proposed in [Gómez, 2006], uses the energy of estimated sinusoidal peaks (rather than the power spectrum) as its basis. Additionally, several other strategies are applied to make the harmonic description more accurate: such as using transient detection to ignore noisy frames, estimating the fine-tuning of the reference frequency, a spreading function that distributes energy to neighboring pitch classes (used below), and a weighting function that distributes energy from sinusoids to harmonically related pitch classes. Müller and Evert have also proposed a number of variants [Müller and Ewert, 2011].

In general, it is possible to compute chroma at a resolution greater than a single semitone, in order to capture the harmonic contour and effects in greater detail. For most of our experiments, we used $B_c = 36$ chroma bands (3 bands per semitone).

We estimate the energy c_b in the b^{th} (of B_c) chroma band as:

$$\begin{aligned} c_b(|X|^2) &= \alpha_c \sum_{f_k \in F_c} s(\Delta p_{\text{m2}}(k,b))\,|X_k|^2\,, \qquad b \in \{0,\dots,B_c-1\} \\ \text{where } f_k &= \frac{k}{N} f_s, \qquad k \in \left\{0,\dots,\frac{N}{2}\right\}, \qquad F_c = [50, 4000]\,\text{Hz} \end{aligned} \tag{6.1}$$

According to pattern, c_b is a linear function of the power spectrum $\left(|X_k|^2\right)$. The main action of c_b consists in the application of a spreading function s (defined in Eq. 6.4) that distributes energy from each DFT bin k to each chroma band b, and depends only on the pitch difference in semitones Δp_{m2} between k and b. f_k is the frequency in Hz of bin k with DFT size N, f_s is the sampling frequency, and F_c is the frequency range in Hz of bins that contribute to chroma. α_c is a scaling factor for the energy estimation, i.e. to bring the total energy estimation in line with that of the mel-spaced filters.

To compute the spreading function s, we first compute the fractional pitch classes, $p_{\text{bin}}(k)$ of a frequency bin k, and $p_{\text{band}}(c)$ of a chroma band b, where the range $[0,1]$ represents the octave, and $f0_{\text{ref}}$ is the reference frequency in Hz, i.e. the center frequency of the 0th chroma band:

$$p_{\text{bin}}(k) = \log_2 \frac{f_k}{f0_{\text{ref}}} \bmod 1, \qquad p_{\text{band}}(b) = \frac{b}{B_c}. \tag{6.2}$$

The relative pitch difference Δp_{m2} between frequency bin k and chroma band b is given in semitones:

$$\Delta p_{\text{m2}}(k,b) = 12\left(p_{\text{bin}}(k) - p_{\text{band}}(b) \bmod \left[-\frac{1}{2}, \frac{1}{2}\right]\right), \tag{6.3}$$

where $x \bmod \left[-\frac{1}{2}, \frac{1}{2}\right] = (x + \frac{1}{2}) \bmod 1 - \frac{1}{2}$.

For the spreading function s, a Hann window with a width $w = \frac{4}{3}$ semitones ($133\frac{1}{3}$ cents) is used [Gómez, 2006]:

$$s(\Delta p_{\text{m2}}) = \begin{cases} \cos^2\!\left(\frac{\pi \cdot \Delta p_{\text{m2}}}{w}\right) & \text{if } \frac{-w}{2} \geq \Delta p_{\text{m2}} \geq \frac{w}{2}, \\ 0 & \text{otherwise.} \end{cases} \tag{6.4}$$

As each frequency bin maps only to a few chroma bands, it is more convenient to loop over the frequency bins and distribute energy to the affected chroma bands. In this case, Δp_{m2} is computed for the bands only implicitly, by determining the affected central and neighbor chroma bands.

6.3.2 Mel-spaced Filter Bank

While the first filter bank, the chroma filter bank, accounts for musically relevant pitch, the second filter bank is intended to account for some aspect of the timbre of the target sounds. Timbre is, by a traditional definition, any other quality or attribute that can differ when pitch and loudness are fixed [American Standards Association, 1951]:

> **Timbre (Musical Quality).** Timbre is that attribute of auditory sensation in terms of which a listener can judge that two sounds similarly presented and having the same loudness and pitch are dissimilar.
> Note: Timbre depends primarily upon the spectrum of the stimulus, but it also depends upon the wave form, the sound pressure, and the frequency location of the spectrum of the stimulus.

By that definition, timbre is multi-dimensional, as two sounds with similar pitch and loudness can differ in many possible ways. Acoustic features commonly thought to be important to timbre are the spectral shape, which includes concepts like the harmonic spectra and formants, and temporal envelope characteristics (such as attack and decay) [Handel, 1989, pp. 170-173]. Other related acoustic features include noisiness, dynamic changes in spectral envelope, and micro-changes in pitch [Schouten, 1968]. Yet, the definition above could possibly include many diverse aspects, such as contextual aspects such as room characteristics, or even cultural aspects regarding the sounds.

The proposed approach is currently limited to describing the spectral shape over time (with short-time frames); a filter bank approach. As harmony, estimated by the chroma filter bank, was a strong motivating factor in describing the target signal, it is convenient for synthesis to have a complementary descriptor for timbre that can be treated similarly in terms of mathematical and programming concerns. However, it should be possible to extend this representation with additional aspects of timbre that fit some approximate linearity condition (as in Section 6.3.3).

The Mel-spaced filter bank was used for our main synthesis experiments. This filter bank is the computational precursor to a well-known descriptor, the mel-frequency cepstral coefficient (MFCC). MFCCs are often used as the front-end feature in automatic speech recognition, as well as commonly used as spectral descriptors for genre classification. The transformation to from spectrum to cepstrum, however, includes a logarithmic mapping, which makes them non-linear with respect to mixtures, and hence less appropriate for our synthesis approach.

The filter bank is made by placing filter center frequencies equidistant in the Mel scale, which was originally developed as an equal-distance scale for subjective pitch [Stevens et al., 1937]. Another related scale is the Bark scale, an attempt to approximate critical bands (the inherent bandwidths in the cochlea which predict simultaneous masking, for example) [Zwicker, 1961]. The ERB scale [Moore and Glasberg, 1983] is another method to measure the critical bandwidth in different frequency ranges. (Simultaneous masking is both level-dependent and asymmetric with regard to frequency [Scharf, 1971]. Therefore, merely representing a sound with a linear filter bank based on critical bands is not enough to model masking phenomena; extra modeling steps would be necessary).

Because any other linear filter bank would also satisfy the approximate linearity condition (in Section 6.3.3), there is no loss of generality in having chosen mel-spaced filters. Any other descriptors that satisfy a similar condition should be compatible with the synthesis method.

For convenience, we used the implementation of mel-spaced filter banks provided by Ellis' Rastamat toolkit [Ellis, 2005]. The b^{th} (of B_m) band has the following energy:

$$m_b(|X|^2) = \sum_{f_k \in F_m} \mathrm{tri}_{k,b} \, |X_k|^2 , \qquad b \in \{0, \dots, B_m{-}1\}, \qquad F_m = [0, 4000]\,\mathrm{Hz} \tag{6.5}$$

The width between bands (half the width of each central band), w_{mel}, and center frequency of each band b, $\mathrm{ctr}_{b,\text{mel}}$, are both given in mel units:

$$w_{\text{mel}} = \frac{f_{\text{max,mel}} - f_{\text{min,mel}}}{B_m + 1}, \qquad \mathrm{ctr}_{b,\text{mel}} = f_{\text{min,mel}} + (b+1) \cdot w_{\text{mel}}, \tag{6.6}$$

where $b \in \{-1, B_m\}$ denote the boundaries of the edge bands, and $f_{\text{min,mel}}$ and $f_{\text{max,mel}}$ are the lower and upper frequency edges from F_m converted to mels.

The filters themselves are triangle-shaped. The point-slope line equations sampled at DFT frequencies for the left (low) and right (high) sides of the filters are given by:

$$\mathrm{lo}_{k,b} = \frac{f_k - \mathrm{ctr}_{b-1,\text{Hz}}}{\mathrm{ctr}_{b,\text{Hz}} - \mathrm{ctr}_{b-1,\text{Hz}}}, \qquad \mathrm{hi}_{m,b} = \frac{\mathrm{ctr}_{b+1,\text{Hz}} - f_k}{\mathrm{ctr}_{b+1,\text{Hz}} - \mathrm{ctr}_{b,\text{Hz}}}, \tag{6.7}$$

where $\text{ctr}_{b,\text{Hz}}$ is the center frequency of a band, converted to Hz. Likewise, the equation for the triangle functions:

$$\text{tri}_{k,b} = \begin{cases} \text{lo}_{k,b} & \text{if } \text{ctr}_{b-1,\text{Hz}} \leq f_k < \text{ctr}_{b,\text{Hz}} \\ \text{hi}_{k,b} & \text{if } \text{ctr}_{b,\text{Hz}} \leq f_k < \text{ctr}_{b+1,\text{Hz}} \\ 0 & \text{otherwise.} \end{cases} \tag{6.8}$$

Here, the following conversions between Hz and mel scales are used:

$$\text{mel}(\text{Hz}) = 2595 \log_{10}\left(1 + \frac{\text{Hz}}{700}\right), \qquad \text{Hz}(\text{mel}) = 700\left(10^{\frac{\text{mel}}{2595}} - 1\right), \tag{6.9}$$

these having appeared prior in [O'Shaughnessy, 1987].

6.3.3 Approximate Linearity of Filter Banks

As shown in the previous sections, both descriptor filter banks (chroma and mel-spaced) are linear functions of the power spectrum. This linearity in terms of the power spectrum is helpful in predicting the descriptors of simultaneous mixtures of sounds.

Let v denote the mixture of two fixed-length frames with spectra X and Y and with gains g and h, i.e. $v(g, h, X, Y) = gX + hY$, and let f be a linear function of the power spectrum that represents a descriptor or descriptor vector.

Namely, the descriptors of a mixture can be estimated in the following way:

$$f \circ v = f(gX + hY) \approx g^2 f(X) + h^2 f(Y), \tag{6.10}$$

i.e. a linear approximation in the descriptor domain. This leads to the setting in which a target is approximated by non-negative (and non-destructive) combinations of sources, as will be seen in the following section, 6.4.

Equation 6.10 is based on approximating the power spectrum under two main assumptions, sparsity, e.g. in the case of harmonic sounds, and ignorance of individual sinusoid phases.

Sparsity plays a role in the first case, when two mixed sounds have little common frequency support; for example, harmonic sounds with different fundamental frequencies and few overlapping harmonics. At frequencies with no support overlap, i.e. energy only in X_k or Y_k, but not both, the approximation holds exactly: $|gX_k + hY_k|^2 = g^2 |X_k|^2 + h^2 |Y_k|^2$.

The power spectrum ($|X_k|^2$) is not exactly linear under mixtures, however. When X and Y have common frequency support, the true energy of the mixture in that frequency bin can be less or more than estimated by the sum of the energies. If the phases of X_k and Y_k are opposed, energy can be cancelled in the mixture (going potentially to zero). Likewise, if the phases of X_k and Y_k are similar, the true energy of the mixture can be increased. For example, taking the case where X_k and Y_k are equal, $X_k + Y_k = 2X_k$, consequently $|X_k + Y_k|^2 = 4|X_k|^2$, there is double the expected energy in that bin.

When the phases of individual sinusoids are not known, it is reasonable to guess the sum of energies, for the following reason. When adding two sinusoids of the same frequency, the following equation governs the amplitude of the resulting sinusoid (from the law of cosines in the complex plane):

$$A_3^2 = A_1^2 + A_2^2 + 2A_1A_2\cos(\Delta\theta), \tag{6.11}$$

where A_1, A_2, A_3 are the amplitudes of the two summed and the resulting sinusoid, and $\Delta\theta$ is the phase difference in radians. If phases are uniformly distributed, then so is the phase difference. Taking an expectation over phase differences, the contribution to squared amplitude (energy) of phase difference cancels.

6.4 An Objective Function for Mixture Mosaics: with penalty functions for polyphonic sampling, supporting transposition and variable-length tracks, from constant length frames described by filter banks

In this section, desired characteristics (criteria) for mixture mosaics are not only identified, but also given concrete mathematical form as an objective function. This function expresses a trade-off among these criteria in potential mosaics.

For example, the mixture should approximate the target descriptors well given the model. All else being equal, less extreme transformations should be preferred. The sampling process should not change the tonal transposition (resulting from the resampling factor) much from one frame to the next, when sampling from a continuous context. The sampling position should evolve forward in time according to the resampling factor. Sparser solutions with less polyphonic sampling, and longer continuous tracks, should be preferred when possible.

This function is not optimized directly, for a number of reasons: complexity, causality, etc; but it represents a guess at what the best solutions would look like. The solution methods are proposed in the following section, Section 6.5.

Mosaics are specified by a sparse data structure, the score, that lists for each target frame which set of sources and source positions are active (i.e. that have non-zero gains), along with their transformation parameters and gains. Let the non-gain parameters at an instant of the sampling process, be designated as an atom. Each atom is associated with an instantaneous output signal in descriptor space. As well, atoms are sampled within tracks, where each track represents a unique sampling context that is continued over time.

The score structure w (weights) is shown here as a sparse 3d array of size $T \times Q \times A$, with indices t for target frame, q for track, and a for atom block index (referencing source file index ϕ, the source position s, and transposition index u). The domain of w has the following restrictions:

$$w \succeq 0, \qquad \forall (t,q) : \sharp\{a : w_{tqa} > 0\} \leq 1, \tag{6.12}$$

that is, all weights are non-negative (as discussed in Section 6.3.3), and for each pair of target frame t and track q, there is at most one active atom.

The objective function $f_{\text{mix}}(w)$ is defined as follows:

$$f_{\text{mix}}(w) = \sum_{t=1}^{T} \left[\sum_{g=1}^{2} \lambda_{\text{fb},g} \| \mathcal{D}_g w_t - y_{t,g} \|_2^2 + \sum_{\substack{(q,a_{tq}) \in W_{t+}, \\ a_{t-1,q} = \lhd(t,q)}} \sum_{p \in \mathbb{P}} \pi_p(a_{tq}, a_{t-1,q}, \sigma_t) \right],$$

$$\text{where } W_{t+} = \{(q, a_{tq}) : w_{tqa} > 0\}, \qquad \lhd(t,q) = \begin{cases} a & \text{if } \exists a : w_{(t-1),q,a} > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{6.13}$$

Here W_{t+} is the set of active atoms and tracks in target frame t (a support set), and $\lhd(t,q)$ is a function that gives the atom index a used in the previous frame in track q, if any. The rest of the symbols are defined as:

a_{tq}	index of an atom active in target frame t and track q
$a_{t-1,q}$	index of an atom preceding a_{tq} in the same track, if any
w_t	vector (length n) of all atom weights in frame t
$w_{tqa}, w_{t,q,a}$	the weight in target frame t and track q for atom a
$g, \lambda_{\text{fb},g}$	index for filter type, parameter weighting for filter type g
$\mathcal{D}_g$	dictionary for filter type g: a $B_g \times A$ matrix of descriptors of A atoms
$y_{t,g}$	vector of target descriptors in t^{th} target frame for filter type g
$\pi_p, p, \mathbb{P}$	penalty function with index p, set of all penalty functions
σ_t	accumulated state about the score at target frame t.

For each frame t, the cost has two quite distinct parts. The first part, consisting of fidelity terms, measures how close the mixture of source frames is, based on the assumption of Section 6.3.3, to the target frame descriptors. This is a convex function known as least squares (sum of squared errors) that varies smoothly with changes in weights; punishing larger deviations in a filter proportionally more than smaller ones.

The second part, consisting of penalty functions, represents both the preference of certain atoms over others; as well as the dynamics, how the sampling context and transformation should evolve forward in time. By contrast, this second part is not smooth with respect to w; costs are incurred, as in a step function, when the weight for an atom goes from zero to non-zero. (This could be alternately represented as an additional input β_{tqa}, a binary mask to be multiplied by the penalty functions). These penalties are sparsity-inducing, because each penalty must be justified by improvement in fidelity, but fidelity improvements are diminishing with each new atom.

This two-part objective function can be interpreted as a maximum a priori (MAP) estimation problem, that is, a trade-off between a fit to the observed data, the likelihood (according to the target signal), and prior information (which sampling parameters are expected). In order to facilitate intuitive understanding of the trade-offs between fidelity and the various penalties, in each frame, the fidelity error of an empty score is normalized to unity (as described in the method, Section 6.5.2).

In a related issue, when $\mathcal{D}_g$ is low rank (e.g. due to source descriptors being similar), the least squares problem could be ill-posed, due to the solution not being unique (if two sources are the same in descriptor space, least squares has no reason to prefer one or the other). In this case, regularization, adding another function to the objective, can make the problem well posed by distinguishing between otherwise equal solutions. With respect to permuted track assignments, the objective is not well-posed, but this does not affect the proposed solution method to be discussed.

Optimizing $f_{\text{mix}}(w)$ directly is quite difficult. In fact, a similar problem, that of finding a linear approximation under a certain error with the least number of terms, has been shown to be NP-hard [Natarajan, 1995] (see Appendix C.2 for definition). The combinatorial aspect of this problem, in that one must choose subsets of active atoms, makes it particularly complex to solve, as the number of subsets scales exponentially with the number of atoms (2^A). The combination of the quadratic objective and binary choices in this objective qualify it as a mixed integer quadratic program (MIQP).

Representing the sampling and transformation process as a dictionary, the matrix $\mathcal{D}_g$, reflects not only the mixture assumption; it also implies discretizing the space of source positions and transformation parameters. (Preparation of the dictionary in our methods are detailed in Subsection 6.5.2).

An alternate form (of $\mathcal{D}_g w_t$) with a generic transformation would be: $\sum w_a f(h(\phi_a, s_a), \vec{p}_a)$, where h is the sampling function in descriptor space, f is a predictive model of the transformation, and $\vec{p}_a$ is the vector of transformation parameters for the atom with index a. In any case, when the other parameters are fixed, the subproblem of assigning weights is convex. This means that we can find precise assignments of the weights efficiently with local search.

6.4.1 Penalty Functions

Here, we describe a range of penalty functions relevant to mosaicing. As mentioned in the previous section, these functions and their associated parameters allow an explicit trade-off between fidelity error, and other wanted/unwanted characteristics of the score. The first set of penalties concern only the parameters chosen at a given instant.

6.4.1.1 Type I: Simple, Instantaneous

One of the simplest, and most necessary penalties, is to limit the amount of transformation (in this case, transposition using speed/pitch change). Here u_{P8} means transposition in octaves, $\lambda_{u\text{sqr}}$ is the weight for the quadratic penalty (more lenient for small transpositions), and λ_u is the weight for the l_1 penalty (more strict for small transpositions, and more lenient for large ones, relative to quadratic):

$$\pi_{\text{transp}}(u_{\text{P8}}) = \lambda_{u\text{sqr}} u_{\text{P8}}^2 + \lambda_u |u_{\text{P8}}|, \tag{6.14}$$

where u_{P8} is a function of u_L, the resampling factor used:

$$u_{\text{P8}} = -\log_2 u_L. \tag{6.15}$$

Another penalty prefers matches between frames of similar relative signal levels. This ensures, for example, that we don't heavily amplify quiet signals. Let $y_{\text{dB},t}$ and $z_{\text{dB},s}$ be the relative power in decibels (normalized to the mean energy of the each source or target file) for the target and source at frames t and s, respectively. Then, $\lambda_{20\text{dB}}$ is the cost of applying 20 dB of gain (a power of 10) to these relative levels:

$$\pi_{\text{gain}}(y_{\text{dB},t}, z_{\text{dB},s}) = \lambda_{20\text{dB}} \left| \frac{y_{\text{dB},t} - z_{\text{dB},s}}{20} \right|. \tag{6.16}$$

As this penalty also depends on the target (unlike the others), it fits more as a likelihood penalty, than as a prior.

6.4.1.2 Type II: Markov Dynamics, Continuity

This next group of penalties concern the changes in parameters at a given instant; put more descriptively, the continuity of individual tracks. Most of these only apply when existing tracks are continued.

For one, the best sampling position is the one exactly specified by the forward evolution given by the average tape speed. Let $\Delta s_{\text{fr}} = s_t - s_{t-1}$, the difference between source positions in frames, and $\lambda_{\Delta\text{fr}}$ be the cost for deviating by one frame.

$$\pi_{\Delta\text{src}}(\Delta s_{\text{fr}}, u_{\text{P8},t}, u_{\text{P8},t-1}) = \lambda_{\Delta\text{fr}} \left| \Delta s_{\text{lin}}(u_{\text{P8},t}, u_{\text{P8},t-1}) - \Delta s_{\text{fr}} \right|, \tag{6.17}$$

where the average relative speed in frames is defined by:

$$\Delta s_{\text{lin}}(u_{\text{P8},t}, u_{\text{P8},t-1}) = (2^{u_{\text{P8},t}} + 2^{u_{\text{P8},t-1}})/2. \tag{6.18}$$

To complement the above cost, which distinguishes between fine deviations in source position, a step cost, λ_{mask}, is applied to strongly discourage jumps in sampling position outside of a short window in the future. Let Δs_{s} be the difference in source positions in seconds, and $\tau_{\text{mask,s}}$ be the duration of the largest allowed jump ahead in seconds:

$$\pi_{\Delta\text{src mask}}(\Delta s_{\text{s}}, \phi, \phi_{t-1}) = \begin{cases} 0 & \text{if } \phi = \phi_{t-1}, \text{and } 0 < \Delta s_{\text{s}} < \tau_{\text{mask,s}}, \\ \lambda_{\text{mask}} & \text{otherwise.} \end{cases} \tag{6.19}$$

In terms of the transformation, it is a good idea to limit the changes in transposition within a track. We impose both quadratic and l_1 penalties, $\lambda_{\Delta u\text{sq}}$ and $\lambda_{\Delta u}$:

$$\pi_{\Delta\text{trans}}(\Delta u_{\text{P8},t}) = \lambda_{\Delta u\text{sq}}(u_{\text{P8},t})^2 + \lambda_{\Delta u}|\Delta u_{\text{P8},t}|. \tag{6.20}$$

Finally, we apply a fixed cost for creating a new track, so that the number of tracks should be as low as possible.

$$\pi_{\text{track}}(a_{t-1}) = \begin{cases} \lambda_{\text{track}} & \text{if } a_{t-1} = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{6.21}$$

6.4.1.3 Type III: Dynamics involving Accumulated State

This last set of penalties concern some wider scope, either outside of the instantaneous time frame, or outside of the current track. The structure σ_t, accumulates some aspect of the state at target frame t.

For example, one penalty aims to encourage the diversity of sampling in different source positions. The accumulation cost for source position s in track q is defined as:

$$\pi_{\text{accum}}(\sigma_t, s, q) = \lambda_{\text{accum}} \sum_{q' \neq q} \sigma_{\text{accum}}(s, q'), \tag{6.22}$$

so that sampling in a track q only affects the costs for the other tracks (including new tracks).

The source accumulator $\sigma_{\text{accum},t}(s, q)$ is initialized to zero in the first frame. When adding new atom to the score at sampling position s' and track q, a Gaussian blob with standard deviation $\tau_{\text{accum,fr}}$ in frames is accumulated at that position, via the max function:

$$\sigma_{\text{accum},t}(s, q, s') = \max\left(\exp\left(\frac{-(s - s')^2}{2(\tau_{\text{accum,fr}})^2}\right), \sigma_{\text{accum,t}}(s, q)\right). \tag{6.23}$$

The accumulator is updated each frame according to the decay constant $\beta_{\text{accum,fr}}$:

$$\sigma_{\text{accum},t}(s,q) = \beta_{\text{accum,fr}} \cdot \sigma_{\text{accum},t-1}(s,q). \tag{6.24}$$

The last two adjustments, rewards rather than penalties, work to counteract the sparsity penalties in some cases. The first reduces the total penalty when the number of current atoms is under a threshold, $\beta_{\text{min atoms}}$:

$$\pi_{\text{min atoms}}(\sigma_t) = \begin{cases} -\lambda_{\text{min atoms}} & \text{if } \sharp W_{t+} < \beta_{\text{min atoms}}, \\ 0 & \text{otherwise.} \end{cases} \tag{6.25}$$

The second favors continuing all tracks, but differentially favors continuing shorter tracks versus continuing longer tracks, according to an average duration parameter:

$$\pi_{\text{track len}}(\sigma_t) = -\lambda_{\text{track len}} \cdot \exp\left(\frac{-\sigma_{\text{prev frames},t,q}}{\tau_{\text{len,fr}}}\right), \tag{6.26}$$

where $\sigma_{\text{prev frames},t,q}$ is the previous length of track q in frames, and $\tau_{\text{len,fr}}$ is the desired mean duration in frames.

6.5 Mosaicing Methods based on Greedy Sparse Decompositions

In this section, we propose methods for finding mixture mosaics with the desired criteria, as expressed in the objective function in Section 6.4. We begin by giving background on the so-called "greedy" methods for sparse approximation.

6.5.1 Background: Matching Pursuit (MP) and OMP

There are many sparse approximation algorithms, possibly many of which would be suitable for this application. Their main action is to project a target signal onto signal components, known as atoms, of a collection known as the dictionary. This creates a new set of coefficients, that when multiplied by corresponding atoms of the dictionary, and summed together, gives the approximation of the target signal, such that only a few non-zero coefficients are used.

Greedy approximation algorithms seem to offer the simplest and most computationally efficient approach: to make locally optimal decisions (not considering the set as a whole) at each step. Matching Pursuit (MP) [Mallat and Zhang, 1993] is a well-known and simple method. In the "add" step, it selects the atom with the highest correlation with the residual signal, and adds it to the representation. Then the atom's contribution is subtracted from the residual. Starting from an empty set of atoms, it repeats the "add" step until stopping criteria are met, e.g. the error is below a threshold, a limit on number of active atoms is reached, etc.

Orthogonal Matching Pursuit (OMP) [Pati et al., 1993] is a refinement to MP. It adds a second "reweight" step: to adjust the weights of atoms in the solution to be optimal in a least-square error sense. This results in making the residual orthogonal to the active set of atoms; it reduces the chances of revisiting an atom (modifying its coefficient) from a previous "add" step, (which tends to occur in plain MP), speeding convergence to the solution.

Our proposed methods use a non-negative variant of the OMP algorithm (i.e. subject to the non-negativity constraint on w in Section 6.4, Equation 6.12), for every frame of the target signal, applied to target and dictionary consisting of normalized filter energies (see normalization schemes in Section 6.5.2 below), using penalty functions (such as those in Section 6.4.1) as criteria for atom inclusion. As this first method does not create coherent chains of atoms over time, the second method employs a second greedy strategy to continue atoms into coherent chains.

6.5.2 Preparation of Dictionary and Target

The main preparation tasks, before the algorithms can run, are the extraction and normalization of descriptors of the target and transposed sources. The descriptors are sampled uniformly in time (for source and target) and uniformly in transposition, in fractions of semitones, for the dictionary.

In the batch setting, target and source descriptors are extracted by a standard short-time Fourier transform (STFT) approach, i.e. according to some hop size, overlapping window size, and zero-padding factor. From these short-time spectra, chroma and mel-spaced filter banks are extracted as described in Sections 6.3.1 and 6.3.2.

6.5.2.1 Target Descriptors

As mentioned in Section 6.4, the target signal is normalized so that the fidelity error of an empty score is unity. This is in order to have an intuitive instantaneous trade-off for the penalty functions, in this case, the portion of target error reduced (improvement of fidelity). For example, imagine a cost parameter λ_{atom}, that adds a cost for each additional atom used in the frame. If $\lambda_{\text{atom}} = 0.2$, that means that an additional atom would have to reduce the initial fidelity error of that frame by 20% in order to be added in the optimal solution.

Normally, the target is not normalized in standard sparse decompositions, such as Matching Pursuit computed on audio signals. Rather, there is an implicit trade-off between adding new atoms, and the squared norm of the error, which itself corresponds to a physical quantity, the target signal energy. Therefore, atoms are allocated preferentially to the high-energy spectral and temporal regions of the target, and the signal-to-noise ratio (SNR), for example, given a fixed budget of atoms is maximized.

The penalty functions introduced in Section 6.4.1, although the costs vary over the different atom characteristics, are proxies for sparsity, because each new atom added to the score incurs these costs. The scale of the fidelity terms, as shown for the case of audio MP above, modulates the sparsity of the solution according to this scale.

However, in the case of polyphonic sampling, we might not want the number of voices to increase only in the louder parts, but rather with complexity, which is harder to pin down. As well, the fidelity term defined in Equation 6.13, i.e. the weighted sum of squares of residual filter bank energies, does not correspond to a physical quantity at all. Rather, in the case of two equally energetic frames, one with energy all in one band, the other with energy divided equally among two bands, the initial fidelity of the first frame is twice that of the second. So, the target signals are normalized in order that the sparsity, all other things being equal, stays approximately constant.

The target scale factor for a target frame t, tsc_t, is computed and stored as:

$$\text{tsc}_t = \sqrt{\sum_{g=1}^{2} \lambda_{\text{fb},g} \|y_{t,g}\|_2^2}, \tag{6.27}$$

and the corresponding $y_{t,g}$ signals are divided by tsc_t.

6.5.2.2 Dictionary (Transposed Source Descriptors)

For computing descriptors of the transposed sources, one has two options. If sources are known in advance, it is sufficient to process the source files at all transpositions (sampled uniformly in semitone fractions) and extract the descriptors by modulating the hopsize. However, to avoid this extra processing, or in the case that new sources should be added live, one can instead use models of transposition to predict the transformed descriptors.

Currently, the following simple models are used. For the chroma bank, one can assume that the energy is just shifting between pitches, using the circular shift:

$$\hat{c}_b(u_b) = c_{[b+u_b \bmod B_c]}, \tag{6.28}$$

where $u_b = u_{\mathrm{P8}}/B_c$ is the transposition in whole bands. Though this is not exactly true, as bandlimited interpolation can shift energy above and below the chroma region F_c, it is a simple and quick approximation.

To predict the mel-spaced bands, we first predict the interpolated power spectrum with linear interpolation (shown as lin[]), and apply the mel-spaced filters:

$$\hat{m}_b(u_L) = m_b\left(\frac{(|X|^2)_{\mathrm{lin}[k/u_L]}}{u_L}\right). \tag{6.29}$$

where u_L is the resampling factor for transposition index u, k is the frequency bin index, and the filters are defined in Section 6.3.2, Equation 6.5. This approach could also be used to predict the transposed chroma bank.

As in the case of the target descriptors, the transposed source descriptors are also normalized by the same function. This is done because the correlation, the optimal weight for a single atom (the solution to unconstrained least squares), has the form $y^T\vec{a}/\|\vec{a}\|^2$ (for target y and atom $\vec{a}$). Scaling the atoms to unit norm thus saves computation on a frequent and essential step of this algorithm.

The source scale factor for a target frame s and transposition index u, $\mathrm{ssc}_{s,u}$, is computed and stored as:

$$\mathrm{ssc}_{s,u} = \sqrt{\sum_{g=1}^{2} \lambda_{\mathrm{fb},g}\|\vec{a}_{s,g}\|_2^2}, \tag{6.30}$$

and the corresponding $\vec{a}_{s,g}$ signals, either the extracted or predicted descriptors, are divided by $\mathrm{ssc}_{s,u}$.

The dictionary matrix for filter type g, $\mathcal{D}_g$, is created by constructing the block index $a = s + (u-1) * S$, where s is the source index (of S) and u is the transposition index. The a^{th} column of $\mathcal{D}_g$ is given by $\vec{a}_{a,g}$, i.e. the normalized descriptor vector for that filter type.

6.5.3 Algorithm 1: Frame-based Mosaicing (FrOMPmos)

Now, the first decomposition can be introduced: a simple mixture method that decomposes individual target frames into source frames. This is done separately primarily for didactic reasons, but it could also be considered a special case of Algorithm 2. This algorithm was used as the *mix* method in the evaluation of the next chapter, although the alternate strategy of Section 6.7.1 (p. 119) could be used to produce similar solutions.

For each frame considered independently, this algorithm computes the fidelity improvements for each atom, to which it adds a subset of the penalty costs (the ones not considering predecessor atoms in tracks) as described in Section 6.4.1. If there is an atom that improves the solution, then this atom is added to the representation. This is followed by an OMP reweighting step over the subdictionary $\mathcal{D}_{g+}$ corresponding to the current active atoms. Then, the process is repeated until there are no more improving atoms.

This could be simply described as using the OMP approach, applied to the $f_{\text{mix}}(w)$ objective function (Section 6.4, Equation 6.13), independently for each frame. In this setting, track assignments are ignored, and only Instantaneous (Type I) and a subset of the Accumulated State penalties not requiring track assignments (Type III, π_{accum} and $\pi_{\text{min atoms}}$, Section 6.4.1.3) are used.

One quantity in our approach should be explained, because it is not typically part of pure signal, non-regularized MP/OMP methods, in which there is no need to relate in signal fidelity to other competing criteria. That is, they use the atom correlation denoted by ρ_a, which is the optimal gain as mentioned earlier, but not ρ_a^2, as will be shown, is the change in the fidelity term with the optimal gain applied.

To see this, we examine the quadratic fidelity term from Eq. 6.13, rewritten for a single frame and in terms of the current residual r and a candidate atom with index a, normalized descriptor vector $\vec{a}$ and correlation ρ:

$$\begin{aligned} f_{\text{fid}} &= \sum_{g=1}^{2} \lambda_{\text{fb},g} \|r_g - \rho\vec{a}_g\|_2^2 = \sum_{g=1}^{2} \lambda_{\text{fb},g} \left[(r_g - \rho\vec{a}_g)^T (r_g - \rho\vec{a}_g)\right] \\ &= \sum_{g=1}^{2} \lambda_{\text{fb},g} \left[r_g^T r_g - 2\rho\vec{a}_g^T r_g + \rho^2 \vec{a}_g^T \vec{a}_g\right]. \end{aligned}$$

By recalling the following, f_{fid} can be further simplified:

$$\sum_{g=1}^{2} \lambda_{\text{fb},g}\, \vec{a}_g^T r_g = \rho, \qquad \sum_{g=1}^{2} \lambda_{\text{fb},g}\, \vec{a}_g^T \vec{a}_g = 1, \qquad f_{\text{fid}} = \left[\sum_{g=1}^{2} \lambda_{\text{fb},g} \|r_g\|_2^2 \right] - \rho^2.$$

This shows that when adding an atom with correlation ρ_a, the change in the quadratic fidelity term is $-\rho_a^2$.

A pseudocode version of Algorithm 1 is given on this page.

Algorithm 1 Frame-based mosaicing (FrOMPmos)

```
    **Inputs**: dictionary $\mathcal{D}_g$, target desc. sequence $\{y_{t,g}\}_{t\in\{1...T\}}$
    **Parameters**: fidelity and penalty parameters of $f_{\text{mix}}(w)$
    **Outputs**: sparse score $w$

 1: $w \leftarrow$ initScore()
 2: for all target frames $t \in \{1,...,T\}$ do:
 3:     $r \leftarrow y_t$                                   ▷ Initialize target frame residual.
 4:     repeat  $(a, \rho_a, \kappa_a) \leftarrow$ SelectAtom($r$)
 5:         if $\kappa_a < 0$ then:                          ▷ Is there an improving atom?
 6:             $w_{t,a} \leftarrow \rho_a$                  ▷ Add atom $a$ with weight $\rho_a$.
 7:             if $\sharp W_{t+} > 1$ then,                 ▷ More than one atom?
 8:                 $w_{t+} \leftarrow$ ReweightFrame($w_{t+}$)
 9:             $r \leftarrow r - \rho_a \vec{a}_a$          ▷ Update residual.
10:     until $\kappa_a \geq 0$                              ▷ No more improving atoms.
11: return sparse score $w$.

12: function SelectAtom($r$)
13:     $\{\rho_a\}_{a=1...A} \leftarrow \max(\lambda_{\text{wt},g} \sum_g r_g^T \mathcal{D}_g, 0)$  ▷ Compute all correlations+
14:     $\{\kappa_a\}_{a=1...A} \leftarrow -\rho_a^2 + \sum_{p\in\mathbb{P}} \pi_p(a, \sigma_t)$  ▷ Improvements plus penalties
15:     $a \leftarrow \arg\min_a \kappa_a$                   ▷ Find minimum cost atom.
16:     return $(a, \rho_a, \kappa_a)$                      ▷ Best atom, correlation, and cost

17: function ReweightFrame($w_{t+}$)
18:     $w_{t+} \leftarrow \arg\min_{w_{t+}} \sum_g \lambda_{\text{wt},g} \|\mathcal{D}_{g+} w_{t+} - y_{g,t}\|^2$ s.t. $w_{t+} \succeq 0$
19:     return $w_{t+}$
```

There are many possible methods to solve the non-negative minimization in ReweightFrame. As the objective is smooth, projected gradient methods are a sufficient approach. The problem dimension should be small, given few entries of W_{t+}. We use the TFOCS toolkit [Becker et al., 2014] to solve instances in our implementation.

Once the final linear weights for each frame are selected, they are used to compute the signal gains for the atoms in that frame:

$$\mathrm{gain}(w_{t,a}) = \sqrt{\frac{w_{t,a} \cdot \mathrm{ssc}_{a,u}}{\mathrm{tsc}_t}}. \tag{6.31}$$

This method gives the best match at each frame, but cannot do much to promote continuity between frames. The best we could do within this algorithm is imposing additional costs on atoms that are not continuations of atoms in the previous frame. In some cases, this may work; but in some cases it can promote duplicate successor atoms, and it does not create a clear links of succession.

In order to organize continuations of the sampling context, we introduce the next algorithm. Algorithm 2 assigns atoms to tracks, allowing it to select unique successor atoms according to the Dynamics (Type II) penalties.

6.5.4 Algorithm 2: Tracks-based Mosaicing (TrOMPmos)

Each atom created by this algorithm is assigned to either a preexisting or a new track. A two-step process is proposed for approximating each target frame. In the first step, tracks existing in previous frames are either continued, by choosing atoms related to these previous ones, or terminated. Here the continuity penalties strongly affect which atoms are added to previous tracks. In the second step, new tracks are created by adding any improving atoms left.

This algorithm uses the same reweighting scheme as the previous algorithm, which might cause abrupt changes in weight within a track, or might prematurely drop tracks that might be continued. It remains future work to more closely assess the negative effects resulting from this, on output quality and track continuations, and to possibly propose a smoothing scheme.

Pseudocode is given for Algorithm 2 on the next page.

6.5.5 Further Enhancements

Algorithm 2 is structually complete for creating mixture mosaics with variable-length continuous tracks. However, two problems persist. First,

Algorithm 2 Tracks-based mosaicing (TrOMPmos)

```
**Inputs**: dictionary $\mathcal{D}_g$, target desc. sequence $\{y_{t,g}\}_{t\in\{1...T\}}$
**Parameters**: fidelity and penalty parameters of $f_{\text{mix}}(w)$
**Outputs**: sparse score $w$; track start and end markers
SELECTATOM and REWEIGHTFRAME as defined in Algorithm 1

1:  $w \leftarrow$ initScore()
2:  for all target frames $t \in \{1, ..., T\}$ do:
3:      $r \leftarrow y_t$                                ▷ Initialize target frame residual.
4:      for all $q \in W_{(t-1)+}$ do                    ▷ Continue previous tracks
5:          $(a, \rho_a, \kappa_a) \leftarrow$ SelectAtomTr($r$, $a_{t-1}$)
6:          if $\kappa_a < 0$ then:                      ▷ Is there an improving atom?
7:              $(r, w_{t+}) \leftarrow$ AddReweight($r, t, q, a$)
8:          else trackEnd$_q \leftarrow (t-1)$  ▷ Track $q$ ends in the previous frame.
9:      repeat $(a, \rho_a, \kappa_a) \leftarrow$ SelectAtom($r$)   ▷ Create new tracks
10:         if $\kappa_a < 0$ then:                      ▷ Is there an improving atom?
11:             $q \leftarrow$ NewTrackIndex()
12:             $(r, w_{t+}) \leftarrow$ AddReweight($r, t, q, a$)
13:             trackStart$_q \leftarrow t$              ▷ Track $q$ starts in this frame.
14:     until $\kappa_a \geq 0$                          ▷ No more improving atoms.
15: return sparse score $w$.

16: function AddReweight($r, t, q, a$)
17:     $w_{tqa} \leftarrow \rho_a$             ▷ Add atom $a$ to track $q$ with weight $\rho_a$.
18:     if $\sharp W_{t+} > 1$ then,                     ▷ More than one atom?
19:         $w_{t+} \leftarrow$ ReweightFrame($w_{t+}$)
20:     $r \leftarrow r - \rho_a \vec{a}_a$                ▷ Update residual.
21:     return $(r, w_{t+})$

22: function SelectAtomTr($r, a_{t-1}$)        ▷ Includes Continuity penalties
23:     $\{\rho_a\}_{a=1...A} \leftarrow \max(\lambda_{\text{wt},g} \sum_g r_g^T \mathcal{D}_g, 0)$  ▷ Compute all correlations+
24:     $\{\kappa_a\}_{a=1...A} \leftarrow -\rho_a^2 + \sum_{p\in\mathbb{P}} \pi_p(a, a_{t-1}, \sigma_t)$   ▷ Impr. plus penalties
25:     $a \leftarrow \arg\min_a \kappa_a$                ▷ Find minimum cost atom.
26:     return $(a, \rho_a, \kappa_a)$           ▷ Best atom, correlation, and cost
```

it might choose sources that cannot match the target well in future frames, causing shorter sampling tracks. Second, by confining atoms to the uniform grid (discussed in Sections 6.4 and 6.5.2), particular distortions are introduced in the synthesis output. The following enhancements are intended to mitigate the above problems.

6.5.5.1 Lookahead (Rolling Horizon)

In Algorithm 2, when selecting the best atom to add in a certain frame, it is unknown which atoms have successor atoms that can continue the track into future atoms. If some frames of the target are known in advance, one can look ahead, and estimate how likely it is that a current atom in a frame will lead to fidelity improvements for successor atoms in future frames.

For this estimate, or heuristic, there are many possibilities, trading off between accuracy and computational efficiency (and simplicity). Here, a simple method is proposed.

Specifically, to the cost for an atom at frame t, we accumulate costs associated with frames $t+1$ to $t+\tau_{\text{look,fr}}$, assuming no change in transposition and steady evolution in source. For the cost/utility of the current atom we take the minimum of these accumulated time steps. Thus we select atoms maximizing utility at the current frame and over $\tau_{\text{look,fr}}$ frames into the future. Once atoms are selected, we update the residual which we additionally maintain for the lookahead frames.

In Section 6.6.1, this strategy is shown to increase the mean and maximum track length in recovered scores.

6.5.5.2 Continuous sampling

By default, the SelectAtom and SelectAtomTr routines from Algorithm 2 select only atoms from the grid divided uniformly in time (in spectrogram frames defined by the hopsize) and uniformly in transpositions (in semitone fractions). However, synthesizing coherent tracks using only atoms from this grid causes artifacts relating to phase cancellation, leading to results sounding “phasey” or “granulated”.

The artifacts are caused when sampling harmonic sources. When maintaining an approximate sampling context, the overlapping frames have similar frequency support, but the timing inaccuracies between the sampling positions of predecessor and successor atoms lead to small differences in phase or frequency that can cause unwanted modulation in the mixture. However, by ensuring that the sampling times for exactly continued atoms exactly follow the forward evolution in time defined by the tape speed (resampling factor L), and by interpolating L as it evolves at each sample position, overlapped sampling can be made equivalent to continuous sampling, and phase cancellations can be eliminated (within single tracks).

To sample tracks exactly, exact sampling candidates are added to the SELECTATOMTR routine. Once the previous atom's source and transposition are already known, candidates for exact sampling successor atoms are chosen using only a single degree of freedom, the current transposition. The exact change in source position for each candidate is then computed according to Equation 6.18, page 102.

To encourage the use of exact sampling atoms, a fixed cost, λ_{grid} is added for the inexact grid atoms. For computing correlations and reweighting frames at intermediate source positions, the existing dictionary is linearly interpolated to create mini-dictionaries, on demand.

One further refinement can be made regarding the exact sampling candidate atoms. While the previous step continues past sampling positions with the correct continuous values, these candidates are still selected using a grid of transposition values. To produce transposition values not confined to this grid, κ_a, the atom utility function, can be locally interpolated around the peak, using parabolic interpolation to find the maximum utility transposition, as in [Serra, 1989]. Then, the source position is updated according to Equation 6.18.

To hear a comparison of inexact continued tracks (sampling positions defined on a grid) and the exact continued sampling positions, refer to Sound files "phase-exact.wav" and "phase-inexact.wav" (also listed in Appendix B). In these examples, a short source excerpt of a single violin playing is resampled at 7/6 speed, which transposes it up by a whole tone. This transposed signal is used as a target sound using Algorithm 2, using the same parameters (see Appendix) except for turning on or off the exact continuation. Although the exact variant has a number of spurious atoms, it mostly recovers the main "identity" track on the $\nearrow$ diagonal of the score, and the inexact recovers the diagonal completely (see Figure 6.2). However, the phasiness effect of the inexact sampling positions can be seen in the spectrogram, in Figure 6.3, and heard in the sound examples.

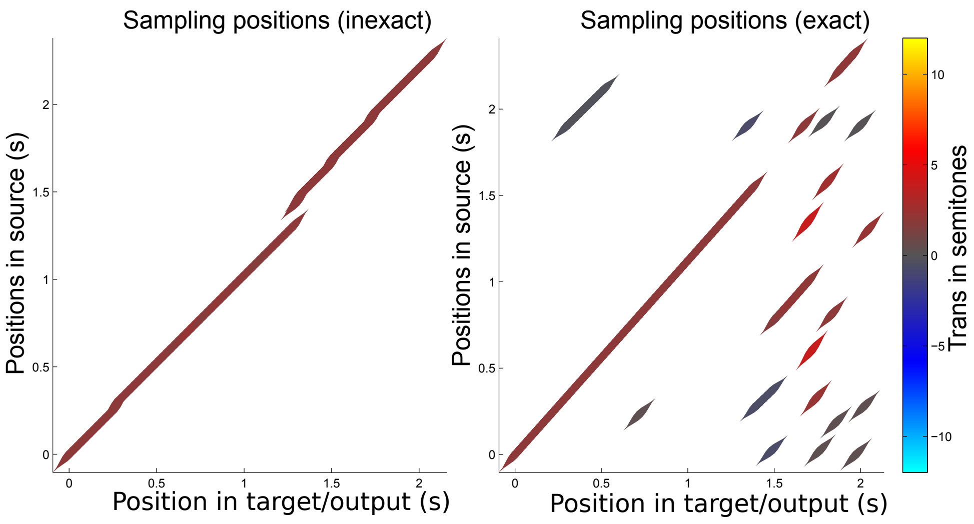

Figure 6.2: Recovered scores for inexact (left) and exact continuation example sounds, where the target is the source resampled at 7/6 speed, equivalent to transposing up a whole tone. The source input positions and output sampling positions in time, along with an approximation of the overlapping windows, and the transposition factor from resampling (color) are shown.

6.5.6 Resynthesis

Once the score is determined, resynthesis is simply a matter of overlapping and adding the sampled frames; for exactly continued frames, this entails interpolation of the resampling rates and sample positions.

6.6 Synthesis Example

In this section, various objective aspects of a synthesis example are shown for TrompMOS, Algorithm 2.

For the example, one of the mosaics produced for the subjective evaluation in the next chapter is used. In this example, the source is an excerpt of monophonic violin passages (Sound 2b, Appendix B) from a minuet of Boccherini (String Quintet in E, Op.13, No.6); the target is an excerpt from the opening of Bach's The Art of the Fugue played on piano.

The following command was used to generate this example:

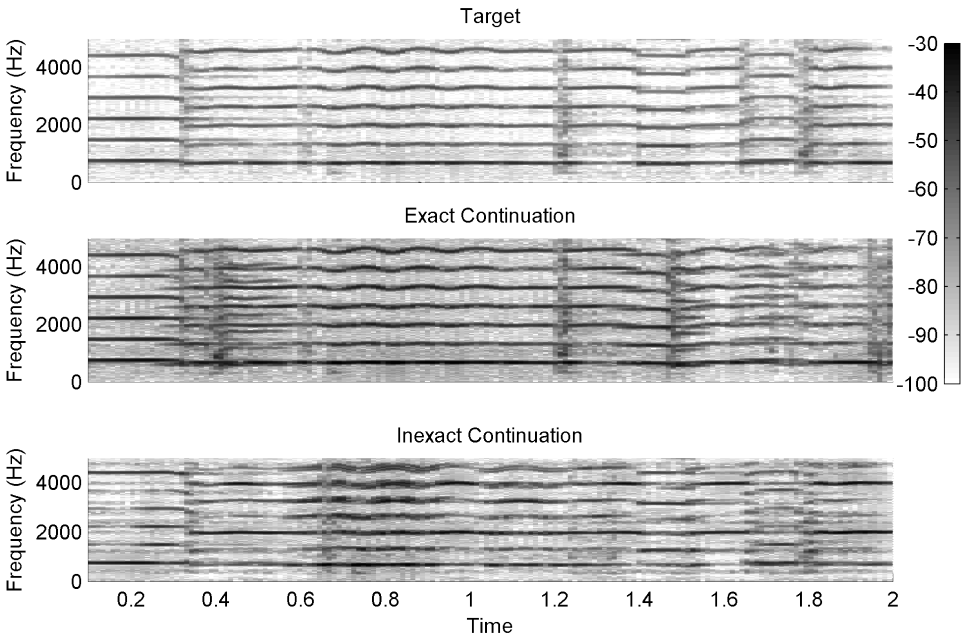

Figure 6.3: Example of phasiness caused by resampling on a grid (inexact) vs. exact resampling. Spectrograms are shown for a target signal (top), same as the source signal but sped up and transposed by a whole tone; and reconstructions using exact continuation (middle) and inexact continuation (bottom). For the inexact continuation (bottom), which samples from a grid of source times, phase cancellation of partials is clearly shown (e.g. in the first 200ms). Spectrograms show 23ms windows with 2x overlap.

```
genMosaic('fugue1','boccherini','hopsize',1024,
'iTargetChannels',1,'iSourceChannels',1,'cEnergyPer20Db',0.2,
'cInexactPenalty',3.4,'cMix',0.7,'cTransDeltaSqrPerOctave',80,
'cTransSqrPerOctave',0.4,'cSrcAccum',1,'cTrack',0.2,
'cTrackLen',0.2,'mInterpResamp',1,'sTracks',1)
```

which specifies a number of parameters, including a hopsize of 1024 samples, a $\lambda_{20\text{dB}}$ of 0.2, a constant penalty of 3.4 for inexactly continued (grid) atoms, a 70% to 30% penalty for fidelity errors in chroma vs. mel-spaced filter bands ($\lambda_{\text{fb},g}$), a penalty of 80 for each octave change in instantaneous transposition ($\lambda_{\Delta u\text{sq}}$), a penalty of 0.4 for each octave of instantaneous transposition ($\lambda_{u\text{sq}}$), λ_{accum} of 1, cost of creating a new track λ_{track} of 0.2, cost favoring

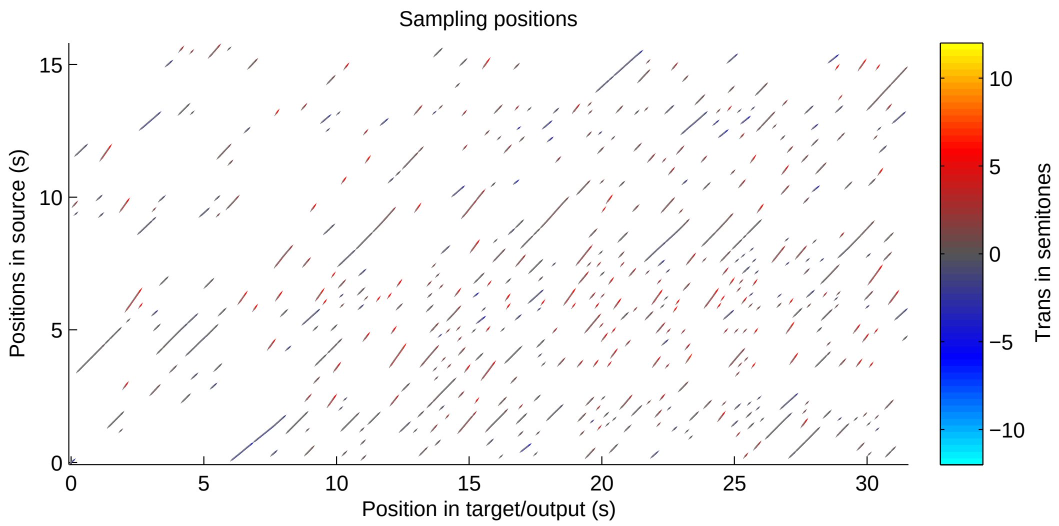

Figure 6.4: Sampling diagram showing target positions (x-axis) vs. source positions (y-axis) with colors denoting the transpositions in semitones.

longer tracks $\lambda_{\text{track len}}$ of 0.2; not to mention a number of default parameters.

To give an overview of the score as a whole, see Figure 6.4, which shows the activations of sources (x-axis) at every instant in the target/output (y-axis). Figure 6.5 shows the target, model, and synthesis descriptor sequences for this example.

6.6.1 Lookahead Test

Using a different target, a pop music excerpt from Madonna's "Give It 2 Me", we compare results over a range of lookahead horizons, using two different sources (the violin, and water pouring). Otherwise, equal parameters were used for mosaicing and synthesis. They are compared by maximum and mean track length, and model fidelity for chroma and timbre. Results are shown in Figure 6.6.

Looking at the track lengths, we see the first two lookahead frames help substantially– just a few frames seem to promote longer tracks. Max length grows up to $\tau_{\text{look,fr}} = 10$, while mean length peaks at $\tau_{\text{look,fr}} = 2$ (suggesting a trade-off; when frames are successfully continued, they can suppress other continuations).

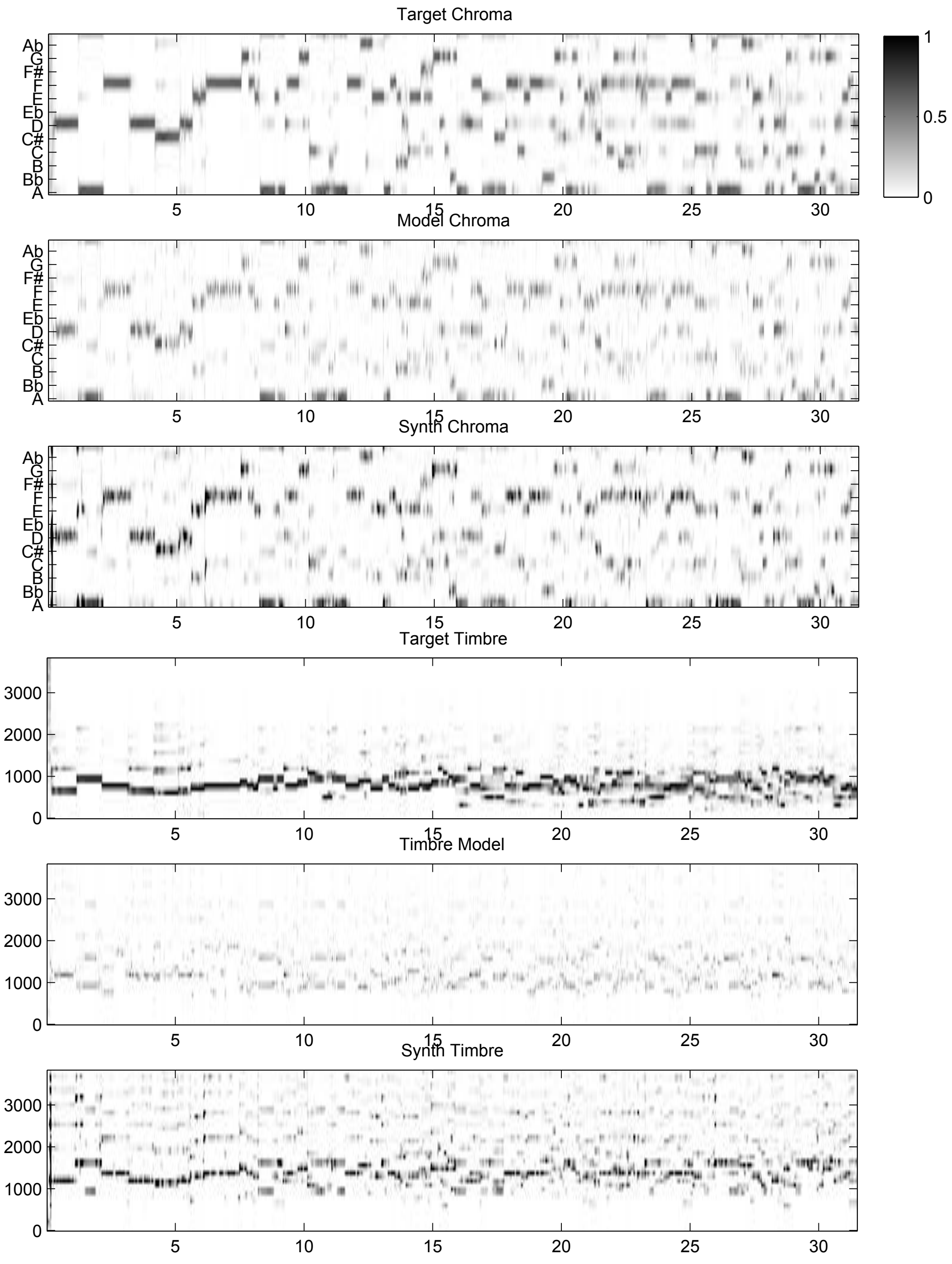

Figure 6.5: Similarities between target, model, and synthesis normalized descriptor sequences for chroma (above) and mel-spaced filter banks (below)

In contrast, lookahead does not seem to negatively affect the fidelity error when more than two frames of lookahead are used. As the lookahead policy is conservative, and does not promote an atom unless it is improving for future frames, this is expected.

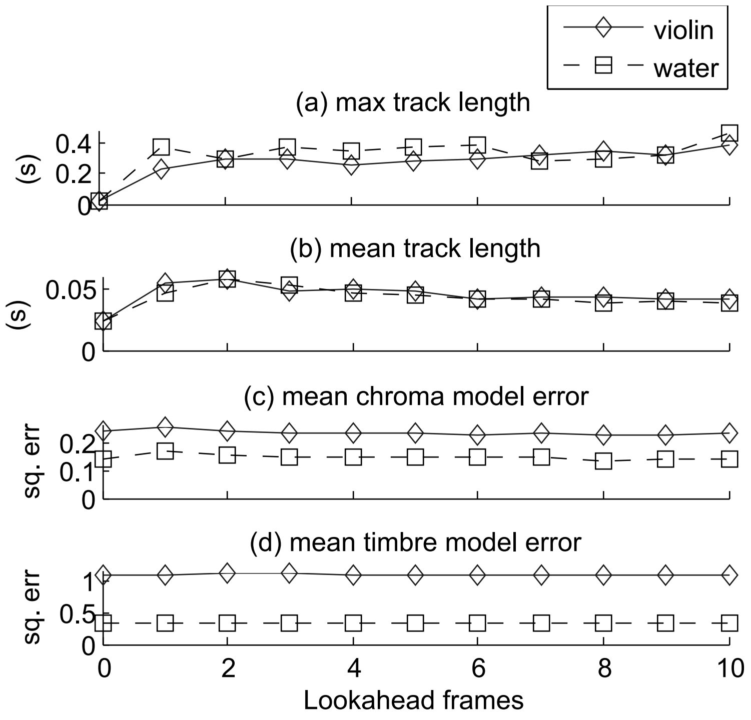

Figure 6.6: Changes in track length and model fidelity due to different lookahead horizons. Max and mean track length both benefit from modest lookahead, while max keeps increasing. Model fidelities (distance from the target) are barely affected by changes in the horizon.

6.7 Alternate Approaches based on real and convex optimization

Prior to developing the methods in Section 6.5, and prior to adopting an explicit representation of tracks, some effort was spent applying another branch of sparse approximation algorithms, convex methods, in the hope of finding mixture methods that could also meet the continuity criteria for multiple frames in the future. In this section, these approaches, their characteristics, and some of their limitations are sketched.

Although real and convex approaches to sparse approximation can be more computationally intensive than greedy approaches, their basic premise and hope is this: that by formulating combinatorial problems as optimization over a real domain, by a series of strictly local moves, one arrives at either a good (general) or an optimal (convex) solution. In a way, this is quite similar to the greedy premise.

Perhaps as computation becomes cheaper and mathematical techniques improve these approaches will become more competitive; or perhaps there will be some situations, in which the higher quality that could be achieved with a long-running batch process might be preferred over the something faster.

6.7.1 Basis Pursuit (BP) and Weighted BP

As mentioned previously, optimizing $f_{\text{mix}}(w)$ (Equation 6.13, page 99) due to its formulation of penalty functions over the support set, is likely to be computationally hard; finding optimal solutions to f_{mix} as posed may take exponential time in the size of the problem.

One promising approach is to replace the combinatorial parts of the objective function, i.e. the parts that depend explicitly on which subsets are active (in f_{mix}, the penalty functions), with a convex surrogate function with similar characteristics as the combinatorial one. As long as the domain meets some restrictions, this creates a convex optimization problem.

Convex problems have the property that methods using local search can be guaranteed to find a global optimum. Some classes of convex problems, such as linear programming (LP), have the remarkable property of admitting algorithms that find globally optimal solutions in polynomial time in the size of the problem [Boyd and Vandenberghe, 2004, p. 6]. The running time is also affected by the desired accuracy and numerical condition of the problem.

A canonical example of this approach is Basis Pursuit (BP) [Chen et al., 2001] approximation method, which requires solving an LP problem. For example, say we want to represent signal y with as few elements of a dictionary $\mathcal{D}$ as possible; and want a trade-off between approximation fidelity and number of elements used. This gives the following optimization problem:

$$\min_x \frac{1}{2}\|\mathcal{D}x - y\|_2^2 + \lambda\|x\|_0, \tag{6.32}$$

where $\|x\|_0$ denotes the number of non-zero elements of x, and λ is some trade-off parameter between fidelity and sparsity.

The BP approach suggests the following convex problem, known as Basis Pursuit Denoising (BPDN):

$$\min_x \frac{1}{2}\|\mathcal{D}x - y\|_2^2 + \lambda\|x\|_1, \tag{6.33}$$

where $\|x\|_1$, the l_1 norm, denotes the sum of the absolute values of the entries. The l_1 norm favors relatively sparser solutions because coordinates with a small non-zero weight pay a relatively larger cost, compared the l_2 norm or the sum-of-squares penalty; and many coordinates in the solution are pushed to zero or near-zero.

The BP approach is thus to find the optimal solution to Equation 6.33, to select the support set of components by checking which elements of x are greater than a threshold, and to fix the support set, finding an optimal set of coefficients.

This strategy can be applied to a limited version of f_{mix}, for example, using only Type I penalties. The idea is to apply the l_1 penalty weighted by the sum of penalties for each atom, which are fixed (denoted here by the vector π_{sum} of length n). Furthermore, as the domain is non-negative, we can replace the l_1 norm with a linear functional, where $\langle x, y\rangle$ denotes the dot product between vectors x and y.

In this case, the optimization for a single frame can be represented as:

$$\min_{w \preceq 0} \|\mathcal{D}w - y\|_2^2 + \lambda\langle \pi_{\text{sum}}, w\rangle, \tag{6.34}$$

where w is a non-negative length-n vector containing the weight for each atom. The above problem may be solved directly with a convex optimization toolkit. The λ that drives the solution uniformly to zero is known, in this case:

$$\lambda_{\max} = \max_a \frac{|2\mathcal{D}^T y|_a}{\pi_{\text{sum},a}}, \tag{6.35}$$

found by equating the gradients of the two competing criteria in Equation 6.34. A smaller value than $\lambda_{\max}$ can be chosen, the solution thresholded as in BP, and the optimal weights found.

This approach can be extended to solve for all frames, as long as there are only Type I penalties (no dynamics). One can adopt a greedy strategy in time, and add some of the Type III penalties, such as π_{accum} (Equation 6.22), which is fixed for each atom once past states are known. One can favor different sparsities, such as in $\pi_{\text{min atoms}}$ (Equation 6.25), simply by thresholding the convex solution at different levels.

When making frame-by-frame decisions, the approach of Equation 6.34 gives satisfactory solutions (as in the greedy approach), but it does not make track assignments or continue tracks forward in time. It was this method that was used in AugMos [Coleman et al., 2010], as well as in the prototypes described in Appendix D.

To fully extend the convex approach to the Continuity (Type II) penalties (with or without tracks) and to compute a so-called smoothing solution (to find an optimal solution when some of the target signal is known in advance), one first needs some sort of representation of dynamics that is compatible with convex or near-convex formulations. Using a linear operator to represent dynamics is a natural choice, especially because the norm of an affine function, i.e. $\|Dx_{t-1} - x_t\|$, is naturally convex for any true norm.

6.7.2 Dynamics and Cost Operators

There are at least two approaches to modeling the second-order costs in real optimization. One, that can be traced to the Kalman filter, provides a point estimate for state evolution. Another involves directly representing the second-order cost matrix implied in f_{mix}. Both can be computationally expensive to represent and compute.

6.7.2.1 Dynamics Operators: General and 2D Convolution

The function of a dynamics operator is, given a past state x_{t-1}, to give a likely future state $x_t = Dx_{t-1}$. Sparse linear dynamics can be interpreted in the following way: if the previous state contains an atom a, then the next state likely contains an arbitrary subset of future atoms; and the linear function D gives the superposition of all of those arbitrary subsets, according to which previous atoms are active. When used in conjunction with a convex constraint or penalty term, the dynamics operator predicts the most likely future state as the center point around which greater penalties are applied.

In the general approach, this is complex to even represent, let alone to do repeated computations, i.e. multiplication with an $A \times A$ matrix, where A, the number of atoms in the dictionary, is large.

The basic concept we would like to represent, for mixture mosaicing, is that they are branching; i.e. for a single atom in a previous state, it is

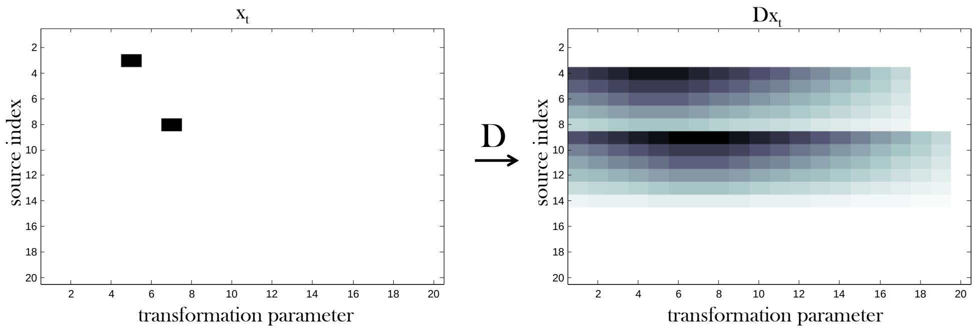

Figure 6.7: Illustration of spreading/branching effect of dynamics operator D. This operator depicts dynamics in which the source index always moves forward, and the transformation parameter can shift in either direction.

likely continued by a small set of related atoms in the future. This matches the general form present in these Continuity (Type II) penalties: $\pi_{\Delta\text{src}}$, $\pi_{\Delta\text{src mask}}$, and $\pi_{\Delta\text{trans}}$ (Equations 6.17, 6.19, and 6.20).

One alternate class of dynamics operators, shift-invariant functions, are easier to specify and compute (with 2D convolution). Yet, they still represent this branching property. In our domain, where each atom represents a given source and transposition, this represents the following: for each presence of an atom in a past state, an equally shaped blob of related states is deposited into the future state. See Figure 6.7 for an illustration of a 2D convolution dynamics operator.

Note that the shift invariance of this operator is contradicted by the non-shift invariance of the $\pi_{\Delta\text{src}}$ penalty and the Δs_{lin} function (Equations 6.17 and 6.18). That is because the most likely future state in source position, is less or more depending on the tape speed/transpositions of past and future states.

Another contradiction present in this approach: as usual, we are searching for sparse approximations, but a branching operator is anti-sparsifying; that is, it converts sparse previous states into non-sparse future states.

6.7.2.2 2D Cost Operators

Perhaps a more direct approach would be to form the cost matrix K implied by sampling the second order costs directly, i.e. $\sum_{p\in\mathbb{P}} \pi_p(a, a_{t-1})$ from

Equation 6.13, page 99. This leads to a non-convex formulation, discussed in Section 6.7.4.

Taking in mind the computational and sparsity limitations of these operators, two approaches to sparse dynamics can be discussed.

6.7.3 Convex Dynamics: Kalman Filters

The convex approach to dynamics is simply to penalize the norm deviation (commonly sum-of-squares) from the expected state, i.e. $\|Dw_{t-1} - w_t\|_2^2$. This is equivalent to a Gaussian prior probability around the point estimate, but of course, other norms like l_1 can also be used. Deviations from expected state in this approach are referred to as "innovations".

It is possible to set up a non-negative dummy variable x_t to absorb only the negative innovations (leading to sparser future states) and so penalize the positive and negative innovations separately. This gives "discounts" to the negative innovations [Coleman et al., 2011], in the following way:

$$\min_{x,w \preceq 0} \sum_{t=1}^{T} \|\mathcal{D}w_t - y_t\|_2^2 + \lambda_1 \langle \pi_{\text{sum}}, w_t \rangle + \lambda_2 \|Dw_{t-1} - w_t - x_t\| + \lambda_3 \|x_t\| \quad (6.36)$$

where both x and y are constrained to be elementwise non-negative, and the innovation costs are any norms. In this scheme, successor states are penalized less for atom transition alternatives not taken, and penalized more for unlikely transitions that are taken.

This scheme was tested with promising results. One drawback is finding suitable λ parameters that give non-zero solutions for future states, as it's not so easy to find closed form $\lambda_{\max}$ as in Equation 6.35 (as the multiple parameters interact).

6.7.4 Non-convex Dynamics

Two other approaches involve the quadratic or bilinear form i.e. $\langle w_t, Dw_{t-1} \rangle$. These functions, like the function $f(x, y) = xy$, are generally non-convex, having the shape of a saddle-point. This means that optima found with local search are only guaranteed to be locally but not globally optimal.

In special cases, if D is positive semi-definite or negative semi-definite, the form can be convex or concave. In general, it can be formed as a difference-of-convex function (a function class that has good performance in heuristic methods) through singular value decomposition of the matrix.

The first non-convex approach uses a Kalman dynamics operator D, to maximize the overlap between the future state, and the future states indicated by the past state. This is done by minimizing their negative dot product: $-\langle w_t, Dw_{t-1}\rangle$.

The other minimizes the second-order cost function, in a way similar to the l_1 penalty (more weight is penalized linearly) applied to future and past states: $\langle w_t, Kw_{t-1}\rangle$.

Model objective functions covering these approaches would be:

$$\min_{w \preceq 0} \sum_{t=1}^{T} \|\mathcal{D}w_t - y_t\|_2^2 + \lambda_1 \langle \pi_{\text{sum}}, w_t\rangle - \lambda_2 \langle w_t, Dw_{t-1}\rangle, \tag{6.37}$$

$$\min_{w \preceq 0} \sum_{t=1}^{T} \|\mathcal{D}w_t - y_t\|_2^2 + \lambda_1 \langle \pi_{\text{sum}}, w_t\rangle + \lambda_2 \langle w_t, Kw_{t-1}\rangle, \tag{6.38}$$

One heuristic method to finding local solutions to non-convex problems is sequential convex programming (SCP). This approach is to solve a non-convex problem by solving a sequence of convex approximations to the original problem.

Two possible SCP strategies can be applied to Equations 6.37 and 6.38. The first is known as alternating convex optimization. In many problems, if you isolate a subset of variables, the problem becomes convex when the other subset is fixed. Then, you can find solutions by alternately fixing and improving the subsets. When fixing the solution at odd or even times t, the bilinear terms become linear and convex. Another common strategy is linearizing, replacing with a first-order Taylor expansion, of concave or non-convex terms, and updating this expansion every iteration.

6.7.5 Summary of Real and Convex Methods

Real or convex approximations to the mixture mosaics over multiple frames can be costly, due to computing a linear operator of size A^2, where the number of atoms, A, is large. However, they are capable of finding solutions that are in some sense optimal over multiple frames in the future. Perhaps in the future these convex approaches can be combined with explicit track assignments.

6.8 Conclusions

In this chapter, new methods for audio mosaicing that operate in the filter bank energy domain have been presented. These methods are able to imitate a target signal with mixtures, while also promoting longer subsequences of the sampling context, and promoting consistency of transformation along these subsequences.

They are causal, and efficient (thanks to OMP); hence they have promise for interactive applications. They are adaptable, as penalties can be added for arbitrary properties.

In addition, lookahead heuristics can be used to increase track length, although this seems to be most effective for a small number of lookahead frames. Finally, a refinement was described for estimating exact sampling positions, which seems to counteract the problem of phasiness encountered when sampling using a regular grid in source position and transposition factor.

Chapter 7

Subjective Evaluation of Sound Mosaics using Listening Tests

7.1 Abstract

Listening tests are considered the gold standard methodology for evaluating and comparing different treatments of sound signals. Unfortunately, they are not often conducted, especially concerning experimental sound and music synthesis techniques. In this study, old and new approaches to the application of mosaicing synthesis were compared.

In the listening tests performed, own implementations of two classic algorithms: nearest-neighbor matching and dynamic programming path search (DP), were compared along with two new methods based on spectral mixtures: the first, which considers only mixtures at single frames, and the second, which encourages longer continuous sampling tracks. These algorithms were implemented in a common framework consisting of short fixed-duration synthesis frames, perceptually relevant spectral features, and resampling transformations (affecting pitch, timbre, and speed) of the input sounds, including controls over key synthesis parameters and their evolution. As well, the above methods were compared with Matching Pursuit (MP), a leading method for sparse signal decomposition, computed in the audio domain.

The test consisted of three parts. First, an identification task (ID): on a subset of isolated samples, subjects attempted to identify the source material being sampled and additionally rated samples on a single quality

scale. Second, a main rating task based on MUSHRA [ITU Radiocommunication Assembly, 2003]: subjects gave ratings to groups of related sound excerpts differing by algorithm. Rather than using a single quality scale, as is common in audio quality ratings, three quality dimensions were used, corresponding to similarity to the target in harmony (Q1), similarity to the target in timbre (Q2), and preservation of timbre characteristics from the source (the sampled excerpts, Q3) in the synthesized mosaics. Finally, an interview was conducted in which the subjects were free to comment on any of the testing material, the testing procedure, or the research itself.

Results: Matching Pursuit (MP) performed significantly best on the two target similarity measures (Q1 and Q2). However, on the measure of source timbre preservation (Q3), it performed significantly worst; and subjects were least able of all methods to correctly identify the source sound. In contrast, the continuity-encouraging mixture method (*tracks*) performed significantly best on Q3, was in the runner-up group for Q2, and had average performance on Q1. As well, the source signals from the mosaics created with *tracks* were highly identifiable. In general, mixture methods (including *mp*, *mix*, and *tracks*) seemed to dominate non-mixture methods, for all tasks except ID, in which the leading group included mixture and non-mixture methods. By contrast, while continuity preservation seemed to have helped *tracks* in Q3, it seemed to have cost performance in Q1. Additionally, MP and DP both took orders of magnitude more computation time than the other three causal methods.

7.2 Introduction

Sampling synthesis is the domain of audio synthesis that uses databases of sampled signals to produce a result. Mosaicing is a sampling synthesis technique that composites many small pieces of source material to produce a different result. One application of mosaicing is texture transfer, in which the textures of source sounds are transferred to musical targets, e.g. as an imitation or accompaniment. (See Chapter 2, State of the Art for more background).

In the course of the dissertation research, several new mosaicing strategies were developed: algorithms combining methods from sparse signal processing (supporting mixtures of several sounds at once to a spectral target) with perceptually relevant spectral descriptors, and automatic control of sound transformations. In order to validate and compare them to classical synthesis algorithms, an experiment was designed to measure the effect of the algorithm on several quality attributes of mosaics of identical input material. The experiment took the form of several listening tests and an interview.

The listening tests were designed to answer the following basic questions about the mosaics. First, could subjects identify which source sounds were used to generate individual mosaics? Second, to which extent did each mosaic sound similar to the musical or sonic target? Third, to which extent did each mosaic preserve sonic characteristics of the source? Synthesis algorithms that perform favorably according to these questions should be good candidates for texture transfer and mosaicing.

The mosaicing algorithms developed were intended to be used by music makers, composers, and sound designers. As potential users, subjects needed a certain cultural knowledge of harmony and timbre, i.e. not necessarily known to naïve listeners. Thus, listeners were recruited that had a certain amount of musical training.

The tests were conducted at Pompeu Fabra University (UPF), Music Technology Group (MTG) in Barcelona, during the week of Monday the 17th November to Friday the 21st (2014). Twenty subjects in total completed the test, which took around 2 hours per subject, and one additional subject completed just the individual mosaics and interview. All subjects had prior musical training and experience, many were associated with the MTG, and a few were associated with the Sonology Department of the Escola de Musica Superior de Catalunya (ESMUC).

7.3 Design of the Listening Tests

7.3.1 Hypothesis

As the design of our synthesis algorithms (treatments) addressed the conflicting requirements of spectral mixtures (allowing a better fit to harmony) and continuity (to preserve characteristics of the source, and to minimize the artifacts created in transformation and compositing), we wanted to know if blending these strategies would pay off. Therefore, the hypothesis was that a hybrid system would perform broadly well on the several quality measures, though not necessarily the best at any; in contrast to specialist algorithms that might perform well in one but not in others.

One example of a specialist system would be a standard dynamic programming path search algorithm, which maximizes continuity of the source (and therefore seem more natural) but cannot properly match mixtures, as it produces only a single path. A diametric example would be a mixture algorithm that matches spectral targets more closely, but without support for continuity.

In addition, it was hoped that mixture strategies would give improvements in target quality (harmony and timbre), and that continuity strategies would give improvements in preservation of the source quality. This would provide evidence for our and other future hybrid strategies.

7.3.2 Mosaicing Algorithms compared

group	**method**	**mixtures**	**continuity**	**causal**
Perceptual Filter (PF)	*near*	no	no	yes
	mix	yes	no	yes
	tracks	yes	yes	yes
	dp	no	yes	no
-	*mp*	yes	no	no

Table 7.1: Summary of methods tested with groups and capabilities.

The following mosaicing algorithms were compared:

The first group of algorithms were implemented in a common framework using descriptor vectors consisting of perceptual filter banks (chroma and timbre filters). These algorithms belong to the Perceptual Filter (PF) group:

1. *near* – Classic nearest-neighbor matching done on a frame-by-frame basis. Supports neither mixtures nor continuity.

2. *mix* – A Mixture method resembling non-negative matching pursuit done on a frame-by-frame basis. Supports mixtures but not continuity.

3. *tracks* – Hybrid tracks algorithm, heuristic algorithm allowing for mixtures and continuity. (*near* and *mix* are both implemented by this method with certain features turned off).

4. *dp* – My implementation of dynamic programming path search, supporting continuity but not mixtures.

The first three methods of the PF group (*near*, *mix*, and *tracks*) belong to a subgroup of algorithms that are causal (compute using only signal past). These methods were all implemented with different parameter variations using the TrOMPmos algorithm (see Chapter 6, Section 6.5.4, p. 110). *mix* ignores track-building in time, and so is equivalent to FrOMPmos (Section 6.5.3, p. 108).

Lastly, in contrast to the PF methods, the following method operated directly in the audio signal domain:

5. *mp* – Matching pursuit decomposition in the audio time domain, using a dictionary of analytic signal atoms extracted from the sources, as implemented by MPTK [Krstulovic and Gribonval, 2006]. This algorithm is not exactly a proper mosaicing algorithm, as it lacks a descriptor space as well as cost parameters relating to structure of the mosaic. However, it is capable of producing signal imitations based on mixtures to arbitrarily high SNRs, making it apt for comparison. *mp* supports mixtures but not continuity.

As mentioned in Section 2.1.4, *mp* is essentially equivalent to the strategy proposed in [Collins, 2012, Section 8.1, Sparse Concatenation], albeit with the following enhancement (as with the other methods).

In order to improve the similarity of a short source signal with the target, all algorithms were augmented with access to all transpositions from one octave down to one octave up, with three divisions per semitone, for 73 transpositions in total (including the identity). The utility of adding transpositions (or other transformations) to a small sound corpus to improve similarity to the target in descriptor space was discussed earlier (in Section 1.3, Introduction).

A summary of methods and groups by capabilities is given in Table 7.1.

Two additional non-algorithm conditions, *hidden reference* and *anchor*, are introduced by the MUSHRA quality tasks; see Section 7.3.5 below.

7.3.3 Test Protocol

Each test started with an overview of the test, describing the purpose of the experiment as well as the protocol with component parts (interview, individual mosaics, MUSHRA training, MUSHRA tests, exit interview).

Then, subjects were interviewed about their music training and past music activities, including previous experience with music technologies. Next, some basic mosaicing concepts like source and target signal were explained with the aid of several sound examples. The closed sound categories for identification were explained, and subjects could complete the ID task for individual examples in sequence (see Section 7.3.4 below). After the above steps were completed, the main quality rating task could proceed.

The bulk of the test consisted of the MUSHRA quality ratings (see Section 7.3.5). First, the three quality scales were defined. Then, the testing software was explained and subjects rated three screens (one for each scale) as training with the experimenter present. Finally, subjects completed the main ratings task, rating 18 screens in total. To close the experiment, subjects were interviewed about the test procedures, the testing material (whether they liked it or not), and any other comments they had about future applications in audio mosaicing.

7.3.4 Identification (ID) Task, individual mosaics

One main question of importance in this study (mentioned in the Introduction, first question) concerned how identifiable the source sounds were within the mosaics; and consequently how they were affected by treatments. This was determined by playing individual mosaics for the subjects, and having them guess for each mosaic what the source sound was.

Before the listeners were exposed to the source sounds, it was measured how well the sources could be identified in individual mosaics of differing treatment. A small subset of target/source signal pairs (hereafter referred to as *objects*) was prepared and processed over all algorithm conditions. To prevent subjects from recognizing a source signal from a previous listening, they rated only one of each of the five objects, and one of each of the five algorithm conditions.

The subjects listened to them, with repeats if necessary, and then completed two tasks relating to the source sound of the mosaic:

- closed selection from broad categories of sound: environmental, single instrument music, multiple instrument music, or non-music speech (see Questionnaire, Appendix A for details)
- open description: verbally identify, in broad and specific terms, the source sound

score component	**full credit**	**partial credit**
category	0.5 if correct	0.25 if one of several answers
description	0.5 if correct	0.25 if one of several answers

code - source	**ok categories**	**short correct description**
J – waterPour	ENV	water pouring
L – ducks	ENV/SPEECH	ducks mixed with human
M – tuning	POLY	an orchestra tuning
H – breathy	ENV/SPEECH	either noise or breath is ok
F – boccherini	MONO	a single violin playing

Table 7.2: Condensed scoring rubric used in assigning ID scores; along with correct categories and example descriptions; multiple in ok categories indicates either would be correct. Object codes correspond to those listed in Table 7.6, page 141.

Each trial was then graded on a scoring rubric, giving an ID score for each mosaic. Half a point was given for the correct category, and half a point was given for a correct description. If several conflicting options were given for category or description, one quarter point was given instead. Table 7.2 gives a condensed scoring rubric along with the correct answers for each source to identify.

The following design was used to distribute the conditions and presentation orders among objects and subjects (shown in Table 7.3). Between subjects, the mapping from objects to conditions never coincides in more than one mosaic. The absolute order of presentation (which mosaic is presented first, etc.) was balanced over both objects and conditions. The precedence order of presentation (which mosaic follows which other mosaic) was unbalanced but symmetric over S1-5 versus S6-10 in objects, and completely unbalanced and regular over conditions.

In addition to identifying the source sounds, subjects were also asked to rate the naturalness of each mosaic (from 1-least natural to 5-most natural) and to describe what informed their scores (artifacts, noise, control gestures). For details, see the Questionnaire, Appendix A (p. 167).

7.3.5 Source and Target Quality Tasks

The second and third research questions (from the Introduction) ask how similar the mosaic is to the target, and how well source characteristics are preserved in the mosaic. To this end, a listening test based on the MUSHRA

	subject number (in group of 10)									
	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
order	**object code and condition number**									
1st	F4	M1	J3	L5	H2	J4	M1	F3	H5	L2
2nd	H3	F5	M2	J4	L1	L3	J5	M2	F4	H1
3rd	L2	H4	F1	M3	J5	H2	L4	J1	M3	F5
4th	J1	L3	H5	F2	M4	F1	H3	L5	J2	M4
5th	M5	J2	L4	H1	F3	M5	F2	H4	L1	J3

Table 7.3: Ordered distribution of individual mosaics to ten subjects (top row, S1-S10) in presentation order (left column, from top to bottom). Each letter corresponds to a source-target pair in the testing material (given in Table 7.6, page 141), and each number corresponds to one of the five algorithm conditions (given in Subsection 7.3.2, but reordered).

standard (Multi Stimulus test with Hidden Reference and Anchor), given by International Telecommunications Union (ITU) standard ITU BS.1534 [ITU Radiocommunication Assembly, 2003], was implemented. This method was designed for "the comparison of high quality reference sounds with several lower quality test sounds". This seems appropriate for our task, as the distortions induced by mosaicing algorithms are quite strong compared to transmission or compression systems.

In this method, groups of test material are compared and rated on a common quality scale; the test works with the aid of interactive software that allows the subject/evaluator random access to any of the excerpts including an explicit reference (the target), allowing comparison between all excerpts within a group. The standard also defines a *hidden reference* (perfect quality) and *anchor* signals (highly distorted, low quality); these frame the quality of distortions introduced by the algorithms. The design of the anchor is specific to the distortions in question, and is addressed in its own section (Section 7.3.6.3, Anchor Sounds).

Rather than using a single quality scale as in similar listening tests (for instance, in transmission or compression systems), mosaics were evaluated on three different quality scales (referred to as "attributes" in the MUSHRA standard). This was based on two requirements. First, sound and music perception are multidimensional; two dimensions important for the similarity to the target are *harmony* (this will be scale Q1) and *timbre* (Q2).

Second, rather than transmission systems, which have a single input signal, mosaicing is a kind of cross-synthesis with two input signals: the *target* sound (what sound to imitate), and the *source* sound, where the samples and texture are derived from. We would like to know if the synthesis process transforms the source into something unrecognizable, or adds strong artifacts. Thus, the third quality scale (Q3) concerns the preservation of timbre characteristics of the source. See Figure 7.1 for a synthesis model that relates the three quality scales Q1-Q3 to the target, source, and mosaic signals.

While some aspects of *harmony* from the *source* can be carried over (e.g. the source only has certain intervals or chords present) this is not an explicit goal of the system and can be seen as a side-effect. As a sound is transferred into a new context, the absence or presence of artifacts could be seen as an aspect of realistic source timbre preservation.

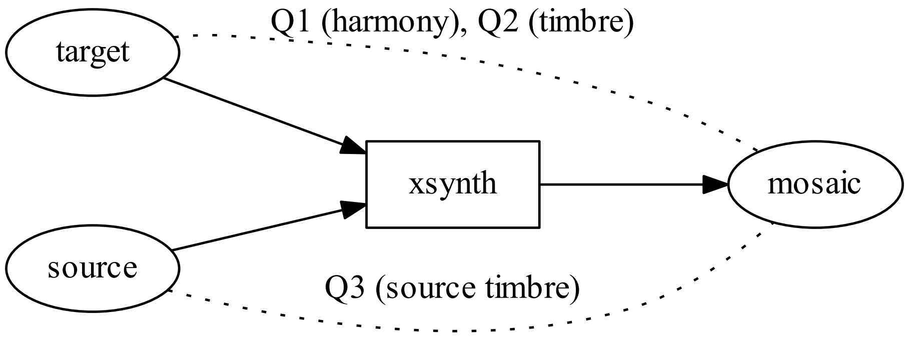

Figure 7.1: Conceptual model of a cross-synthesis process "xsynth" taking two input signals (target and source), and outputting a single mosaic signal. The dotted lines show the two quality scales concerned with the similarity between target and mosaic (Q1-harmony and Q2-timbre) and the scale concerned with similarity between the source and mosaic (Q3-timbre preservation).

In the testing interface, the following questions associated with each quality scale were asked:

Q1 How well is the *harmony* of the target/reference sound recreated?

Q2 How well is the *timbre* of the target/reference sound recreated?

Q1 and Q2 used the following scale, with descriptions on the top row (intermediate values shown as dashes on the test screen), and scores on the bottom row:

badly recreated	-	not so well recreated	-	average	-	well recreated	-	perfectly recreated
1	2	3	4	5	6	7	8	9

Q3 How well is the *timbre* of the source/reference sound preserved?

Q3 used the following scale:

badly preserved	-	not so well preserved	-	average	-	well preserved	-	perfectly preserved
1	2	3	4	5	6	7	8	9

In addition, the testing interface was translated into Catalan and Spanish for ease of use with potential evaluators, and was used for some quality ratings. The questions were translated as statements in Catalan: Q1-'El so recrea l'harmonia de l'objectiu/la referencia...', Q2-'El so recrea el timbre de l'objectiu/la referencia...', and Q3-'El so conserva el timbre del so de l'origen/la referencia...', with the following quality scale: 'malament', '-', 'no gaire bé', '-', 'més o menys', '-', 'força bé', '-', 'perfectament'.

Likewise, the questions were translated as statements in Spanish: Q1-'El sonido recrea la harmonia del objetivo/la referencia...', Q2-'El sonido recrea el timbre del objetivo/la referencia...', Q3-'El sonido conserva el timbre del sonido de la base/la referencia...', with the following quality scale: 'malamente', '-', 'no tan bien', '-', 'más o menos', '-', 'bastante bien', '-', 'perfectamente'.

In order to measure the consistency of ratings, some of the groups of material were retested a second time. Due to the large amount of test material and limited time per subject, not all of the material was retested. The retested ratings were used only to measure the consistency of the subjects; just the first test of each screen was used in the further analysis of ratings.

As it was not a main goal of the study to correlate the different attributes, and for reasons of test length, not all scales were uniformly tested on all of the test material. Rather, target material was selected with prominent characteristics in either harmony or timbre; the algorithms themselves had a parameter to favor either harmony or timbre. Yet for all targets, we found it relevant that the source characteristics should be preserved. Thus, the sets of material tested in Q1 and Q2 were disjoint, but Q3 included screens also tested on Q1 and Q2. (Which objects were tested on which scales is listed in Table 7.6, page 141).

The complete MUSHRA test was composed of screens in which a related group of excerpts were rated. Each group comprised a single object (with defined source and target) processed over all conditions (algorithms, anchor, and hidden reference). The training session was composed of three unique screens (not repeated later), one from each quality scale Q1-Q3. The test session was composed in blocks of a single quality scale at a time, starting with Q1, with four unique and two retested screens (total six); then with Q2, with three unique and one retested screen (total four); and finally Q3, with six unique and two retested screens (total eight). The order of screens within blocks was randomized for each evaluator, though in a few unanticipated cases (due to the test software used), repeated screens were presented back-to-back. In total, there were thirteen unique and five retested screens.

This test was implemented by configuring MUSHRA test software developed by Thomas Bisitz at Hörtech. This software uses buttons instead of sliders for the quality scales; it also supports multiple quality scales with changing color labels. An example screen from the test is shown and explained in Figure 7.2.

7.3.6 Selection and Preparation of the Test Material

In selecting test material, a range of input signals with different characteristics was sought: source sounds that would be challenging to identify under harsh treatments, target sounds with polyphonic harmonies that would show the benefit of new synthesis approaches, and difficult to imitate dynamic timbres to see what is possible with mosaicing. At the same time, it was desired to present interesting and entertaining material to subjects, who volunteered freely for a long test. Therefore, some earlier proposed objects (that weren't working in synthesis) were later omitted from the tests. To summarize, this process was both informal and subjective; some material was selected to test the benefits of new approaches and the limits of older ones; the tuning of parameters and final selection could have been prone to some bias.

Selection tasks are integral to sample-based sound design [Coleman, 2007]; not all sound transformations work well on all input material, partly necessitating the selections made. To ensure that each method could perform sufficiently, parameters for each object/method intersection were tuned, and these were reused over quality scales between Q1/Q2 and Q3 (excepting

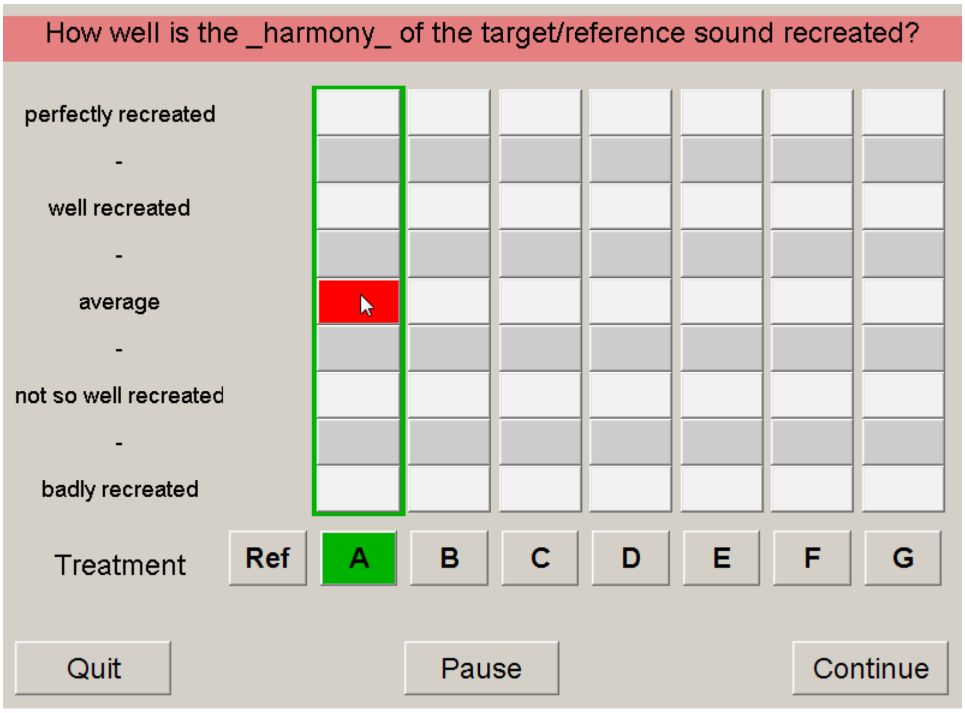

Figure 7.2: Screenshot (condensed) from MUSHRA implementation used in collecting quality ratings. At the top, the title shows the question relating to the current quality scale. At left, descriptions of quality gradations are given, with the corresponding ratings given to the right. On the bottom row are buttons corresponding to the treatments (including the reference signal, not rated); when selected, the button is highlighted green and the given excerpt is played, crossfading from the previously playing excerpt.

anchor signals, which were separately constructed according to the quality scale being tested).

The process for selecting and preparing the test material was roughly as follows: first, a pool of potential input sounds was selected to cover a wide range of categories of sound (next section, see Table 7.4, p. 140). Next, based on our knowledge of the material and algorithms, a partially specified list that identified potential target and source signals for different categories was proposed (see Table 7.5, p. 141). Then, potential source and target pairs (objects) were proposed and systematically tried out with the algorithms. During this process, parameters were tuned for the algorithms

(Section 7.3.2) to produce better sounding mosaics. Some objects, when no parameters were found producing good mosaics, were excluded. As well, three new input files were added at this stage (see last rows of Table 7.4, p. 140), giving the final selected target-source pairs (see Table 7.6, p. 141).

For the MUSHRA tasks, anchor signals were also generated (see Section 7.3.6.3, Anchor Sounds). Once all material was prepared, it was all normalized to the same RMS value, in order to have roughly comparable loudness.

7.3.6.1 Input Sounds

To begin, a set of input sounds were proposed coming from three broad categories: music signals, speech signals, and environmental sounds. Music signals were either polyphonic music, consisting of ensembles playing multiple parts or an instrument playing simultaneous notes, or monophonic music, with single instrumental lines. As speech, non-musical speech was specified. Finally, environmental sounds are predominantly neither music nor speech (though they occasionally might include one or the other), but are emitted from physical interactions in the world. Gaver's taxonomy of everyday listening [Gaver, 1993] helps to further divide simple sonic events from the environmental sounds into solid interactions (Vibrating objects), gas interactions (Aerodynamic sounds), and liquid interactions (Liquid sounds), each from which an input sound was selected. The initial input sounds are listed in Table 7.4.

7.3.6.2 Sound Mosaics (Algorithms)

The final target-source pairs selected are given in Table 7.6. The majority of the results used the following basic parameters at 44.1k sampling rate: a window size of 8193 samples (186 ms), and a hop size of 1024 samples (23 ms, hop rate of 43 per second) for 4x overlap between windows; except in some cases using the (madonna) targets, in which greater temporal resolution was sought; thus, a smaller hop size of 512 sample and 8x overlap was used.

The causal PF algorithms (*mix*, *near*, and *tracks*) all shared the same parameter set, consisting of the basic overlap-add parameters, and the cost parameters defined in Chapter 6, Section 6.4.1, page 101. The non-causal PF algorithm, *dp*, had a more limited parameter set for two reasons. First, it was not completely integrated into the framework of the others; and second, the full integration of some state elements is tricky (such as the source accumulator, described in Section 6.4.1.3, which maintains an accumulated

main category / name	**subcategory; description (length); tested?**
environmental	
bees	animal, air sound; binaural recording of buzzing bees (23s); Y
breathy	human-like, synthesized; breathy sounds generated from formant filters (15s); Y
inflation	air sound; someone blowing up a balloon (12s); Y
mixsteps	solid sounds; vibrating objects, keys, footsteps on floor (9s); N
waterPour	water sound; water poured into a filling container, some echo at the end (12s); Y
music	
8bit	synthesized instrumental; distinct 5ths tonality (14s); N
boccherini	bowed-string instr.; solo violin line from a minuet (15s); Y
brenda	human singing voice; female soprano singing a major arpeggio with nonsense words (5s); Y
fugue	piano instrumental; polyphonic (fugue) intro of 4 voices from the Art of the Fugue (Bach) w/ medium tempo (31s); Y
gradus	piano instrumental; fast passage from Debussy's Doctor Gradus ad Parnassum (17s); N
kidsing	human singing voice; kid singing song in Hindi (22s); Y
madonna	pop music; Madonna's "Give It 2 Me", chorus with voices, synthesizers, percussion, and harmonizing effects (15s); Y
madonnaS	short version of madonna, above (4s); Y
orchestra	orchestral; an orchestral texture in a minor key and fast waltz tempo (10s); Y
tuning	orchestral; tuning sequence with rich open chords (23s); Y
wordsFall	folk music; male and female voices singing in harmony with guitar and piano accompaniment, "Falling Slowly" from Once (7s); Y
speech	
challenge	male voice; Polish words, tough to pronounce (2s); N
exposure	female; voice gives a warning about loud sounds (8s); N
goodwill	male; president speaks about goodwill (7s); N
purenoise	female; says "this is pure noise". Sibilant, high-passed (1s); Y
environmental	
ducks	animal sounds and human voice; man imitating birds has a conversation with ducks (32s); Y
crickets	animal sounds; crickets and frogs on a summer night (81s); Y
music	
mandolin	plucked string; monophonic minor key melody played on mandolin (49s); Y

Table 7.4: List of sounds collected as potential input sounds, showing the main category, excerpt name, subcategories and text description (length), and whether or not included in final test material; the last three sounds were added in the final stage of parameter tuning/selection.

task	**source or target**	**suggested inputs w/ summarized reasons**
ID	source	boccherini, bees, tuning, breathy, mixsteps, waterPour
Q1 (harmony)	target	fugue (very hard polyphony), wordsFall (unison harmony), madonna, orchestra (minor harmony), gradus (difficult and quick)
Q2 (timbre)	target	purenoise (speech), orchestra (has interesting timbral rhythm, high and low instruments), inflation (imitation of a natural sound with clear dynamics), madonna (prominent vocal timbre)
Q3 (src. timbre)	source	boccherini, kidsing, goodwill, breathy, tuning

Table 7.5: Partially-specified list of ideas for proposed targets and sources for different tasks

object	**target**	**source**	**task**	**source anchor Q3**
A	fugue	boccherini	Q1, Q3	ducks
B	inflation	breathy	Q2, Q3	boccherini
C	madonna	breathy	Q2, Q3	mandolin
E	madonna	kidsing	Q3	brenda
F	madonnaS	boccherini	ID, Q1	NA
G	orchestra	boccherini	Q1, Q3	crickets
H	orchestra	breathy	ID, Q2 (training)	NA
J	orchestra	waterPour	ID	NA
K	purenoise	brenda	Q2	NA
L	wordsFall	ducks	ID, Q3 (training)	mandolin
M	wordsFall	tuning	ID, Q1 (training)	NA
N	wordsFall	bees	Q1, Q3	inflation

Table 7.6: Final list of target-source pairs (objects) along with the tasks they were used in the experiment; last column shows which alternate sources were used in generating anchors for Q3 task.

weight for source frames as they are sampled) without causing the state space, and consequently the computation time and space, to explode (the state space is exhaustively searched in standard dynamic programming).

The signal-domain approach, *mp*, was limited in several ways in order to produce results in a finite (but large) amount of time. First, the algorithm was run with a fixed number of iterations (n=1000). Second, the analytic signal dictionaries were limited to only the first 100 source frames, before transpositions (multiply by 73), and before translations to all given hops in the target (multiply by frames in target). The second measure was necessary because the dictionaries had become so large that MP seemed not to achieve any number of iterations in limited time. Finally, due to a rendering bug in MPTK, there was slight clipping in some of the outputs that was unable to be fixed before the test.

Neither of the restrictive measures nor the bug appear to have hindered *mp* significantly, as it produced the best approximations of target signals (Q1 and Q2; see Section 7.4.2).

7.3.6.3 Anchor Sounds

Anchor sounds are meant to provide a lower bound on MUSHRA quality measures. As such, they should be related to the distortion in question.

The anchor signals used for each task were created with the following procedures:

- Q1 (target harmony): A special version of the *near* method was programmed such that the chroma target was modified by shifting it by random step functions. It thus followed the chroma contours of the target, but with clearly wrong notes.
- Q2 (target timbre): Again, a version of the *near* method with a distorted target was used. In this case, both the chroma and timbre target signals were modified (as they are correlated) by speeding up the progression by 50%, and flipping the spectrum and inversing time. Therefore, the spectral and temporal shape of the target should have been quite different.
- Q3 (source timbre): For distorting the timbre of the source, the *near* method was used, changing the source input so it was a different source sound (thus not preserving the source in comparison to the reference).

7.4 Statistical Analysis and Results

The previous section (7.3) has described in detail the testing material and procedures. In this section, the statistical analysis and basic results are covered.

In designing the analysis, an earlier version of the MUSHRA standard was consulted [ITU Radiocommunication Assembly, 2003], that suggests estimating means and standard deviations of each group, and forming confidence intervals around those means for significance tests. However, a recent update of the standard gives considerably more detail to the question of statistical analysis, including a section recommending the parametric ANOVA [ITU Radiocommunication Assembly, 2014, Section 9.3, Application and usage of ANOVA, p. 15]; while the permutation test is recommended as a nonparametric alternative [ITU Radiocommunication Assembly, 2014, Attachment 3, Description of non-parametric statistical comparison between two samples using re-sampling techniques and Monte-Carlo simulation methods, p. 23].

Whereas general MUSHRA scores are defined on real number scales with many gradations, the push-button MUSHRA implementation used was confined to discrete scores (refer to Section 7.3.5, Figure 7.2). The ID scores and naturalness ratings were also given on discrete scales. As all scales were discrete, it would have been harder to justify a parametric approach. Therefore, nonparametric methods were used for the analysis, which could be less sensitive (would find fewer significant differences if the residual distributions were truly Gaussian) but do not make any distributional assumptions.

Rather than the permutation test, another standard nonparametric approach was chosen, that of methods based on rank statistics. In following, the responses were interpreted either as ordinal data (only ordered ranks of items matter) or interval data (degree of difference between items matters), using methods that compared either ranks of items or ranks of their differences. Where present (in the MUSHRA data), anchors and hidden reference groups were excluded from analysis, as they were artificially created to give the lowest and highest scores and give no information on the relations between the algorithms.

For each data type, a Kruskal-Wallis one-way anova was first conducted over the five algorithm conditions. This established if condition affected scores, by testing for stochastic dominance (ordinal analysis, if one group consistently beat any of the others in rank in the sample among all pairs of

data	**test type**	**test name**	**scale type**
ID scores and naturalness ratings	omnibus	Kruskal-Wallis one-way anova	ordinal
	post-hoc	Wilcoxon rank-sum test	ordinal
MUSHRA ratings	omnibus	Kruskal-Wallis one-way anova	ordinal
	post-hoc	Wilcoxon signed-rank test	interval
MUSHRA test-retested screens	correlation	Kendall τ correlation	ordinal

Table 7.7: Overview of statistical methods used in the analysis

ratings).

Afterwards, a post-hoc analysis was conducted to see which differences between pairs of groups were significant. For the identification and naturalness scores, which were measured independently for individual mosaics, a Wilcoxon rank-sum test was used (ordinal analysis) to compare pairs of conditions. By contrast, for the MUSHRA scores, which could be thought of as grouped (a single subject on a single screen rates a group of conditions by comparing between them), a Wilcoxon signed-rank test was used (interval analysis) to compare pairs of conditions. In both cases, to prevent false positives from multiple comparisons the Bonferroni correction was applied (n=10, for $\binom{5}{2}$ pairs of algorithm conditions), which could either be seen as dividing the target p-values or multiplying the observed p-values. Table 7.7 (p. 144) gives an overview of the methods used for the analysis.

Why did we analyze the individual mosaics only with ordinal methods, but analyze the MUSHRA data with more powerful interval methods? In the case of the individual mosaics, each evaluator rated five mosaics, each having differing object and method. Thus, there were no privileged paired comparisons among the individual ratings given by a single evaluator. Therefore, as in the ordinal methods, each rating is ranked against each other rating. However, the situation with MUSHRA ratings was the reverse. For example, when the evaluator or the object differed, we don't know if the same scale was used to give the ratings. But we do know that on each screen, the evaluator made direct comparisons of the mosaics on the same screen. In this case, the differences (and the intervals) between the methods can be analyzed directly (by the signed-rank test).

7.4.1 Individual Ratings (ID scores and naturalness ratings)

For the identification and naturalness rating tasks, individual mosaics were presented to the subjects (one for each of the five conditions). The ID scores are created by rating the subject's open identification of the sampled source and closed selection of a category for the sampled source. These were graded by rubric as earlier described in Section 7.3.4, giving a discrete distribution over scores [0, .25, .50, .75, 1]. The naturalness ratings were given on a scale from 1 (least natural) to 5 (most natural), but some subjects specified some ratings between points on the scale (in half-points) in order to be consistent with quality ratings given for previous excerpts.

Conditions and objects were distributed to subjects in balanced sets of 10 subjects (as described in Section 7.3.4; see Table 7.3, p. 134), each contributing two ratings for each mosaic; two balanced sets were measured, along with an extra subject, yielding 4-5 ratings for each mosaic (between 20-22 for each condition or object). This scheme gives fewer overall measurements than the MUSHRA data; however, clear patterns still emerged. Figure 7.3 shows the distribution of ID scores and naturalness ratings first by condition, and second by object (so that distribution by conditions can be compared with distribution by object) as box plots.

Most saliently, *mp* samples led to the worst identification of the source sound. A Kruskal-Wallis test found that condition has a highly significant effect on ID scores (H=25.0, df=4, $p < 0.01$). Post-hoc analysis using a Wilcoxon rank-sum test with Bonferroni correction (n=10) showed differences between (*dp*, *mix*, *tracks*) and *mp* and were all highly significant ($p < 0.01$) with (80%, 71%, 77%) of the pairs from the first group beating *mp*, and that differences between *near* and *mp* were significant ($p < 0.05$) with 73% favorable comparisons for *near*. Differences among the (*dp*, *mix*, *near*, *tracks*) group were however not significant (all $p > 0.05$).

As far as the naturalness scores, the *tracks* condition gave the highest absolute scores. A Kruskal-Wallis test found a significant effect of condition on naturalness scores (H=10.5, df=4, $p < 0.05$), but post-hoc analysis revealed only negligible effect sizes between pairs of groups.

By examining the ID and naturalness scores together, one can gain additional insights on their relationship. Figure 7.4 consists of heat maps, or 2D histograms, of the ID scores and naturalness ratings collected for all mosaics of each condition. One insight: in the *mp* condition (3rd column) along the

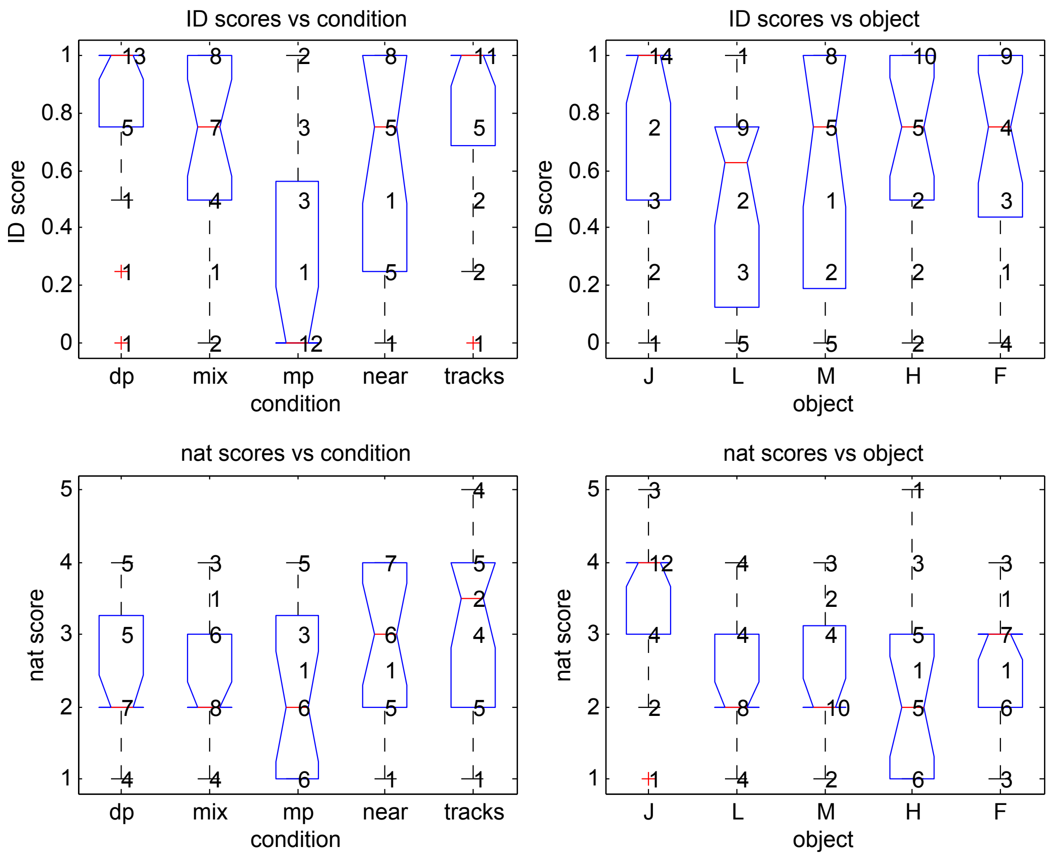

Figure 7.3: Box plots comparing ID (top, left) and naturalness scores (bottom, left) by algorithm condition. For each column, the actual score distributions are overlaid in text. Variation by object is shown (top and bottom, right) only for comparison.

bottom row, there is a series of mosaics which subjects rated naturalness higher than 1, but could guess no information correctly about the source (ID score=0). This situation is precisely reversed for *dp* (1st column) and *tracks* (5th column), in which there is a large row of mosaics with complete identification (ID score=1) but varying in naturalness rating.

The summaries of comment transcriptions for individual mosaics (see Table 7.8, p. 148) further emphasize differences in how identifiable the source sounds were. Note for the *mp* method (3rd row for each object) especially for some sources, in many cases subjects either misidentified the target as

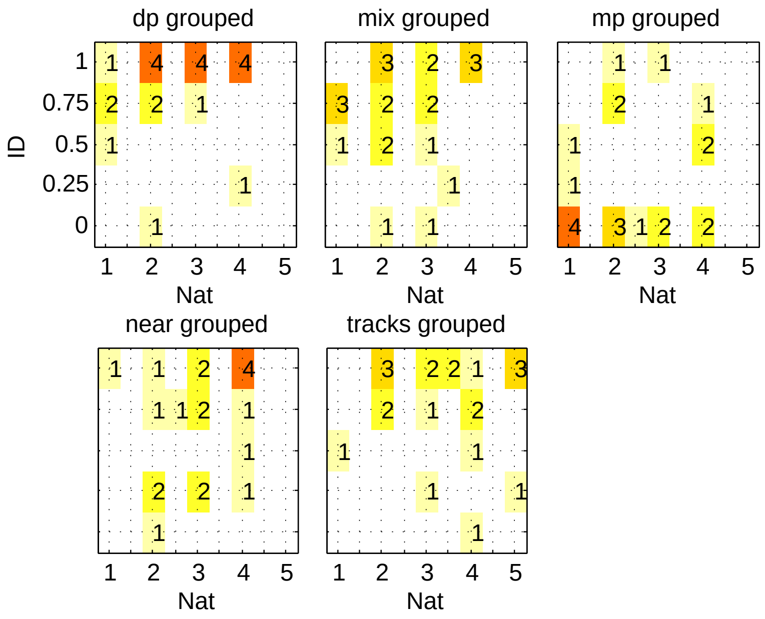

Figure 7.4: 2D histograms of ID scores and naturalness ratings grouped by condition. At each intersection, the count indicates the number of mosaics rated with that naturalness rating (x-axis) and ID score (y-axis).

the source, or stated that they couldn't identify the source, whereas for the other methods that was largely not the case. The table of qualitative comments (Table 7.9, p. 149) will be referenced in further sections to support characterization of the methods.

Comments were interpreted by the author and annotated in English, although they might have been spoken in Catalan or Spanish. The author maintains the recordings of the interviews.

7.4.2 MUSHRA Ratings (Q1, Q2, and Q3)

Data from the MUSHRA tasks could be interpreted as both ordinal data; or as interval data, in which each pair of two conditions given by a single user on a single screen forms a quality difference interval. The ratings were measured on differing scales (defined in Section 7.3.5, p. 133) according to questions (Q1, Q2, and Q3) from 1 to 9.

Figure 7.5 shows the distribution of Q1, Q2, and Q3 ratings by condition and object as box plots. Distributions by object are shown only for comparison

object:	**source** target
method (mean ID score %):	open identification
J:	**water pouring** orchestra
dp (100%):	“water being poured” “water in a tube” “water?” “water”
mix (95%):	“water in the toilet” “rippling liquid” “water poured in a jar” “water flowing from a pitcher or in nature” “water or fire”
mp (55%):	“running water” “water, keys, boot, or wooden instrument” “noise and tonal, metallic impact” “windy sound and knocking on wood” “organ or synth?”
near (69%):	“pouring water” “can’t ID source” “water opened tap, cleaning hands”
tracks (81%):	“water gargling and pouring” “water pouring” “water falling into a container” “festival organ”
L:	**ducks+human** folk song
dp (63%):	“could be anything” “animal sound” “like a child’s voice singing” “geese + goats + humans”
mix (75%):	“noisy, could be speech, ‘gr gr’ ” “speech” “transformed speech” “group of speakers”
mp (0%):	“a folk song” “string instruments” “bowed string” “piano, woman, man, other instrument”
near (31%):	“singing voice, young boy or soprano” “circus organ + noisy, non-tonal” “monkeys shouting” “speech, male voices” “human voice + synth”
tracks (69%):	“speech” “speech, mouth sounds” “man talking very fast” “male voice”
M:	**orch. tuning** folk song
dp (88%):	“string instruments” “jazz or metal wind instruments” “string instrument, tuning?” “strings and voice”
mix (55%):	“synthesizer, no attack, long decays” “symphonic music” “orchestra, strings and percussion” “stretched string instrument” “violin or machine-like”
mp (19%):	“singing voices and instruments” “can’t ID, not music or speech” “two people singing” “touching papers, can’t ID source well”
near (69%):	“orchestra, string ensemble, some percussion” “violin sound” “orchestra, can’t tell which instrument” “strings”
tracks (75%):	“creak (noise)” “baroque orchestra, strings, trumpet, drums” “orchestra” “tuning orchestra, can’t distinguish source from target”
H:	**breathy noises** orchestra
dp (75%):	“animal voice, like a cat” “breathing sound, snoring” “a child or a baby “can’t say source”
mix (50%):	“machine, like a vacuum cleaner” “group of people talking in a bar” “dog sound, dry and reverberated” “synth lead”
mp (63%):	“group of male singers or accordion” “noise is the source” “single voice, breathing” “might be natural sound or string”
near (94%):	“breathing, could be zombies” “hissing or noise + voice” “monkey or primate” “someone whispering” “animal or speech”
tracks (80%):	“male voice, unvoiced, background” “mouth sounds, snort, laugh” “human voice, breathing sounds” “human or animal voice, scratchy”
F:	**solo violin** Madonna pop
dp (90%):	“squeaky door/violin” “violin” “violin could be source or target” “wind instrument” “a violin”
mix (75%):	“strings” “violin” “strings” “string instrument, mistuned”
mp (0%):	“can’t hear source” “can’t ID source, distorted and noisy” “metallic percussion sound” “friction, object being dragged”
near (75%):	“strings and percussion” “violin” “synth or violin”
tracks (81%):	“violin, layered” “several violins, could be polyphonic source” “violin, with bow noise” “strings or accordion”

Table 7.8: Source open identification summaries for all subjects grouped by object, along with average ID scores for each condition (left side). ID scores were also based on closed selection of source category (not shown).

object: **source** target method (mean nat score %):	selected qualitative comments
J: **water pouring** orchestra	
dp (3.5):	“pitch altered. artifacts at the end, frequency is too high, sound is unnatural.” “seems natural, not strange, follows some dynamic changes, like tube modulation” “energy not directed, randomly constructed”
mix (3.2):	“can hear filtered sound, a lot of pieces that are connected slices repeated in the excerpt.” “echo is not natural. pitch changing indicates processing.” “a good example of how arbitrary sounds can be processed to make something else”
mp (3.2):	“sound is strange. rhythm seems unnatural or unexpected. missing the target rhythm. uncanny valley“ “tonally coherent, very natural sounds.” “doesn’t remind him of an instrument. old piano inside an automatic car. scratchy, like songs of tom waits”
near (4.0):	“notes sliced and transitions have sudden changes, not smooth. less natural, because musical content played in unnatural style” “changes pitch like a musical instrument, like bassoon or clarinet. can hear a tube character but also water”
tracks (4.5):	“can hear layered overdubbed sounds” “very natural, didn’t hear artifacts.”
L: **ducks+human** folk song	
dp (2.5):	“more artifacts than noise.” “not too many artifacts.” “exotic mixing, sounds like nature. sounds like a mashup of animals.”
mix (2.0):	“speech is pitched by tech, effect is very current. voice like a smurf.”
mp (3.0):	“like the sound with some added noise or like some of the instruments transformed (like a snare drum with a loose spring)” “has a filter, structure good, volume steady, there is interference and weird compression”
near (2.5):	“quite granular, more than macroscopic mosaicing” “tries to make a polyphonic texture, like target, but doesn’t succeed” “source seems less natural, doesn’t sound like they should“
tracks (2.0):	“kind of noisy” “velocity is unnatural” “cannot hear artifacts, changes are smooth. no distortions or loops. seems like instruments, maybe not natural. in some moments, fast changes could be seen as strange”
M: **orch. tuning** folk song	
dp (1.75):	“MP3 artifacts, like birdies” “I like the ‘flutter’ effect” “not so nice, not playing a melody, each note an unnatural change” “not flowing, discontinuous in intensity unnatural”
mix (2.3):	“high frequency rubbing sound is an artifact, or score would be 3.” “pitch shifting effect not natural for strings.” “not playing in a natural way, stretched sound.”
mp (2.0):	“original material [note: the target] easy to hear, but sounds boomy, like in a cave.”
near (3.5):	“sliding unnatural, rapid transitions from soft to percussive.” “noises appear and disappear without sense, disturbing noise bursts” “goes off pitch. swishing sound between notes.” “some transients made it sound artificial.”
tracks (3.13):	“heavily processed. target sound is not clear”
H: **breathy noises** orchestra	
dp (2.5):	“less natural because of artifacts” “could not be acoustic, not too many artifacts.” “mosaic is granulated. don’t hear a space.”
mix (1.5):	“comes from a broken amp/loudspeaker. perceived musical sense, but it sounds noisy, like something broken”
mp (1.5):	“clicks or artifacts, breathy from voice. can hear a coherent tune garbled” “noises between note changes that sound artificial” “transposition used to make it pitched, sounds artificial”
near (2.63):	“only part of the animal sound is present. doesn’t sound like natural sound.” “it sounds like a mix of two things with echo. hears loops for some reason, hears sonic repetition”
tracks (3.6):	“disagreeable, annoying, aggressive, danger”
F: **solo violin** Madonna pop	
dp (2.4):	“natural instrument playing with a lot of effort.” “synthetic. short rapid sounds, hard to play” “violin here is more human violin than before, but might be a synth”
mix (3.0):	“transitions between phrases, dips in amplitude, warble/tremolo” “no noisy cuts, sound is more continuous” “mistuning characteristics of strings. polyphonic effect”
mp (1.88):	“clicking, beats, background noise, and high frequency distortions” “low frequency bumps make the rhythm. strongly filtered, like a notch filter, very low and high pass“ “timbre metallic, scraping sound, can hear clipping”
near (2.25):	“a click, or some modulation, but more or less continuous, with some pattern. the impression of something played in reverse”
tracks (3.38):	“sounds like an unknown but imaginable instrument with several notes. not 5 because some effects being used”

Table 7.9: Selected qualitative comments on individual mosaics, grouped by object, along with average naturalness ratings for each condition (left side).

with the distribution by conditions; in general the distributions by object are more uniform and similar, showing a comparatively smaller effect by object.

On the target harmony task (Q1), the *mp* condition clearly gave the highest ratings. A Kruskal-Wallis test confirmed a highly significant effect of condition on Q1 scores (H=122.8, df=4, $p < 0.01$). Using Wilcoxon signed-rank tests with Bonferroni correction (n=10), all pairwise differences were significant ($p < 0.05$) except for (*mix*, *near*), giving the following partial order of groups by Q1 scores (from highest, descending): *mp*, (*mix*, *near*), *tracks*, with *dp* last.

On the target timbre task (Q2), the *mp* condition also gave the highest ratings. A Kruskal-Wallis test confirmed a highly significant effect of condition on Q2 scores (H=44.5, df=4, $p < 0.01$). The signed-rank test with Bonferroni correction found all pairwise differences highly significant ($p < 0.01$) except for the trio of (*dp*, *mix*, *near*) giving the following partial order of groups by Q2 scores (descending): *mp*, *tracks*, with the trio (*dp*, *mix*, *near*) last.

On the source timbre preservation task (Q3), the *tracks* condition gave the highest ratings. The Kruskal-Wallis test confirmed a highly significant effect of condition on Q3 scores (H=196.6, df=4, $p < 0.01$). The signed-rank test with Bonferroni correction found all pairwise differences highly significant ($p < 0.01$) except for the pair of (*mix*, *near*) giving the partial order of groups by Q3 scores (descending): *tracks*, (*mix*, *near*), *dp*, with *mp* last.

7.4.3 Summary of Post-hoc Results

The significant post-hoc results are summarized in a family of graphs (Figure 7.6, p. 158). For each data type, arrows are drawn showing stochastic dominance between conditions (relations are transitive). Naturalness ratings had no significant post-hoc group differences, and so were excluded. One salient point: there were no significant post-hoc differences for any task between *mix* and *near*. These results are interpreted in Section 7.4.5, Discussion of Method Results.

7.4.4 Retest Analysis

The underlying question of a retest analysis is: how reliable, or on the contrary, how random were the ratings given by the listeners? Given two vectors where the coordinates are ranked by order, the Kendall τ correlation [Kendall, 1938] measures how many orders between coordinates are preserved or inverted; where $\tau = -1$ means all orders are inverted, and $\tau = 1$ means all orders are preserved.

The Kendall τ was used to analyze the test/retest screens for all five retested MUSHRA screens (Section 7.3.5). Two measures were computed for each test/retest screen pair: τ_A, tau over all conditions (including hidden reference and anchor), and τ_C, tau just over the five algorithm conditions.

Figure 7.7 (p. 159) shows the distributions of τ_A and τ_C over all retest pairs, as well as averages for each subject. Starting with τ_A, we can see that for each subject screen, there were always more condition pairs (over all conditions) preserved than inverted ($\tau_A > 0$). Looking at τ_C, (over only algorithm conditions) we can see this is not the case; there seem to be four screen pairs with ($\tau_C < 0$) and several at ($\tau_C = 0$). Some of these inversions could be attributed to the 2-3 algorithm conditions (1-3 condition pairs) in the post-hoc results that were statistically insignificant according to Kruskall-Wallis (see summarized results in Section 7.4.3). However, when averaging by subject over retest screens, all subjects were consistent enough, by the criteria of preserving more algorithm orders than not ($\bar{\tau}_C < 0$).

We chose to retain all subjects with average $\tau_C > 0$, or who on average maintained more algorithm conditions in order than not. As all subjects added (on average) consistent information according to this criterium, no subjects were rejected. In addition, no individual screens were rejected to avoid unbalancing the data. (As mentioned in Section 7.3.5, only the first instances of each screen, and not the corresponding retested screens, were used in the analysis of the MUSHRA ratings in Section 7.4.2).

7.4.5 Discussion of Method Results

7.4.5.1 Matching Pursuit (*mp*)

As a technique for producing signal imitations using dictionaries of transposed source frames, *mp* mosaics performed the best on average (among all other methods) in quality scales Q1 and Q2 (recreation of target harmony and timbre).

However, the source qualities seem to have been poorly preserved by this approach, such that subjects had significant trouble identifying the sources (see Section 7.4.1); and that subjects rated *mp* excerpts, on average, the worst in quality scale Q3 (timbre preservation of source; see Section 7.4.2).

In the view of the author, a goal of mosaicing is to reproduce/imitate target characteristics, while still preserving some source characteristics (see Section 7.2, Introduction). But rather than preserving the source characteristics, the mosaics generated by *mp* seem to have preserved more of the target, with source characteristics poorly preserved as evidenced by the low ID scores and low Q3 ratings.

As a listener, my impression of the *mp* excerpts was this: they seem as if they were temporally precise copies of the target under a non-linear filtering process. This insight was partially supported by subject comments on mosaics from the ID task: when describing the quality of *mp* mosaics, some subjects used language consistent with filtering processes (see Table 7.9, p. 149, third row; “boomy”, “compression”, “strongly filtered”).

This temporal closeness with the target could be due to a predominance of phase in the objective function. In the matching pursuit family of signal decompositions, signal correlations are used to drive the optimization; this prioritizes phase over magnitude such that a phase inverted signal with the same magnitude spectrum would have negative correlation with the target, despite being similar by other measures, such as spectral shape measures (e.g. spectral moments) that disregard or discount phase.

When using dictionaries of sampled signals to recreate target sounds, matching pursuit can subtractively combine atoms to cancel frequencies present in the sources, and can even add arbitrary phase shifts to atoms in the case of analytic signal dictionaries. This subtractivity would seem to allow these approaches to allocate energy to frequencies co-occurring in sources and target, while suppressing leftover energy from used source atoms in frequency regions of low or no energy in the target.

Making reference to an unrelated phenomenon in cosmology, the term *dark energy* [Sturm et al., 2008b] was coined to refer to the property of atomic decompositions to create groups of imperceptible atoms compensating for error signals. One could coin a similar term, *spooky filtering*, to refer to the filter-like quality imparted by the process of reconstructing a target signal in the time-domain with sampled signal dictionaries. That is, the quality of imposing some subtle temporal-spectral qualities of the source (which ones exactly would be an interesting subject of further research), but mainly recreating the target characteristics. Kernel-based filters from image processing [Milanfar, 2013], being filters driven by a dictionary or corpus, also warrant some comparison and consideration in the further study of spooky filtering.

Due to the implementation of *mp* in this study, the atomic detail of reconstructions was dependent on target length. This was because a fixed budget of atoms was used for recreating all target signals, short or long. This resulted in low atomic detail for some long targets and high detail for some short targets. Alternatively, constant atom density per target duration could have been used. However, in that case, atoms would still be attracted to the most energetic temporal regions rather than being spread more uniformly in time, as was done with the structured sparsity approach of the proposed algorithms.

Another quality flaw that was noticed when recreating a slowly varying tonal target (fugue, object code A), was that the repetition of atoms in *mp* was more prominent than in *tracks*. This might have been caused by two possible factors. For one, all of the causal PF methods including *tracks* incorporated a cost penalizing recently used source frames (see Equation 6.22, p 103). As well, *tracks* searched for longer continuous sampling contexts, which might also have reduced repetition.

Finally, *mp* was one of the most computationally expensive methods; see Section 7.4.6 below.

7.4.5.2 Causal PF methods (*mix*, *near*, and *tracks*)

The mosaics generated by the causal methods from Group PF (*mix*, *near*, and *tracks*) seemed to have much in common, as evidenced by their quality ratings. In fact, ratings from *mix* and *near* were not statistically distinguished on any task. As a group, they were beaten by *mp* (Q1 and Q2) but beat *mp* in Q3. If we want to transfer or keep source qualities in the mosaic (Q3), the *tracks* method performed best. This is probably because *tracks* sampled longer continuous excerpts from the source material; otherwise the mosaics were generally similar to those of *mix* and *near*.

For the target similarity scales (Q1 and Q2), there appears to have been some trade-off. For the target harmony scale (Q1), *mix*/*near* seems to have beat *tracks*. Perhaps *tracks* overly constrained the atoms used to reconstruct the target, where it only allowed atoms that contribute to several target frames, whereas *mix* and *near* would have both made decisions based on the signal in each target frame (and so could perhaps better follow the spectral peaks of the target). In a more puzzling reversal, *tracks* beat *mix*/*near* in target timbre (Q2). It could be that the longer continuous tracks afforded by *tracks* were somehow more convincing as a timbre imitation than something with a more granular texture, often the case with *mix*/*near*.

7.4.5.3 Dynamic Programming method (*dp*)

Along with the Group PF methods, *dp* was more identifiable probably due to maintaining longer sampling tracks (as well as for the causal methods; not using spooky tricks, and heuristic methods for not reusing segments). On Q1 (harmony) *dp* performed the worst, perhaps because it was the most constrained, having to find a single global path through the material (and thus less able to hit local maxima like the causal PF methods). On Q2 (timbre) *dp* performed on par with *mix*/*near*; it might be that local maxima were less important than continuity for timbre perception.

Phase distortion (cancellation of out-of-phase sinusoids) seemed more prominent in *dp* than in the other Group PF methods. As the three methods: *dp*, *mix*, and *near* all sampled from discrete parameter spaces (sampling position and transposition), they all should have been prone to it; But among the three only *dp* maintains the sampling context, virtually assuring that the overlapped frames had common frequency support. Perhaps that is why *dp* performed worse than the other Group PF methods on Q3 (source timbre preservation). By contrast, *tracks* used a later developed refinement strategy: of sampling subsequent atoms of a track with continued values for the sampling position (Section 6.5.5.2) which should have prevented this distortion. A similar refinement could also potentially be applied to *dp*-like methods (albeit differently implemented).

7.4.6 Computational Effort

In addition to comparing the method results perceptual properties, we offer a brief comparison of computational performance of the methods. Timing averages are reported over twelve generated results for each method, run on the same cluster (HPC cluster at Pompeu Fabra University, DTIC). The quickest methods were *near* and *mix*, instantaneous methods (using no information out of the current frame), taking on average 2m53s and 3m46s per mosaic. The median performing method was *tracks*, which looks ahead some frames but is still causal, taking 15m36s on average per mosaic. The slowest methods were *mp* and *dp*, taking on average 8h33m and 9h08m per mosaic. These methods both operated over the whole signal.

MPTK [Krstulovic and Gribonval, 2006], the toolkit used to implement *mp*, is highly optimized, using several clever linear algebra optimizations to speed up computation [Sturm and Christensen, 2012]. However, we had to limit the dictionary size to get results in a reasonable amount of time (see Section 7.3.2). Most of our implementations using perceptual descriptors (*mix*, *near*, and *tracks*) were faster than the exact signal oriented approach of *mp*, partly because the dimensionality of the signals were greatly reduced prior to the main computations in the Group PF methods (from thousands of samples in a frame, to less than a hundred descriptors per frame), and partly because sub-problems for frames or groups of frames were considered in isolation in these methods.

7.4.7 Additional Feedback from Subjects

In general, most subjects were enthusiastic about the test material. One subject asked for a beta version of the software to aid musical composition; several afterwards said they would be interested in using it to compose if offered in a further evaluation. In addition, some selected their favorite excerpts from the MUSHRA testing material. However, one subject mentioned that many of the mosaics seemed far from the synthesis goal, and marked at least one mosaic (ID task) as “annoying”.

7.5 Conclusions

Regarding the main hypothesis, the hybrid system did indeed perform relatively well on a variety of quality measures, and outstandingly on Q3, which measured preservation of the source.

Regarding two auxiliary hypotheses: first, did mixture methods help to produce better target quality (harmony and timbre) mosaics? In support, *mp* (a mixture method) performed clearly the best on target quality measures. However, a mixture method, *tracks*, was beaten by a non-mixture method, *near* in Q1 (harmony), although this was reversed in Q2 (timbre); and the differences between *mix* (mixture) and *near* were not significant on target quality ratings. Perhaps then, the reason for the deficiency of *tracks* was due to its being constrained in time, as *near* and *mix* were not; or possibly due to the fact that solutions of *tracks* could have been non-optimal.

Mixtures clearly played a leading role in the success of *mp* in target quality measures; but part of that success must have relied on *spooky* effects such as cancellation of energy in unwanted frequencies. The mixture methods based on a perceptual filter bank domain (without the same capabilities afforded by *mp*), *mix* and *tracks*, may or may not had improved target quality, as the evidence was indecisive. *tracks* was also constrained in time due to continuity support, which may have limited the target quality gain from mixtures, and also used greedy solutions in time, which might have missed better solutions.

The second auxiliary hypothesis was: did continuity supported methods (*tracks*, *dp*) help to better preserve source quality in mosaics? The evidence in favor is strong, as *tracks* mosaics were highest in Q3 quality. In addition, *dp*, a weak method overall in quality, beat non-continuity method *mp* in Q3; although it lost to *near*. *dp* mosaics were subject to phase distortion, which should have also reduced the source quality, in opposition to any quality improvement from continuity. Therefore, it appears that continuity played a strong role in preserving source quality.

Regarding the several suppositions made in explaining the method results, in this and previous sections: in order to tease out the cause of target quality deficiencies in *tracks* (Q1 and Q2), further experiments could be done to see the effects of different parameter settings, and to test the optimality of the solutions given by *tracks*.

Our study showed performance gains for certain mixture methods over non-mixture methods for tasks related to target similarity; and showed that encouraging continuity can also improve quality attributes related to preserving source qualities. Given this evidence, we see the most promise for texture transfer applications, in methods that combine mixtures and continuity preservation.

7.6 Acknowledgments

Thanks to Matthias Vormann and Rainer Huber, who provided input on the experimental design; to Markus Meis who provided feedback on the questionnaire and suggested an oral interview component; and to Thomas Bisitz, who provided his implementation of button-based MUSHRA. Thanks to Iván Jiménez who helped by installing and facilitating MPTK on the HPC cluster in the Technology department (DTIC) at UPF.

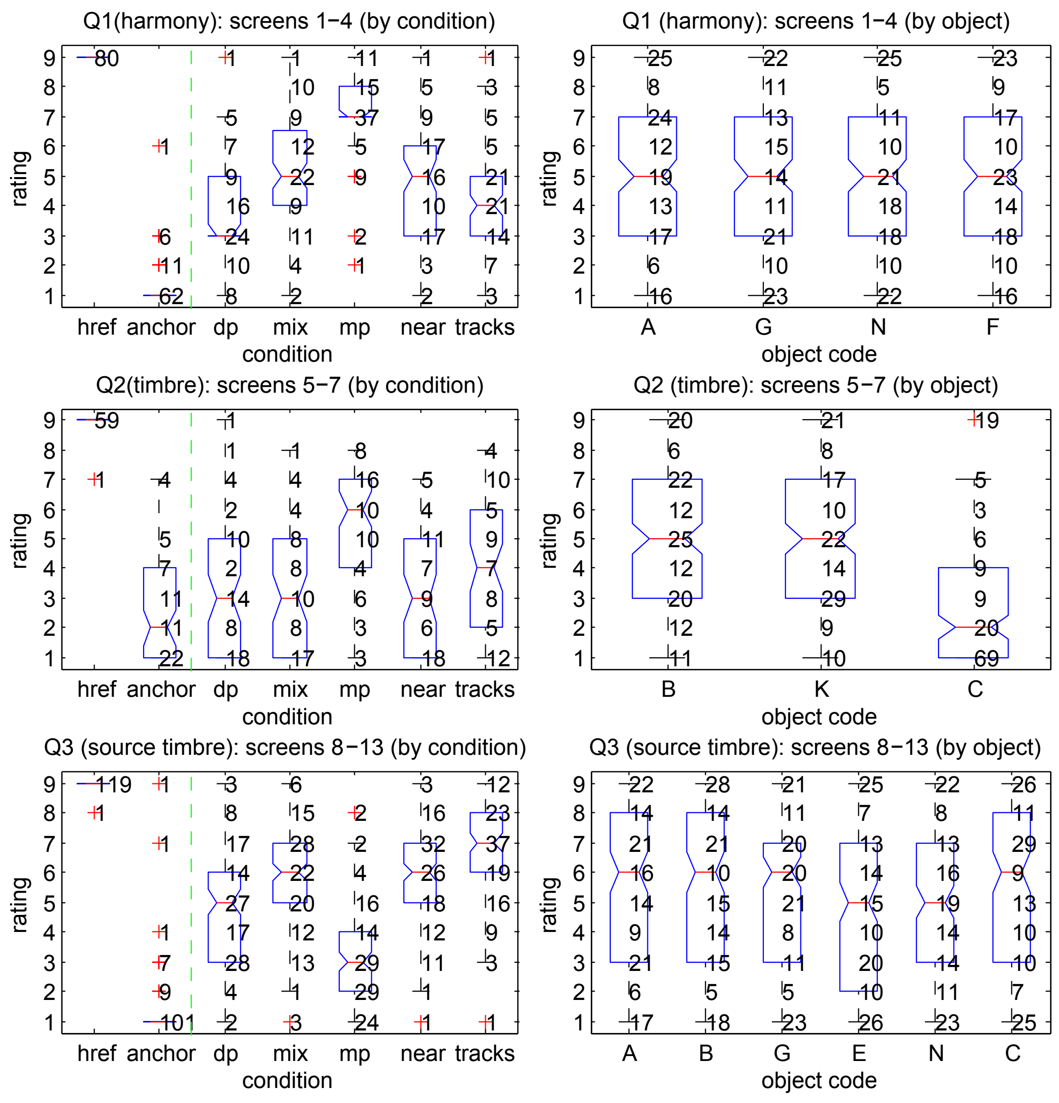

Figure 7.5: Box plots comparing grouped MUSHRA scores by condition (left column) for Q1 (top), Q2 (middle), and Q3 (bottom). Dotted green line divides the control conditions (left of line, excluded from analysis), from the algorithm conditions (right of line). For each column, the actual score distributions are overlaid in text. Variation by screens (objects) are shown (right column) only for comparison; see Table 7.6 for descriptions of objects.

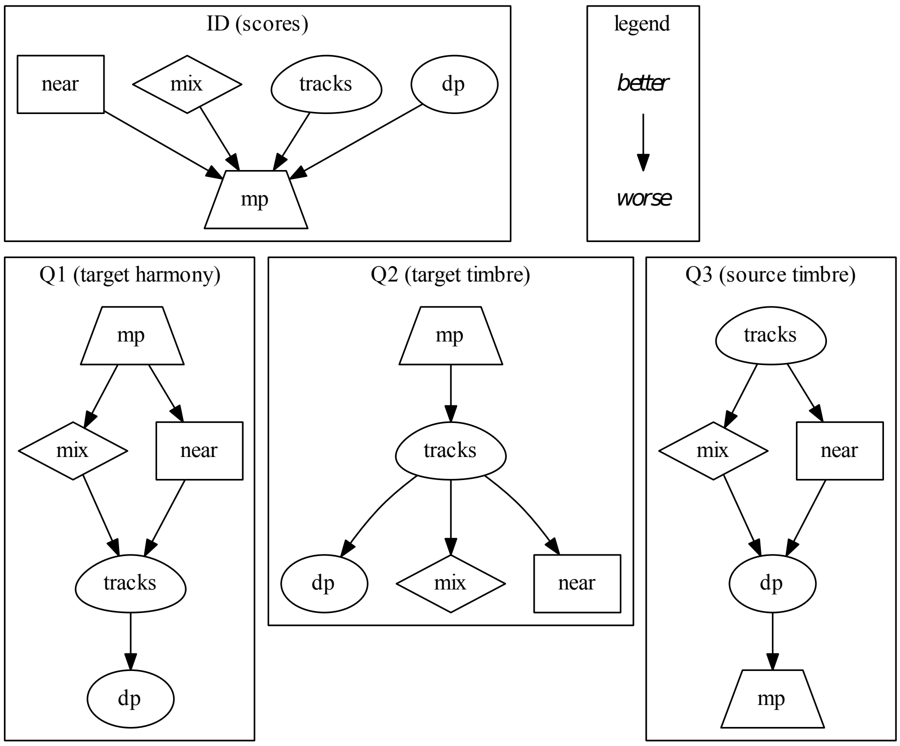

Figure 7.6: Significant post-hoc group differences for identification scores and MUSHRA ratings (Q1-3). Arrows point from higher quality to lower quality.

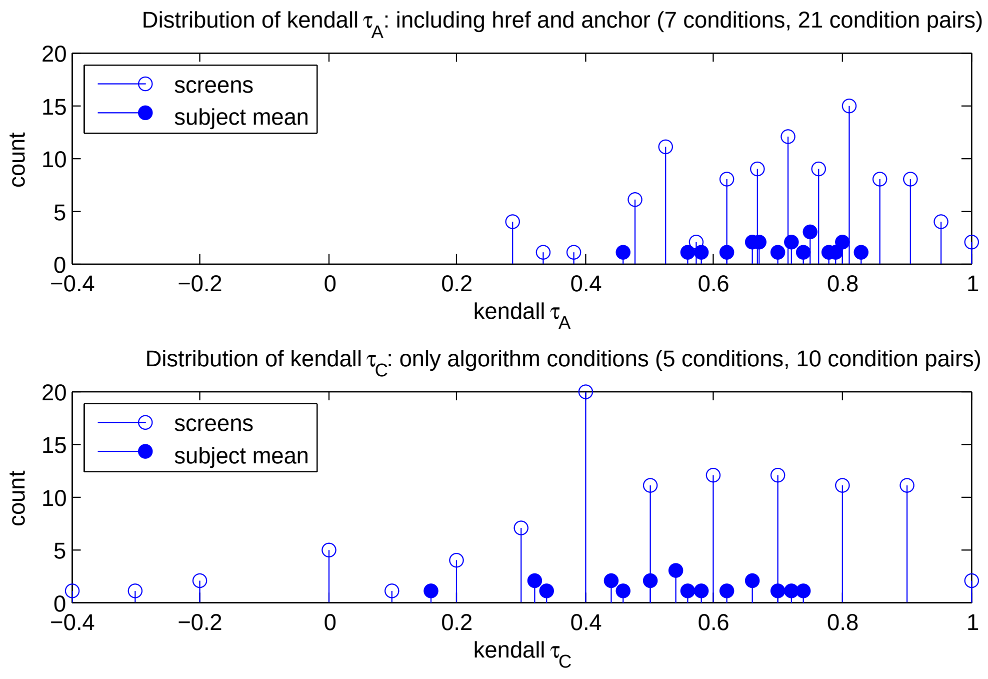

Figure 7.7: Kendall tau correlation distribution over all test/retest screen pairs; over all conditions (top) and over only algorithm conditions (bottom). Empty stems represent single test/retest screen pairs; filled stems are averages per subject over the five retested screens.

Chapter 8

Conclusion

In the research of this dissertation, the techniques of sampling synthesis were expanded in the following ways. First, predictive models of transformations were examined and a tentative framework concerning these models and their use was also proposed. Numerical experiments concerning predictive models were conducted, concerning descriptor-based automatic control and learning models from examples. The use of audio mixtures in mosaicing synthesis was also furthered, in particular, by proposing sparse decompositions in a perceptually motivated, lower dimensional space. This technique formed the basis of a mosaicing synthesis algorithm, which was evaluated using perceptual listening tests.

In the listening tests, the quality ratings corresponding to the hybrid method performed well on a variety of measures, including a measure of the source quality preservation (in which it performed best individually) and how well the source could be identified, although it did not perform individually best in other measures.

The main numerical problems encountered in this dissertation, those of transformation parameter selection and learning of predictive models from data, are prone to two curses of dimensionality (in the original sense of R. Bellman: that of function optimization, and additionally that of statistical learning). Knowing this and having a clearer view of the big picture, going forward I would place a higher emphasis on heuristic optimization methods and learning of predictive models with lower general complexity (for example, that include some element of linearity) than I have during the course of this research.

Although the research has proposed new methods, from a practical perspective several tasks remain to be done, such as: including enabling more transformations and allowing these to be controlled jointly, facilitating mosaicing with larger source databases, and finally implementing similar methods as real-time interactive instruments. The following section addresses these challenges.

8.1 Future Work

8.1.1 Model-based Descriptor Control of Transformations

One goal that needs work is to enable larger sets of transformations, along with relevant descriptor domains, to be used in descriptor-controlled transformations, and subsequently, in mosaicing systems like the one proposed.

A part of this goal is creating more models to support a desired set of descriptors and transformations, and solving the practical problem of how to efficiently learn predictive models from data (of appropriate complexity) would satisfy that goal. The SVR models tried in Ch. 5 were a proof of the viability of this concept, but they suffered from multiple practical issues, those being: they were less accurate than a simple derived model (perhaps due to the curse of dimensionality and the model class used), they had a cumbersome training procedure that took on the order of days (as a function of the data size needed), they stored multiple copies of essentially the whole training set which was already quite large.

In contrast, an ideal system would only need a linear or small polynomial number of examples in terms of the problem dimension, and should be able to train with only modest computational resources. An approach to finding the appropriate model class can be found in Section 5.7, p. 81.

The models in this dissertation have focused on spectral descriptors (for harmony, timbre, and spectral shape), but a general purpose system could include other descriptors such as roughness, and temporal or spectral modulation features as well.

As mentioned in Ch. 3 and demonstrated in 4, predictive models can be composed to model several effects in series. The second part of the goal of enabling more transformations is to deal with the complexity of optimizing a nonlinear function of many variables (the original curse of dimensionality). This means that grid search, the method used to minimize a function of two variables, source position and resampling factor, will be insufficient when more effects are added with accompanying "nonlinear" parameters. Moving from the exhaustive method previously used to a heuristic real optimization method, such as genetic algorithms (as employed in [Caetano and Rodet, 2009]) will probably yield efficiency improvements.

Perhaps there are additional criteria or priors that can reduce the search space, like a sparsity constraint preferring that only few effects are active at a time.

8.1.2 Mixture Mosaicing

In Chapter 6, causal greedy algorithms are presented that create mixture mosaic scores by optimizing over multiple criteria. However, it is not known how good these solutions are compared to the global optimum. Additionally, for synthesis applications not operating in real time, it might be interesting to use more heavy optimization methods to see how much the solutions can improve.

One technique for evaluating optimization algorithms is generating problems with known optimal solutions. In this case, this would mean to generate sparse sampling and transformation trajectories along with gains, and rendering them to a target signal, possibly with added noise (even though the results might not be that musical). Then, the mosaicing algorithms can be run to see how often and under which conditions these original trajectories are recovered. This would show how good the recovered scores are, even though in the general case of different source and target material, there might not exist sparse trajectories that recreate the target with low error.

A number of alternate approaches could be taken, in an effort to find more optimal solutions. As mentioned previously, the f_{mix} objective function defined in Equation 6.13 (page 99) is a mixed integer quadratic program (MIQP), for which toolkits (using the Branch and Bound strategy) are available, and should always find the global optimum, although in general have exponential complexity.

Besides the greedy methods used in the algorithms of Ch. 6, and the global methods discussed above, there are several other paradigms that may be used, outlined in the sections below.

8.1.2.1 Bayesian (Graphical) approaches

Another common way to describe an estimation problem is a maximization over probability distribution of possible solutions (e.g. a ML or MAP estimation). In some sense, this is equivalent to the cost function representation. However, rather than writing out a joint distribution explicitly, it is possible to represent probability distributions as graphs of conditional distributions, i.e. graphical models.

This permits more complex models to be expressed, as well as new optimization methods to be used. For example, in the system described in [Hoffman et al., 2009] and [Hoffman, 2010, Chapter 8], which had very similar goals to the one described in Chapter 6, Markov Chain Monte Carlo (MCMC) is used to sample from the posterior distribution over sources and shifts in sampling positions, giving an alternate way to find an optimal score. Variational Bayes methods [Fox and Roberts, 2011] which minimize a sequence of simplified point-wise models, are another technique for MAP estimation in graphical models. Sequential Monte Carlo (SMC) methods (such as Particle Filters) represent yet another approach to optimizing probabilistic models, one that is heuristic and causal, and that has been used in a number of tracking applications.

8.1.2.2 Structured Sparsity

The sparsity structure in many applications is better described by having a small number of related groups of atoms, rather than the having the absolute smallest number of atoms. This setting has been referred to as structured sparsity. For example, the setting described by the f_{mix} objective could easily be seen as structured sparsity: tracks are groups of atoms in contiguous sets of target frames, with each frame having a single atom, which are related to predecessor and successor atoms by distinct limits on changes in source position and transformation parameters.

One promising approach for structured sparse problems is the generic approximation algorithm of [Hegde et al., 2015]. Another consists of the "social sparsity" operators of [Kowalski et al., 2013].

8.1.2.3 Convex methods

In this approach, the combinatorial problem of selecting atoms is reformulated as a convex optimization problem, solved with local search, and some process like thresholding determines the combinatorial solution. These approaches to optimizing f_{mix} are outlined in Chapter 6, Section 6.7.

8.1.2.4 Approximate Dynamic Programming

See Appendix C.4, p. 175.

Finally, some practical improvements to our concrete methods are possible.

8.1.3 Improving the Efficiency of TrOMPmos

The most costly part of the OMP algorithm, in our case, was computing the correlations of all of the atoms with the target. This effort could be reduced by using an approximate method, such as an approximate k-Nearest Neighbors approach, to quickly find a few nearest neighbors in logarithmic time, perhaps like the one of [O'Hara and Draper, 2013]. [Vitaladevuni et al., 2011] and [Tjoa and Liu, 2011] are two such approaches in this direction.

Similarly, another way to reduce the load on the atom selection loop, would be to first cluster the atoms corresponding to transformed units, and then to select atom clusters instead of individual atoms in the SelectAtom routine. This would reduce the number of calculations for this step.

It has been shown that where sparse solutions exist, random projections, which effectively lower the dimensionality of the problem to be solved, can be performed under certain conditions without affecting the solution quality, the paradigm of which has been referred to as "compressed sensing" [Donoho, 2006]. This would further reduce the computational burden of the sparse approximation algorithm.

8.2 Summary

This dissertation contributes to the field of sampling synthesis and to the use of descriptor-controlled transformations therein.

1. It examines both relevant methods in sampling synthesis and descriptor-controlled synthesis and transformations in order to find gaps in the state of the art.

2. It proposes a framework which clarifies the role of predictive models in sampling synthesis.

3. It formulates a number of predictive models in various descriptor domains for resampling.

4. It validates a method of generic control for sound transformations using heuristic gradient-based methods.

5. It reports on initial experiments to learn predictive models from data, and examines shortcomings of the approach.

6. It proposes new hybrid methods of audio mosaicing that integrate perceptually relevant target descriptors, descriptor-controlled transformations, criteria traditionally important for sampling synthesis such as continuity, and audio mixtures of simultaneous source sounds.

7. It presents listening tests designed for the musical texture transfer application, that measure the quality of sound excerpts in several distinct quality dimensions, as well as other properties such as identifiability of the source sounds.

8. It evaluates the hybrid mosaicing methods along with classical path-based methods, simple mixture methods, and atomic signal approximation methods applied to mosaicing, using listening tests.

9. It proposes future directions for improving the current system.

Appendix **A**

Questionnaire

This form was used as a prompt for the subject interviews (initial and exit, Section 7.3.3, p. 131), as well as to sequence and record subject responses for the ID task (Section 7.3.4, page 132). It was also re-typeset to fit the dissertation format.

Do you have music training? How many years of training?

Do you play an instrument? What is your primary instrument? Do you sing?

Do you work in audio technology / mixing / music or sound production?

Introduction to Mosaics (play the example mosaics)

Mosaics are hybrid sound signals generated from a target signal and a source signal. The *source* signal is the material used to reconstruct the mosaic, such that the mosaic still retains some qualities, for example: timbre, of the source. The *target* signal is what the output should sound like, what the mosaic should try to imitate.

When asking about the quality of the *harmony* with respect to a target, we mean: are the notes correct? Are some notes missing? If the target signal has multiple melodies (polyphony) are they reproduced?

When asking about the quality of the *timbre* with respect to a target, we mean: if the target has low frequency and high frequency sounds, are their counterparts present in the mosaic? Are rhythms in timbre from the target reproduced?

Questionnaire, Page 2

Definitions of sound categories:

Environmental sound
– sounds generated by nature, or by simple object interactions

Monophonic music
– music with only single melodic line or note being played at a time

Polyphonic music
– music with multiple melodic lines or notes being played at a time

Non-music speech
– human speech outside of any musical context

Do you agree with these definitions for the purpose of the questionnaire?

Excerpt code 1:

Please circle the category of sound source:

Environmental sound / Monophonic music / Polyphonic music / Non-music speech

Describe, in broad and more specific terms, the sound source:

Please rate the naturalness of the generated sound:

1-least natural 2-less natural 3-average 4-more natural 5-most natural

Do you have any more comments about this example?

Questionnaire, Page 3

Excerpt code 2:

Please circle the category of sound source:

Environmental sound / Monophonic music / Polyphonic music / Non-music speech

Describe, in broad and more specific terms, the sound source:

Please rate the naturalness of the generated sound:

1-least natural 2-less natural 3-average 4-more natural 5-most natural

Do you have any more comments about this example?

Excerpt code 3:

Please circle the category of sound source:

Environmental sound / Monophonic music / Polyphonic music / Non-music speech

Describe, in broad and more specific terms, the sound source:

Please rate the naturalness of the generated sound:

1-least natural 2-less natural 3-average 4-more natural 5-most natural

Do you have any more comments about this example?

Questionnaire, Page 4

Excerpt code 4:

Please circle the category of sound source:

Environmental sound / Monophonic music / Polyphonic music / Non-music speech

Describe, in broad and more specific terms, the sound source:

Please rate the naturalness of the generated sound:

1-least natural 2-less natural 3-average 4-more natural 5-most natural

Do you have any more comments about this example?

Excerpt code 5:

Please circle the category of sound source:

Environmental sound / Monophonic music / Polyphonic music / Non-music speech

Describe, in broad and more specific terms, the sound source:

Please rate the naturalness of the generated sound:

1-least natural 2-less natural 3-average 4-more natural 5-most natural

Do you have any more comments about this example?

Appendix B

Sound Files

1. Phase artifacts from grid continuations, from Ch. 6, Section 6.5.5.2:
    a) phase-exact.wav
    b) phase-inexact.wav
    c) source: boccShort.wav
    d) target (source resampled up a major 2nd): boccShortUM.wav
2. Input sounds from Ch. 7 used from Freesound:
    a) bees - "20080127_1300_Panales_abejas_cerca.wav" from user "Manuel Calurano" link
    b) boccherini - "violin minuet_boccherini (edit).wav" from user "FreqMan" link
    c) kidsing - "2002_indien_kids_sing_8.wav" from user "dosa1" link
    d) inflation - "Balloon inflation.mp3" by user "Perry Duke" link
    e) waterPour - "water pour into jar.wav" by user "modcam" link
    f) tuning - "Orchestra Tuning.wav" by user "gelo_papas" link
    g) purenoise - "this is pure noise.wav" by user "epanody" link
    h) goodwill - "men of goodwill.wav" by user "ERH" link
    i) ducks - "Speaking with Ducks.wav" by user "Puzze Dao" link
    j) crickets - "FrogsAndCrickets_ExcerptB_JMA_24Bit_48k.wav" by user "greysound" link

   k) exposure - "ExcessiveExposure.wav" by user "acclivity" link

3. Demonstration sounds from Music Hacks, Appendix D:
   a) makeitrick-violin.mp3
   b) makeitmussorgsky-violin.mp3

4. Sounds used in listening tests of Ch. 7 (normalized.zip):
   a) AT.wav A1.wav A2.wav A3.wav A4.wav A5.wav AAH.wav AAS.wav AS.wav AT.wav
   b) BT.wav B1.wav B2.wav B3.wav B4.wav B5.wav BAS.wav BAT.wav BS.wav BT.wav
   c) CT.wav C1.wav C2.wav C3.wav C4.wav C5.wav CAS.wav CAT.wav CS.wav
   d) ET.wav E1.wav E2.wav E3.wav E4.wav E5.wav EAS.wav ES.wav
   e) exT.wav exM1.wav exM2.wav exM2b.wav exS.wav
   f) FT.wav F1.wav F2.wav F3.wav F4.wav F5.wav FAH.wav FS.wav
   g) GT.wav G1.wav G2.wav G3.wav G4.wav G5.wav GAH.wav GAS.wav GS.wav
   h) HT.wav H1.wav H2.wav H3.wav H4.wav H5.wav HAT.wav HS.wav
   i) J1.wav J2.wav J3.wav J4.wav J5.wav JS.wav JT.wav
   j) K1.wav K2.wav K3.wav K4.wav K5.wav KAT.wav KS.wav KT.wav
   k) L1.wav L2.wav L3.wav L4.wav L5.wav LAS.wav LS.wav LT.wav
   l) M1.wav M2.wav M3.wav M4.wav M5.wav MAH.wav MS.wav MT.wav
   m) NT.wav N1.wav N2.wav N3.wav N4.wav N5.wav NAH.wav NAS.wav NS.wav

The codes used in the filenames are the following: the first capital letter is the object code (Table 7.6, p. 141). The second string denotes T=target, S=source, AH=anchor harmony, AS=anchor source, AT=anchor timbre, and the numbers denote algorithm conditions (i.e. the order given by Table 7.8, p. 148). 'ex' denotes an example used to explain mosaics and quality scales.

Appendix C

Additional Background

C.1 The Curse of Dimensionality in Statistical Estimation

The "curse of dimensionality", originally coined by Bellman [as cited by Donoho, 2000], can apply to many different domains (e.g. optimization, search, machine learning, numerical integration, distance measures), but the common cause is differences in the nature of low and high dimensional spaces.

The curse hereto mentioned, the Curse of Dimensionality in Statistical Estimation, refers to the fact that in a quite general formulation of machine learning (in which it is only assumed the modeled functions are mathematically smooth), exponentially more examples are required as the number of input dimensions increases in order to guarantee the same error rate [Donoho, 2000, Section 6.2: Curse in Statistical Estimation].

The broad model class subject to this curse is thought to include many otherwise effective nonparametric models such as k-nearest neighbors, kernel density estimators, support vector machines (SVMs) with Gaussian kernels, Gaussian processes, and several manifold learning algorithms; what these algorithms have in common is that their models are mostly determined by training examples near the query point [Bengio et al., 2005]. Neural networks also fall into the class of models that approximate smooth functions in general, although it has been argued that some learning architectures, such as convolutional or pooled neural networks, provide additional structure that might help alleviate that curse [Bengio et al., 2013].

Now, it is not the case that all learning problems suffer exponentially, nor that learning problems with more dimensions are always harder. Rather, parametric models and regularized models, which both add information and structure to the learning process, allow generalizing with fewer examples (as long as their structural assumptions are met).

For example, it is shown that data generated by a linear plus noise model, in order to estimate linear model coefficients, the number of examples needed grows only linearly with the dimension, rather than the general case of needing exponentially more examples, as above [Hastie et al., 2009, Section 2.5: Local Methods in High Dimensions].

C.2 P versus NP

"**P** versus **NP**" refers to a yet unsolved, but important problem in worst-case complexity analysis of computational problems [Fortnow, 2009]. Using the formalism of decision problems (equivalent to yes-or-no questions) on Turing machines (a widely accepted, general, yet minimal formalism of computation), **P** is the class of all problems that can be solved in a number of steps polynomial in the input size. **NP** is the class of all problems that when given a solution, the solution can be checked in polynomial time.

The main question, whether these two classes of problems are different ($\mathbf{P} \neq \mathbf{NP}$) or the same ($\mathbf{P} = \mathbf{NP}$) is unresolved; but many computer scientists believe the former possibility over the latter [Gasarch, 2012].

NP-hard refers the class of problems that, if a polynomial time algorithm were found for an **NP**-hard problem, it could also be used to solve problems in **NP** in polynomial time (by way of reduction: a way to convert instances of **NP** problems into instances of the **NP**-hard problem). **NP**-complete refers to the **NP**-hard problems that are also in **NP** (are checkable).

This is mentioned in the thesis merely to establish a connection between the class of optimization problems discussed therein (sparse approximation problems) and standard computational complexity theory.

For a more in-depth look at these complexity classes, please see [Sipser, 1997, Part 3, Complexity Theory. Ch. 7, Time Complexity].

C.3 Regularization and MAP estimation

Often, it happens that there are several competing objectives in an optimization problem. The simplest way of combining multiple objectives is to add together their objective functions, like so:

$$\arg\min_x f(x) + \lambda g(x), \tag{C.1}$$

where x is the variable, and f and g are objective functions, and λ is a parameter expressing the relative importance of loss between f and g. This is referred to as regularization.

Beyond the generic use of combining multiple objectives, several specific uses for regularization are common. For example, if an optimization problem is ill-posed (meaning that many solutions are equivalent), an additional term can be added corresponding to the squared norm of the variables (referred to as Tikhonov regularization, or ridge regression), allowing a unique optimum to be found. Another use for regularization is in statistical learning, in which limiting the complexity of the chosen model (as in SVMs), is thought to help generalization and control overfitting, in which models attend to irrelevant details in the data.

In many regularized optimization problems, the component objective functions can be divided up into those that depend on some observed data, and those that do not, but express some general beliefs about the model shape. In the Bayesian framework, with components expressed as probability distributions, the former can be referred to as a likelihood distribution, and the latter as a prior distribution, and the regularized optimization as "maximum a posteriori" (MAP) estimation. Even without explicit probabilistic formulations, they can still be thought of as such.

Regularization in the context of approximation is discussed in [Boyd and Vandenberghe, 2004, Section 6.3.2, Regularization, p. 306].

C.4 Dynamic Programming

Dynamic Programming is a very general optimization strategy, formulated by Bellman [2003] based around recursively decomposing a larger optimization problem with many decisions into smaller interrelated problems.

This property was referred to by Bellman as the "Principle of optimality", here explained for the case of a process that takes different states in multiple time steps [—, 2003]:

> An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

Problems that support this property are referred to as having "optimal substructure".

Many useful algorithms are based on this principle, including path search/ Viterbi decoding. In these cases, the state space is finite or discretized, and the "value function" (in one formulation, defined as the minimum cost of arriving to state s on the N^{th} step) can be computed recursively using value function tables computed for the previous, $(N-1)^{\text{th}}$ step, over the entire state space (or some useful subset thereof). By enumerating the state space resulting from individual decisions and linking them (a trade-off in space), these methods avoid the combinatorial difficulty of enumerating all possible sequences (e.g.) of states.

For a guide to this approach in discrete optimization and many applications, see [Cormen et al., 2001, Ch. 15, Dynamic Programming].

Now, when mixtures are concerned, states involve a combinatorial superposition of atoms, making the above strategy, based on complete enumeration of the state space, prohibitively expensive in time and space. However, in many cases, there are still ways to approximate this value function without complete enumeration and using less space. These strategies have previously been applied in the field of reinforcement learning. This paradigm has been referred to as Approximate Dynamic Programming (ADP) [Powell, 2011].

ADP methods approach multiple decision problems as statistical learning problems. Although these methods have not been tried in this thesis, they offer yet another avenue of attack for synthesis methods based on sparse mixtures.

Appendix D

Related Music Hacks

Besides the main algorithms explored in Chapter 6, some small mosaicing scripts were prototyped using the Echo Nest's Remix API [The Echo Nest, 2009; Lindsay and Hutchison, 2009]. These prototypes were built and exhibited in the context of two Music Hack days hosted by the Music Technology Group.

The Remix API combines an analysis framework that subdivides music sound files into rhythmic units (tatums, beats, measures) as well as onset/event units, along with a programmatic interface to rearrange and combine these units, and several rendering engines that render to a sound or video file.

The two prototypes, developed in 2010 and 2011, apply the Basis Pursuit (BP) strategy described in Section 6.7.1, p. 119. The first hack, "Remix Sound Harmonizer: MakeItRick", approximates a target music sound file by projecting each non-uniform length target event unit onto a dictionary composed of all transpositions of the source units (implemented by resampling). The second hack, "MakeItMussorgsky", permits a score in midi format to be used as target, by artificially creating the target chroma vectors corresponding with the indicated notes.

Results are demonstrated by sound files "makeitrick-violin.mp3" and "makeitmussorgsky-violin.mp3", which use respectively as targets, Rick Astley's "Never Gonna Give You Up", and the Promenade from Modest Mussorgsky's "Pictures at an Exhibition".

Bibliography

Each reference indicates the pages where it appears.

A. Adler, V. Emiya, M. Jafari, M. Elad, R. Gribonval, and M. Plumbley. Audio Inpainting. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(3):922–932, Mar. 2012. 31

C. Ahlberg, C. Williamson, and B. Shneiderman. Dynamic Queries for Information Exploration: An Implementation and Evaluation. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 619 – 626, 1992. 24

X. Amatriain, J. Bonada, l. Loscos, J. L. Arcos, and V. Verfaille. Content-based Transformations. *Journal of New Music Research*, 32(1):95–114, Mar. 2003. 46

American Standards Association. *American standard acoustical terminology.* [New York], 1951. URL `http://archive.org/details/ameri00amer`. 95

D. Arfib and V. Verfaille. A-DAFx: Adaptive Digital Audio Effects. In *Proc. Workshop on Digital Audio Effects (DAFx-01)*, Limerick, Ireland, 2001. 46

M. Ashikhmin. Synthesizing natural textures. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, I3D '01, pages 217–226, 2001. ACM ID: 364405. 3, 4

F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Structured Sparsity through Convex Optimization. *Statistical Science*, 27(4):450–468, Nov. 2012. ISSN 0883-4237, 2168-8745. 92

C. Bachmann, H. Bischoff, M. Bröer, C. Kaboth, I. Mingers, S. Pfeifer, and B. Schütte. *Cubase 7 Manual: Plug-in Reference*. Steinberg Media Technologies GmbH, Dec. 2012. URL `https://www.steinberg.net/en/support/downloads/downloads_cubase_7.html`. 18, 20

M. Bartsch and G. Wakefield. To catch a chorus: using chroma-based representations for audio thumbnailing. In *Applications of Signal Processing to Audio and Acoustics, 2001 IEEE Workshop on the*, pages 15–18, 2001. 93

S. Becker, E. J. Candès, and M. Grant. TFOCS: Templates for first-order conic solvers, version 1.3. `http://cvxr.com/tfocs/`, 2014. 110

G. Beller, D. Schwarz, T. Hueber, and X. Rodet. A hybrid concatenative synthesis system on the intersection of music and speech. In *Journées d'Informatique Musicale (JIM)*, pages 41–45, 2005. 17

R. E. Bellman. *Dynamic Programming*. Dover Publications, Incorporated, 2003. ISBN 978-0-486-42809-3. 175

R. Bendiksen. *Digitale Lydeffekter*. Masters Thesis, Norwegian University of Science and Technology, 1997. 83

Y. Bengio, O. Delalleau, and N. Le Roux. The Curse of Dimensionality for Local Kernel Machines. Technical Report 1258, Dept. IRO, Université de Montréal, May 2005. URL `http://www.iro.umontreal.ca/~lisa/pointeurs/tr1258.pdf`. 173

Y. Bengio, A. Courville, and P. Vincent. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013. ISSN 0162-8828. 173

G. Bernardes, C. Guedes, and B. Pennycook. EarGram: An Application for Interactive Exploration of Concatenative Sound Synthesis in Pure Data. In *From Sounds to Music and Emotions*, number 7900 in Lecture Notes in Computer Science, pages 110–129. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-41247-9. 24

J. Bonada and X. Serra. Synthesis of the singing voice by performance sampling and spectral models. *Signal Processing Magazine, IEEE*, 24: 67–79, 2007. 21, 46

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University

Press, New York, NY, USA, 2004. ISBN 0521833787. 119, 175

M. Brudnak. Vector-Valued support vector regression. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 1562–1569. International Neural Network Society, July 2006. 69

J. J. Burred. A framework for music analysis/resynthesis based on matrix factorization. In *Proceedings of the International Computer Music Conference (ICMC2014)*, Athens, Greece, 2014. 23

M. Caetano and X. Rodet. Evolutionary spectral envelope morphing by spectral shape descriptors. In *Proceedings of the International Computer Music Conference (ICMC2009)*, Montreal, Canada, 2009. 30, 64, 163

M. Cardle, S. Brooks, Z. Bar-Joseph, and P. Robinson. Sound-by-numbers: motion-driven sound synthesis. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, page 356, 2003. 26

M. Casey. Soundspotting: A New Kind of Process? In *The Oxford Handbook of Computer Music*. Oxford University Press, Apr. 2011. 25

M. Casey and M. Grierson. Soundspotter and Remix-TV: fast approximate matching for Audio-Visual performance. In *Proceedings of the International Computer Music Conference (ICMC2007)*, Copenhagen, Denmark, 2007. 24

C.-C. Chang and C.-J. Lin. LIBSVM: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011. 70

S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001. 19, 119

P. Codognet and D. Diaz. Yet another local search method for constraint solving. In *Stochastic Algorithms: Foundations and Applications, Proceedings of International Symposium, SAGA 2001*. Springer-Verlag London, UK, Berlin, Germany, December 2001. 17

G. Coleman. Mused: Navigating the Personal Sample Library. In *Proceedings of the International Computer Music Conference (ICMC2007)*, volume 2, page 324, Copenhagen, Denmark, 2007. International Computer Music Association. 24, 93, 137

G. Coleman and J. Bonada. Sound transformation by descriptor using an analytic domain. In *Proceedings of the International Conference on Digi-*

tal Audio Effects (DAFx2008), pages 341–347, Espoo, Finland, September 2008. 30, 43

G. Coleman and F. Villavicencio. Predicting Transformed Audio Descriptors: A System Design and Evaluation. In *Proceedings of Workshop on Machine Learning and Music: MML10*, Florence, Italy, 2010. 30, 63

G. Coleman, E. Maestre, and J. Bonada. Augmenting sound mosaicing with Descriptor-Driven transformation. In *Proceedings of the International Conference on Digital Audio Effects (DAFx2010)*, Graz, Austria, 2010. 19, 20, 37, 89, 121

G. Coleman, J. Bonada, and E. Maestre. Adding dynamic smoothing to mixture mosaicing synthesis. In *Proceedings of 4th Workshop on Signal Processing with Adaptive Sparse Structured Representations (SPARS 2011)*, page 121, Edinburgh, Scotland, UK, June 2011. 89, 123

N. Collins. Even More Errant Sound Synthesis. In *Proceedings of the Sound and Music Computing Conference (SMC2012)*, volume 6, Copenhagen, Denmark, 2012. 22, 131

N. Collins and B. L. Sturm. Sound Cross-synthesis and Morphing Using Dictionary-based Methods. In *Proceedings of the International Computer Music Conference (ICMC2011)*, Huddersfield, UK, 2011. 22

A. Cont, S. Dubnov, and G. Assayag. Guidage: A fast audio query guided assemblage. In *Proceedings of the International Computer Music Conference (ICMC2007)*, Copenhagen, Denmark, 2007. 18, 19

T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001. ISBN 978-0-07-013151-4. 176

E. Costello, V. Lazzarini, and J. Timoney. A streaming audio mosaicing vocoder implementation. In *Proceedings of the International Conference on Digital Audio Effects (DAFx2013)*, pages 194–201, Maynooth, Ireland, Sept. 2013. 25

R. Dannenberg. Concatenative Synthesis Using Score-Aligned Transcriptions. In *Proceedings of the International Computer Music Conference (ICMC2006)*, pages 352–355, New Orleans, USA, 2006. 18, 20

M. Davies, A. M. Stark, F. Gouyon, and M. Goto. Improvasher: a real-time mashup system for live musical input. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 541–544, London, United Kingdom, 2014. 25

M. Dolson. The phase vocoder: A tutorial. *Computer Music Journal*, 10 (4):14–27, 1986. 29

D. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, Apr. 2006. ISSN 0018-9448. 165

D. L. Donoho. High-dimensional data analysis: The curses and blessings of dimensionality. Aide-Memoire of a Lecture at. In *AMS Conference on Math Challenges of the 21st Century*, 2000. URL `http://www-stat.stanford.edu/~donoho/Lectures/AMS2000/Curses.pdf`. 173

J. Driedger, T. Prätzlich, and M. Müller. Let It Bee – towards NMF-inspired audio mosaicing. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 350–356, Malaga, Spain, 2015. 23

R. O. Duda and P. E. Hart. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Communications of the ACM*, 15(1):11–15, Jan. 1972. 84

P. Dutilleux and U. Zölzer. Modulators and Demodulators. In *DAFX - Digital Audio Effects, edited by Udo Zölzer*. John Wiley and Sons, Ltd., 1st edition, 2002a. ISBN 0-471-49078-4. 82

P. Dutilleux and U. Zölzer. Nonlinear Processing. In *DAFX - Digital Audio Effects, edited by Udo Zölzer*. John Wiley and Sons, Ltd., 1st edition, 2002b. ISBN 0-471-49078-4. 83

A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *The Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999*, volume 2, pages 1033–1038 vol.2, 1999. 3

A. A. Efros and W. T. Freeman. Image Quilting for Texture Synthesis and Transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 341–346, New York, NY, USA, 2001. ACM. 3, 4

D. P. W. Ellis. PLP and RASTA (and MFCC, and inversion) in Matlab, 2005. URL `http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/`. Accessed: June 2010. 96

J. L. Flanagan, D. I. S. Meinhart, R. M. Golden, and M. M. Sondhi. Phase vocoder. *The Journal of the Acoustical Society of America*, 38:939, 1965. 29

L. Fortnow. The Status of the P Versus NP Problem. *Communications of the ACM*, 52(9):78–86, Sept. 2009. 174

C. W. Fox and S. J. Roberts. A tutorial on variational Bayesian inference. *Artificial Intelligence Review*, 38(2):85–95, June 2011. 164

T. Fujishima. Realtime chord recognition of musical sound: A system using common lisp music. In *Proceedings of the International Computer Music Conference (ICMC1999)*, volume 1999, pages 464–467, Beijing, China, 1999. 93

S. Fukayama and M. Goto. HarmonyMixer: Mixing the Character of Chords among Polyphonic Audio. In *Proceedings of the International Computer Music Conference (ICMC2014)*, Athens, Greece, 2014. 23

S. R. Garner. Weka: The waikato environment for knowledge analysis. In *In Proc. of the New Zealand Computer Science Research Students Conference*, pages 57–64, 1995. 70

W. I. Gasarch. Guest Column: The Second P =? NP Poll. *SIGACT News*, 43(2):53–77, June 2012. 174

L. A. Gatys, A. S. Ecker, and M. Bethge. A Neural Algorithm of Artistic Style. *arXiv:1508.06576 [cs, q-bio]*, Aug. 2015. URL `http://arxiv.org/abs/1508.06576`. arXiv: 1508.06576. 3, 4

W. W. Gaver. What in the world do we hear? An ecological approach to auditory event perception. *Ecological Psychology*, 5:1–29, 1993. 139

E. Gómez. *Tonal Description of Music Audio Signals*. PhD thesis, Universitat Pompeu Fabra, July 2006. URL `http://mtg.upf.edu/node/472`. 93, 94

A. Gretsistas and M. D. Plumbley. A Multichannel Spatial Compressed Sensing Approach for Direction of Arrival Estimation. In V. Vigneron, V. Zarzoso, E. Moreau, R. Gribonval, and E. Vincent, editors, *Latent Variable Analysis and Signal Separation*, number 6365 in Lecture Notes in Computer Science, pages 458–465. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15994-7. 31

R. Gribonval and E. Bacry. Harmonic Decomposition of Audio Signals with Matching Pursuit. *IEEE Transactions on Signal Processing*, 51(1):101–111, 2003. 31

S. L. Groux and P. F. Verschure. Perceptsynth: mapping perceptual musical features to sound synthesis parameters. In *IEEE International Conference on Acoustics, Speech and Signal Processing, 2008.*, pages 125–128, March 2008. 64

S. Handel. *Listening: An Introduction to the Perception of Auditory Events.* MIT Press, Jan. 1989. ISBN 978-0-262-08179-5. 95

M. Haro Berois. *Statistical distribution of common audio features: encounters in a heavy-tailed universe.* PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, Nov. 2013. 7

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer-Verlag New York, 2nd edition, 2009. 174

S. Hazel. soundmosaic (website), 2001. URL `http://awesame.org/soundmosaic/`. Accessed: May 2015. 18, 20

D. J. Heeger and J. R. Bergen. Pyramid-based Texture Analysis/Synthesis. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 229–238, New York, NY, USA, 1995. ACM. 2, 3

C. Hegde, P. Indyk, and L. Schmidt. A Nearly-Linear Time Framework for Graph-Structured Sparsity. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 928–937. JMLR Workshop and Conference Proceedings, 2015. 164

M. Hoffman. The FeatSynth Tutorial - Part 1, 2007. URL `http://featsynth.cs.princeton.edu/documentation/tutorial1.html`. 28

M. Hoffman. *Probabilistic Graphical Models for the Analysis and Synthesis of Musical Audio.* PhD thesis, Princeton, 2010. 164

M. Hoffman and P. Cook. The featsynth framework for feature-based synthesis: Design and applications. In *Proceedings of the International Computer Music Conference (ICMC2007)*, volume II, pages 184–187, Copenhagen, Denmark, 2007. 28, 47, 64

M. D. Hoffman, P. R. Cook, and D. M. Blei. Bayesian spectral matching: Turning young MC into MC hammer via MCMC sampling. In *Proceedings of the International Computer Music Conference (ICMC2009)*, Montreal, Quebec, Canada, 2009. 18, 20, 164

C. W. Hsu, C. C. Chang, and C. J. Lin. *A Practical Guide to Support Vector Classification.* http://www.csie.ntu.edu.tw/˜cjlin/libsvm/, 2003. 69

A. Hunt and A. Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing, 1996. ICASSP-96.*

Conference Proceedings, volume 1, pages 373–376 vol. 1, May 1996. 17

S. Ingram and P. Bhat. Automatic Collage Using Texture Synthesis. In *Smart Graphics*, volume 3031/2004, pages 140–145. Springer, 2004. 3

ITU Radiocommunication Assembly. 1534-1: Method for the Subjective Assessment of Intermediate Quality Level of Coding Systems, Jan. 2003. URL `http://www.itu.int/rec/R-REC-BS.1534-1-200301-S/en`. 128, 134, 143

ITU Radiocommunication Assembly. 1534-2: Method for the Subjective Assessment of Intermediate Quality Level of Coding Systems, June 2014. URL `http://www.itu.int/rec/R-REC-BS.1534-2-201406-I/en`. 143

J. Janer. *Singing-driven Interfaces for Sound Synthesizers.* PhD thesis, Universitat Pompeu Fabra, Barcelona, 2008. 28

J. Janer, M. Haro, G. Roma, T. Fujishima, and N. Kojima. Sound Object Classification for Symbolic Audio Mosaicing: A Proof-of-Concept, 2009. 25

T. Jehan. *Perceptual synthesis engine: an audio-driven timbre generator.* Masters Thesis, Massachusetts Institute of Technology, 2001. URL `http://opera.media.mit.edu/papers/Masters_Tristan.pdf`. 27

T. Jehan. Event-synchronous music analysis/synthesis. In *Proceedings of the International Conference on Digital Audio Effects (DAFx2004)*, Naples, Italy, 2004. 18, 19

T. Jehan and B. Schoner. An Audio-Driven Perceptually Meaningful Timbre Synthesizer. In *Proceedings of the International Computer Music Conference (ICMC2001)*, volume 2001, Havana, Cuba, 2001. 27

M. G. Kendall. A New Measure of Rank Correlation. *Biometrika*, 30(1/2): 81–93, June 1938. 151

S. Kersten and H. Purwins. Sound Texture Synthesis with Hidden Markov Tree Models in the Wavelet Domain. In *Proceedings of Sound and Music Computing Conference (SMC2010)*, Barcelona, Spain, 2010. 3, 5

S. Kersten and H. Purwins. Fire Texture Sound Re-Synthesis Using Sparse Decomposition and Noise Modelling. In *Proceedings of the International Conference on Digital Audio Effects (DAFx2012)*, York, UK, 2012. 3

M. Kowalski, K. Siedenburg, and M. Dorfler. Social Sparsity! Neighborhood Systems Enrich Structured Shrinkage Operators. *IEEE Transactions on Signal Processing*, 61(10):2498–2511, May 2013. 164

S. Krstulovic and R. Gribonval. MPTK: Matching Pursuit made Tractable. In *Proc. Int. Conf. Acoust. Speech Signal Process. (ICASSP'06)*, volume 3, pages III–496 – III–499, Toulouse, France, May 2006. 131, 154

V. Kwatra, A. Schodl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics*, 22(3):277–286, 2003. 3

B. Latour. *Science in Action, How to Follow Scientists and Engineers through Society.* Harvard University Press, Cambridge, Mass., USA, 1987. 2

A. Lazier and P. Cook. MOSIEVIUS: feature driven interactive audio mosaicing. In *Proceedings of the International Conference on Digital Audio Effects (DAFx2003)*, Queen Mary, University of London, 2003. 24

S. Le Groux and P. Verschure. Perceptsynth: mapping perceptual musical features to sound synthesis parameters. In *IEEE International Conference on Acoustics, Speech and Signal Processing, 2008. ICASSP 2008*, pages 125–128, Mar. 2008. 27

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, Oct. 1999. 22

A. Lindsay and D. Hutchison. Fluently Remixing Musical Objects with Higher-order Functions. In *Proceedings of the International Conference on Digital Audio Effects (DAFx2009)*, Como, Italy, Sept. 2009. 177

S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993. 22, 104

P. Milanfar. A Tour of Modern Image Filtering: New Insights and Methods, Both Practical and Theoretical. *IEEE Signal Processing Magazine*, 30(1): 106–128, Jan. 2013. 152

D. Mintz. *Toward Timbral Synthesis: a new method for synthesizing sound based on timbre description schemes.* Masters thesis, University of California, 2007. 27

P. K. Mital. Memory Mosaic iOS – Generative Audio Mashup App | pkmital.com, Aug. 2015. URL `http://pkmital.com/home/2015/08/23/memory-mosaic-ios-generative-audio-mashup-app/`. 24

B. C. Moore and B. R. Glasberg. Suggested formulae for calculating auditory-filter bandwidths and excitation patterns. *The Journal of the*

Acoustical Society of America, 74(3):750–753, 1983. 96

M. Müller and S. Ewert. Chroma Toolbox: MATLAB implementations for extracting variants of chroma-based audio features. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, Miami, USA, 2011. 93

Music Technology Group. Freesound (website). Universitat Pompeu Fabra, 2005. URL `http://freesound.org/`. Accessed: June 2008. 58

T. Nakano and M. Goto. VocaListener: A singing-to-singing synthesis system based on iterative parameter estimation. In *Proceedings of the Sound and Music Conference (SMC2009)*, pages 343–348, Porto, Portugal, 2009. 28

B. K. Natarajan. Sparse Approximate Solutions to Linear Systems. *SIAM Journal on Computing*, 24(2):227–234, Apr. 1995. 101

J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, 1999. 55

S. O'Hara and B. A. Draper. Are you using the right approximate nearest neighbor algorithm? In *2013 IEEE Workshop on Applications of Computer Vision, WACV*, pages 9–14, Clearwater Beach, FL, USA, January 2013. IEEE Computer Society. 165

A. Olivero, B. Torresani, and R. Kronland-Martinet. A Class of Algorithms for Time-Frequency Multiplier Estimation. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(8):1550–1559, Aug. 2013. 30, 37

D. O'Shaughnessy. *Speech communication: human and machine.* Addison-Wesley Pub. Co., 1987. ISBN 9780201165203. 97

T. H. Park and Z. Li. Not just prettier: FMS marches on. In *Proceedings of the International Computer Music Conference (ICMC2009)*, Montreal, Quebec, Canada, 2009. 29

T. H. Park, J. Biguenet, Z. Li, C. Richardson, and T. Scharr. Feature modulation synthesis (FMS). In *Proceedings of the International Computer Music Conference (ICMC2007)*, Copenhagen, Denmark, 2007. 29

T. H. Park, Z. Li, and J. Biguenet. Not just more FMS: taking it to the next level. In *Proceedings of the International Computer Music Conference (ICMC2008)*, Belfast, Northern Ireland, 2008. 29

Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Conference Record of The Twenty-Seventh Asilomar*

Conference on Signals, Systems and Computers, volume 1, page 40–44, 1993. 105

G. Peeters. A large set of audio features for sound description (similarity and classification) in the CUIDADO project. CUIDADO IST Project Report, IRCAM, Analysis/Synthesis team, Paris, France, 2004. URL `http://recherche.ircam.fr/anasyn/peeters/ARTICLES/Peeters_2003_cuidadoaudiofeatures.pdf`. 48

M. Plumpe, A. Acero, H.-W. Hon, and X. Huang. HMM-Based Smoothing for Concatenative Speech Synthesis. In *Proc. of the Int. Conf. on Spoken Language Processing*, Dec. 1998. 21

W. B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons, 2 edition, Oct. 2011. ISBN 978-1-118-02916-9. 176

N. Saint-Arnaud and K. Popat. Analysis and synthesis of sound textures. In D. F. Rosenthal and H. G. Okuno, editors, *Computational Auditory Scene Analysis*, pages 293–308. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1998. 4

B. Scharf. Fundamentals of auditory masking. *Audiology: Official Organ of the International Society of Audiology*, 10(1):30–40, Feb. 1971. 96

J. F. Schouten. The Perception of Timbre. In *Reports of the 6th International Congress on Acoustics*, volume 1, pages 35–44, Tokyo, 1968. Elsevier. 95

D. Schwarz. A system for data-driven concatenative sound synthesis. In *Proceedings of the International Conference on Digital Audio Effects (DAFx2000)*, page 97–102, Verona, Italy, 2000. 16

D. Schwarz. The caterpillar system for data-driven concatenative sound synthesis. In *Proceedings of the International Conference on Digital Audio Effects (DAFx2003)*, page 135–140, Queen Mary, University of London, 2003. 17

D. Schwarz. *Data-Driven Concatenative Sound Synthesis*. PhD thesis, Académie de Paris, Université Paris 6 – Pierre et Marie Curie, 2004. URL `http://recherche.ircam.fr/equipes/analyse-synthese/schwarz/thesis/`. 17, 19

D. Schwarz. Corpus-based concatenative synthesis. *Signal Processing Magazine, IEEE*, 24:92–104, 2007. 5, 47

D. Schwarz. State of the art in sound texture synthesis. In *Proceedings of the International Conference on Digital Audio Effects (DAFx2011)*, Paris, France, Sept. 2011. 3, 4

D. Schwarz, G. Beller, B. Verbrugghe, and S. Britton. Real-Time Corpus-Based concatenative synthesis with CataRT. In *Proceedings of the International Conference on Digital Audio Effects (DAFx2006)*, page 279–282, Montreal, Quebec, Canada, 2006. 24, 25

J. Serrà, E. Gómez, P. Herrera, and X. Serra. Chroma Binary Similarity and Local Alignment Applied to Cover Song Identification. *IEEE Transactions on Audio, Speech and Language Processing*, 16(6):1138–1151, 2008. 93

X. Serra. *A System for Sound Analysis/Transformation/Synthesis based on a Deterministic plus Stochastic Decomposition.* PhD thesis, Stanford University, 1989. 29, 113

B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009. 35

K. Siedenburg and M. Dörfler. Structured Sparsity for Audio Signals. In *Proceedings of the International Conference on Digital Audio Effects (DAFx2011)*, Paris, France, 2011. 31

K. Siedenburg and M. Dörfler. Persistent Time-Frequency Shrinkage for Audio Denoising. *Journal of the Audio Engineering Society*, 61(1/2), 2013. 31

K. Siedenburg, M. Kowalski, and M. Dörfler. Audio declipping with social sparsity. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1577–1581, May 2014. 31

I. Simon, S. Basu, D. Salesin, and M. Agrawala. Audio analogies: Creating new music from an existing performance by concatenative synthesis. In *Proceedings of the 2005 International Computer Music Conference*, page 65–72, 2005. 7, 18, 20

M. Sipser. *Introduction to the theory of computation.* PWS Publishing Company, 1997. ISBN 978-0-534-94728-6. 174

P. Smaragdis, C. Févotte, G. Mysore, N. Mohammadiha, and M. Hoffman. Static and Dynamic Source Separation Using Nonnegative Factorizations: A unified view. *IEEE Signal Processing Magazine*, 31(3):66–75, May 2014. 22, 31

J. O. Smith. Viewpoints on the History of Digital Synthesis. In *Proceedings of International Computer Music Conference (ICMC91)*, 1991. URL `https://ccrma.stanford.edu/~jos/kna/`. 16

J. O. Smith. Digital audio resampling home page, January 28, 2002. URL `https://ccrma.stanford.edu/~jos/resample/`. 50, 66

J. O. Smith. *Mathematics of the Discrete Fourier Transform (DFT) with Audio Applications, Second Edition.* 2007. URL `http://ccrma.stanford.edu/~jos/mdft/`. 30, 49, 50, 66, 67

A. Smola, N. Murata, B. Schölkopf, and K. R. Muller. Asymptotically optimal choice of ε-loss for support vector machines. In *Proceedings of the International Conference on Artificial Neural Networks, Perspectives in Neural Computing*, pages 105–110, September 1998. 69

A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, September 2003. 67, 68

S. S. Stevens, J. Volkmann, and E. B. Newman. A Scale for the Measurement of the Psychological Magnitude Pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190, Jan. 1937. 96

D. Stowell, S. Musevic, J. Bonada, and M. Plumbey. Improved Multiple Birdsong Tracking with Distribution Derivative Method and Markov Renewal Process Clustering. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013. 31

B. L. Sturm. Adaptive concatenative sound synthesis and its application to micromontage composition. *Computer Music Journal*, 30:46–66, Dec. 2006. 18

B. L. Sturm. *Sparse Approximation and Atomic Decomposition: Considering Atom Interactions in Evaluating and Building Signal Representations.* PhD thesis, University of California: Santa Barbara, Sept. 2009. URL `http://pqdtopen.proquest.com/doc/304857914.html?FMT=ABS`. 22, 31

B. L. Sturm and M. G. Christensen. Comparison of orthogonal matching pursuit implementations. In *Proceedings of the 20th European Signal Processing Conference, EUSIPCO 2012*, pages 220–224, Bucharest, Romania, August 2012. IEEE. 154

B. L. Sturm, C. Roads, A. McLeran, and J. J. Shynk. Analysis, visualization, and transformation of audio signals using dictionary-based methods. In *Proceedings of the International Computer Music Conference. Inter-*

national Computer Music Association, 2008a. 22, 31

B. L. Sturm, J. Shynk, L. Daudet, and C. Roads. Dark Energy in Sparse Atomic Estimations. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(3):671–676, Mar. 2008b. 152

M. J. Terrell and J. D. Reiss. Automatic Monitor Mixing for Live Musical Performance. *Journal of the Audio Engineering Society*, 57(11):927–936, Nov. 2009. 30

The Echo Nest. Echo Nest Remix, 2009. URL `https://echonest.github.io/remix/`. 177

J. C. R. Thomas. A New Clustering Algorithm Based on K-Means Using a Line Segment as Prototype. In C. S. Martin and S.-W. Kim, editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, number 7042 in Lecture Notes in Computer Science, pages 638–645. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-25084-2. 84

S. K. Tjoa and K. J. R. Liu. Factorization of overlapping harmonic sounds using approximate matching pursuit. In *Proc. Int. Soc. Music Information Retrieval Conf.*, pages 257–262, Miami, FL, Oct. 2011. 165

H. Valbret, E. Moulines, and J. Tubach. Voice transformation using PSOLA technique. In *1992 IEEE International Conference on Acoustics, Speech, and Signal Processing, 1992. ICASSP-92*, volume 1, pages 145–148 vol.1, Mar. 1992. 21

E. Van Buskirk. The Man Behind Scrambled Hackz. *WIRED*, Apr. 2006. URL `http://archive.wired.com/entertainment/music/commentary/listeningpost/2006/04/70664?currentPage=all`. 26

V. Vapnik, S. E. Golowich, and A. Smola. Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing. In *Advances in Neural Information Processing Systems 9*, pages 281–287. MIT Press, 1996. 67

V. Verfaille. *Effets audionumériques adaptatifs : théorie, mise en œuvre et usage en création musicale numérique.* PhD thesis, Université de la Méditerranée - Aix-Marseille II, Sept. 2003. 46

V. Verfaille and D. Arfib. Implementation strategies for adaptive digital audio effects. In *International Conference on Digital Audio Effects (DAFx-02)*, pages 21–26, Hamburg, Germany, 2002. 46

V. Verfaille and P. Depalle. Adaptive Effects Based on STFT, Using a

Source-Filter Model. In *Proceedings of the International Conference on Digital Audio Effects (DAFx2004)*, pages 296–301, Naples, Italy, 2004. 29, 44, 46

V. Verfaille, J. Boissinot, P. Depalle, and M. M. Wanderley. SSynth: a Real Time Additive Synthesizer with Flexible Control. In *Proceedings of the International Computer Music Conference (ICMC'06)*, New Orleans, USA, 2006a. 27

V. Verfaille, M. M. Wanderley, and P. Depalle. Mapping strategies for gestural and adaptive control of digital audio effects. *Journal of New Music Research*, 35(1):71–93, Mar. 2006b. 46

V. Verfaille, U. Zolzer, and D. Arfib. Adaptive digital audio effects (a-DAFx): a new class of sound transformations. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1817–1831, Sept. 2006c. 46

S. Vitaladevuni, P. Natarajan, R. Prasad, and P. Natarajan. Efficient Orthogonal Matching Pursuit using sparse random projections for scene and video classification. In *2011 IEEE International Conference on Computer Vision (ICCV)*, pages 2312–2319, Nov. 2011. 165

A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, Apr. 1967. ISSN 0018-9448. 17

D. Wessel, C. Drame, and M. Wright. Removing the Time Axis from Spectral Model Analysis-Based Additive Synthesis: Neural Networks versus Memory-Based Machine Learning. In *Proceedings of the International Computer Music Conference (ICMC98)*, pages 62–65, Ann Arbor, Michigan, 1998. 5, 26

M. Yee-King and M. Roth. Synthbot: An unsupervised software synthesizer programmer. In *Proceedings of the International Computer Music Conference (ICMC08)*, pages 184–187, Belfast, Northern Ireland 2008. 28, 64

A. Zils and F. Pachet. Musical mosaicing. In *Proceedings of the International Conference on Digital Audio Effects (DAFx2001)*, Limerick, Ireland, 2001. 17, 19, 90

E. Zwicker. Subdivision of the Audible Frequency Range into Critical Bands (Frequenzgruppen). *The Journal of the Acoustical Society of America*, 33 (2):248–248, Feb. 1961. 96