ABSTRACT

Title of dissertation: SPARSE AND NONNEGATIVE FACTORIZATIONS FOR MUSIC UNDERSTANDING

Steven Kiemyang Tjoa, Doctor of Philosophy, 2011

Dissertation directed by: Professor K. J. Ray Liu
Department of Electrical and Computer Engineering

In this dissertation, we propose methods for sparse and nonnegative factorization that are specifically suited for analyzing musical signals. First, we discuss two constraints that aid factorization of musical signals: harmonic and co-occurrence constraints. We propose a novel dictionary learning method that imposes *harmonic constraints* upon the atoms of the learned dictionary while allowing the dictionary size to grow appropriately during the learning procedure. When there is significant spectral-temporal overlap among the musical sources, our method outperforms popular existing matrix factorization methods as measured by the recall and precision of learned dictionary atoms. We also propose *co-occurrence constraints* – three simple and convenient multiplicative update rules for nonnegative matrix factorization (NMF) that enforce dependence among atoms. Using examples in music transcription, we demonstrate the ability of these updates to represent each musical note with multiple atoms and cluster the atoms for source separation purposes.

Second, we study how spectral and temporal information extracted by nonnegative factorizations can improve upon musical instrument recognition. Musical instrument recognition in melodic signals is difficult, especially for classification systems that rely entirely upon spectral information instead of temporal information. Here, we propose a simple and effective method of combining spectral and temporal information for instrument recognition. While existing classification methods use traditional features such as statistical moments, we extract novel features from spectral and temporal atoms generated by NMF using a biologically motivated *multiresolution gamma filterbank*. Unlike other methods that require thresholds, safeguards, and hierarchies, the proposed spectral-temporal method requires only simple filtering and a flat classifier.

Finally, we study how to perform sparse factorization when a large dictionary of musical atoms is already known. Sparse coding methods such as matching pursuit (MP) have been applied to problems in music information retrieval such as transcription and source separation with moderate success. However, when the set of dictionary atoms is large, identification of the best match in the dictionary with the residual is slow – linear in the size of the dictionary. Here, we propose a variant called *approximate matching pursuit* (AMP) that is faster than MP while maintaining scalability and accuracy. Unlike MP, AMP uses an approximate nearest-neighbor (ANN) algorithm to find the closest match in a dictionary in sublinear time. One such ANN algorithm, locality-sensitive hashing (LSH), is a probabilistic hash algorithm that places similar, yet not identical, observations into the same bin. While the accuracy of AMP is comparable to similar MP methods, the computational complexity is reduced. Also, by using LSH, this method scales easily; the dictionary can be expanded without reorganizing any data structures.

SPARSE AND NONNEGATIVE FACTORIZATIONS FOR MUSIC UNDERSTANDING

by

Steven Kiemyang Tjoa

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2011

Advisory Committee:
Professor K. J. Ray Liu, Chair/Advisor
Professor Min Wu
Professor Rama Chellappa
Professor Jonathan Z. Simon
Professor Lawrence C. Washington

Acknowledgments

First, I thank my advisor, Professor Ray Liu, for guiding me throughout this wonderful and worthwhile journey. In January of 2003, he accepted an untested, unproven undergraduate student with nothing but "potential" into his research group to begin the first of many research projects. Someone with his reputation and busy schedule did not need another student like me. But Dr. Liu is not just another important professor. He really cares about the maturation and well being of others like myself. For that, I am forever grateful.

Next, I thank other members in my dissertation committee – Professors Min Wu, Rama Chellappa, Jonathan Simon, and Lawrence Washington – for taking the time to attend my defense and read this dissertation, as well as Professor Shihab Shamma who served on my proposal committee but was unable to attend the dissertation defense. Outside of this dissertation, these professors have influenced me throughout my years at Maryland as teachers, advisors, and friends. Likewise, I also thank other faculty members at Maryland including Professors André Tits, Adrian Papamarcou, Steven Marcus, Donald Yeung, and Richard La.

I thank each and every member of the Signals and Information Group, especially Wan-Yi Lin, Hong Vicky Zhao, Yan Chen, Matt Stamm, Charles Pandana, and Quoc Lai. I met Quoc through Dr. Liu in 2003 when we were both undergraduates at UMD, and we both decided to follow the same path from then until now. We have remained great friends ever since. I also thank other graduate students including Avinash Varna, Daniel Garcia-Romero, and Vishal Patel for their fruitful discussions related to audio processing, sparse factorizations, and information retrieval, not to mention their effort as my teammates (or opponents) in foosball, frisbee, or tennis.

Most of all, I thank my family – Bing Tjoa, May Tjoa, Debra Tjoa, and Vivian Samson – for always being there for me. Engineering is always a team effort, and every team succeeds or fails together. On this day, each of us is earning a doctoral degree from the University of Maryland.

This dissertation is not only dedicated to them. It *belongs* to them.

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

AMP	Approximate Matching Pursuit
ANN	Approximate Nearest Neighbor
BP	Basis Pursuit
CQT	Constant-Q Transform
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DTW	Dynamic Time Warping
FFT	Fast Fourier Transform
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
IS	Itakura-Saito (Divergence)
KL	Kullback-Leibler (Divergence)
KSVD	K-Singular Value Decomposition
NN-KSVD	Nonnegative KSVD
LPC	Linear Predictive Coding
LSH	Locality Sensitive Hashing
MFCC	Mel-Frequency Cepstral Coefficient
MGFR	Multiresolution Gamma Filterbank Response
MIDI	Musical Instrument Digital Interface
MIR	Music Information Retrieval
MP	Matching Pursuit
NMF	Nonnegative Matrix Factorization
OMP	Orthogonal Matching Pursuit
PCA	Principal Component Analysis
QBH	Query By Humming
STFT	Short-Time Fourier Transform
STOMP	Stagewise Orthogonal Matching Pursuit
STRF	Spectrotemporal Receptive Field
SVD	Singular Value Decomposition
SVM	Support Vector Machine

List of Symbols

M	number of rows (i.e., frequency bins) in spectrogram $\mathbf{X}$
N	number of columns (i.e., frames of $x(t)$) in spectrogram $\mathbf{X}$
K	number of atoms in the dictionary
k	number of hash functions in one locality sensitive hash
L	number of hash tables in locality sensitive hashing
$x(t)$	input signal
$X(f)$	Fourier transform of $x(t)$
$u(t)$	unit step function
$\mathbf{a}_k$	k^{th} column of $\mathbf{A}$; spectral atom k in the dictionary
$\mathbf{c}$	constant-Q transform vector
$\mathbf{s}_k$	$[k^{\text{th}}$ row of $\mathbf{S}]^T$; temporal atom k in the dictionary
$\mathbf{s}[n]$	n^{th} column vector of $\mathbf{S}$
$\mathbf{x}[n]$	n^{th} column vector of $\mathbf{X}$
$\mathbf{A}$	spectral dictionary
$\mathbf{E}$	error matrix
$\mathbf{Q}$	co-occurrence constraint matrix
$\mathbf{S}$	temporal dictionary
$\mathbf{X}$	spectrogram of $x(t)$
$\mathbf{X} \cdot \mathbf{Y}$	element-wise multiplication between $\mathbf{X}$ and $\mathbf{Y}$
$\mathbf{X}/\mathbf{Y}$	element-wise division between $\mathbf{X}$ and $\mathbf{Y}$
$d(\cdot, \cdot)$	distance metric
$[\cdot]_+$	$\max(\cdot, 0)$

Chapter 1

Introduction

For centuries, scientists and mathematicians have experienced the need to *factorize data*. Factorization – the decomposition of data into simple, fundamental factors – allows humans to identify the most meaningful components of data. Today, this need is greater than ever. Factorization has become popular in such diverse fields as in engineering, biology, physics, and statistics. As the complexity of modern systems increases, the need for insightful data decomposition becomes greater than ever. Subsequently, engineers have developed and applied more sophisticated methods of factorization toward real-world data in a variety of research areas.

One such research area is *music understanding*, or music information retrieval (MIR). The goal of MIR algorithms is to extract high-level semantic information from low-level musical data. There are a variety of open problems in MIR: search, classification, transcription, source separation, recommendation, rhythm detection, segmentation, and more. We focus on two problems in particular: transcription and source separation. Music transcription is the process of obtaining discrete musical events such as notes and beats from an acoustic waveform. For example, WAV-to-MIDI conversion is a music transcription problem. Source separation is the act of separating a polyphonic signal into its individual sources; in musical signals, source separation usually aims to decompose a signal into indivdual musical instruments or voices.

Among the many approaches for performing tasks such as music transcription and source separation, one category of approaches – *sparse and nonnegative matrix factorization* – has received plenty of attention due to their elegance and effectiveness. By first expressing a representation of the musical signal as a matrix, these methods decompose the matrix into a sum of individual dictionary atoms, each corresponding to one musical source or note. Within the past five years, sparse and nonnegative factorizations have completely changed the way we perform musical signal analysis.

These methods commonly share two important steps: *dictionary learning* and *sparse coding*. Dictionary learning refers to the construction of a set of atoms – the dictionary – from which the input signal can be represented, and sparse coding is used to compute the contribution of each dictionary atom to the signal at each moment in time. Methods known as nonnegative matrix factorization (NMF) also impose a nonnegativity constraint on the dictionary and its coefficients in order to learn more meaningful atoms. The nonnegativity constraint makes sense considering that we only have the presence or absence of a source from a signal and never the "subtraction" of a source from a signal in which it is already absent. Given a nonnegative matrix $\mathbf{X}$, the objective of NMF is to find two nonnegative matrices, $\mathbf{A}$ and $\mathbf{S}$, that minimizes some divergence between $\mathbf{X}$ and $\mathbf{AS}$. The NMF problem was originally popularized by Paatero and Tapper [4], and Lee and Seung later proposed algorithms for solving the NMF problem using multiplicative update rules [5, 6].

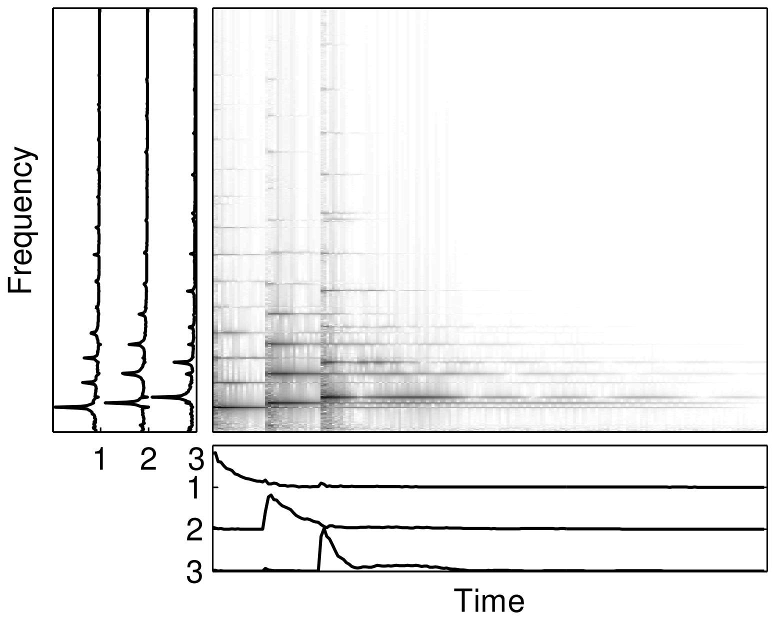


Figure 1.1: Nonnegative matrix factorization of the spectrogram **X** (top right) into **A** (top left) and **S** (bottom right) for three piano notes.

Fig. 1.1 illustrates the use of NMF when applied to a musical audio signal. The matrix **X** represents the magnitude spectrogram of an audio signal containing three notes played by a piano. After decomposition into a rank-three approximation using NMF, the three columns of **A** – also referred to as *dictionary atoms* – represent the frequency spectra of the three piano notes, and the three rows of **S** represent their corresponding temporal activities. Note how the columns of **A** accurately represent the spectra of the three piano notes. To perform source separation, we reconstruct an estimate of the spectrogram for an individual source using only a subset of the learned dictionary atoms.

1.1 Motivation

Unfortunately, the basic sparse and nonnegative factorization methods are limited and underutilized. When there is significant spectral-temporal overlap in the signal among the dictionary atoms, it becomes difficult for these methods to learn atoms properly. Often, information from multiple atoms is represented as a single atom by the learning procedure. If an atom in the output dictionary contains musical information from multiple sources, transcription and source separation cannot be accurately performed. Furthermore, if the dictionary atoms themselves are highly correlated, as is common when harmonic frequencies between atoms overlap, accurate dictionary learning becomes even more difficult.

Objects may require more than a single dictionary atom in order to be approximated accurately. For example, Fig. 2.6 illustrates the decomposition of a spectrogram of an audio signal containing one note played by a violin. Although only one note is played, the vibrato induced by the performer causes the pitch to modulate. As a result, a rank-one approximation is not sufficient to capture this pitch modulation. A user can select multiple dictionary atoms to represent one note. However, in the presence of many other sources, it is unclear which atoms to select. In other words, after learning is complete, there is no straightforward way to cluster multiple dictionary atoms that belong to the same source.

One alternative to dictionary learning is to use a large, predefined, overcomplete dictionary where each atom is already labeled and assumed to contain information from only one musical source. Instead of learning an optimal dictionary for a given musical signal, it may suffice to match the signal to this large set of precomputed, labeled dictionary atoms. Then, by decomposing a signal with respect to this fixed dictionary, classification is easily achieved by simply reading the label of the atom. Of course, the performance of such an algorithm depends upon the breadth of the dictionary. When atoms from more musical sources are added to the dictionary, the dictionary's ability to decompose polyphonic music will improve. However, dictionary growth introduces concerns related to scalability and computational complexity. While the aforementioned algorithms have significantly advanced the state of the art, they remain slow and difficult to scale as the dictionary size increases. Most of the original sparse coding methods such as matching pursuit (MP) [1] and NMF with multiplicative updates [5, 6] have complexity that is linear in the size of the dictionary. As a result, when dictionary sizes grow, the transcription capability of these algorithms diminishes.

Sparse and nonnegative factorization also has uses beyond transcription; for example, it can also be used in musical instrument recognition. Further progress in musical instrument recognition may depend upon recent advances in signal processing such as sparse coding and dictionary learning. Some methods already use NMF to exploit the spectral redundancy in a signal. However, redundancy remains in the *temporal* domain. Classification methods that only utilize spectral information are discarding the potentially useful temporal information that could be used to improve classification performance. By extracting the spectral and temporal information from NMF in a principled manner, we may be able to improve upon the state of the art in musical instrument recognition.

1.2 Dissertation Outline

The remainder of the dissertation is organized as follows.

1.2.1 Constrained Dictionary Learning

First, we propose two novel methods for learning dictionaries of musical atoms. The first method uses *harmonic constraints* to learn meaningful dictionary atoms despite spectral-temporal overlap among the musical sources. The dictionaries learned by this method contain atoms which accurately resemble the original notes and sources which comprise the input signal. While our method is based on matrix factorization, it imposes an additional harmonic constraint that restricts each atom to represent at most one pitch. Furthermore, our method is flexible by allowing the size of the dictionary to grow based upon the complexity of the input signal, unlike other methods which fix the dictionary size a priori. Our method consistently outperforms other dictionary learning methods such as nonnegative matrix factorization with multiplicative updates (NMF-MU) [6], K-SVD [7], nonnegative K-SVD (NN-K-SVD) [8], and the method of optimal directions (MOD) [9], as measured by the recall and precision of learned dictionary atoms.

The second method enforces dependence among sets of dictionary atoms by introducing *co-occurrence constraints* – constraints that specify which dictionary atoms are dependent, or co-occur. We introduce three new update rules to enforce dependence among dictionary atoms by incorporating co-occurrence constraints into NMF. These rules are conceptually simple, easy to implement, and effective for describing sources using multiple dictionary atoms. We formulate the NMF problem with co-occurrence constraints, we derive new update rules for minimizing three common divergence metrics, and we illustrate the use of these update rules in the context of music transcription.

1.2.2 Spectral-Temporal Musical Instrument Recognition

Next, we use NMF to extract novel features from the decomposed atoms to perform musical instrument recognition. We propose the use of a new temporal feature, the *multiresolution gamma filterbank response* (MGFR), which is computed from the temporal atoms extracted from spectrograms using NMF. By combining the beneficial elements of NMF, multiresolution analysis, and supervised classification, this algorithm is rigorous and accurate yet without too many "moving parts." Other methods may require preprocessing such as note segmentation [10], pitch estimation [11], or classifiers such as hierarchical clustering [12, 13]. To our knowledge, no other work has attempted to use temporal information extracted from NMF to classify instruments in a systematic manner. Because NMF has become widely popular for facilitating audio classification, this work is useful to practitioners in the field. We also analyze the spectral and temporal properties of the multiresolution gamma filterbank, and we provide comprehensive experiments on isolated notes and melodic phrases from a diverse set of instruments. We show that our spectral-temporal method is competitive with the state of the art.

1.2.3 Approximate Matching Pursuit

Finally, we discuss the problem of sparse and nonnegative factorization using a fixed dictionary. We propose a variant of MP called *approximate matching pursuit* (AMP). Unlike MP and NMF, AMP can decompose signals into a sparse combination of atoms with complexity that is *sublinear* in the dictionary size while maintaining accuracy.

To do this, AMP uses an approximate nearest neighbor (ANN) method to find approximate matches to the signal residual at each iteration. The ANN method that we choose in this work is locality sensitive hashing (LSH), a probabilistic hash algorithm that places similar, yet not identical, observations into the same bin. LSH can retrieve near neighbors with a complexity that is sublinear in the dictionary size. Not only is LSH fast, but it is also scalable – as the dictionary grows, reorganizations of the data structure are unnecessary. We simply add the new dictionary atom into its respective bin in the hash table.

Our experiments demonstrate that AMP is as accurate and robust as MP variants such as OMP [2] and STOMP [3] under a wide variety of scenarios related to sparsity, dimensionality, and additive noise. At the same time, AMP requires less computation than OMP and STOMP; at convergence, AMP computes fewer inner products than the other algorithms.

Chapter 2

Constrained Dictionary Learning

When performing factorization for music information retrieval, we expect the learned dictionary of musical atoms to be formed of interpretable elements, each exhibiting musical properties such as pitch and timbre. The sparsity and nonnegativity constraints mentioned earlier both provide an important step toward learning meaningful dictionaries. Nevertheless, the results achieved by the community leave room for improvement. In this chapter, I propose two additional constraints – harmonic and co-occurrence constraints – that improve upon the basic factorization formulations.

2.1 Harmonic Constraints

Recently, researchers have proposed many approaches for performing music transcription and source separation. In particular, one such category of approaches – spectral decomposition through matrix factorization – has received plenty of attention. By first expressing a time-frequency representation of the musical signal as a matrix, these methods decompose each column of the matrix into a summation of individual vectors, each corresponding to one musical source or note [14, 15].

These methods commonly share two important steps: *dictionary learning* and *sparse coding*. Dictionary learning refers to the construction of a set of atoms –

the dictionary – from which the input signal can be represented, and sparse coding is used to compute the contribution of each dictionary atom to the signal at each moment in time. Methods known as nonnegative matrix factorization (NMF) also impose a nonnegativity constraint on the dictionary and its coefficients in order to learn more meaningful atoms. The nonnegativity constraint makes sense considering that we only have the presence or absence of a source from a signal and never the "subtraction" of a source from a signal in which it is already absent.

Unfortunately, these methods also share a common limitation. When there is significant spectral-temporal overlap in the signal among the dictionary atoms, it becomes difficult for these methods to learn atoms properly. Often, information from multiple atoms is represented as a single atom by the learning procedure. If an atom in the output dictionary contains musical information from multiple sources, transcription and source separation cannot be accurately performed. Furthermore, if the dictionary atoms themselves are highly correlated, as is common when harmonic frequencies between atoms overlap, accurate dictionary learning becomes even more difficult.

In this section, we propose a novel dictionary learning method designed to perform well despite spectral-temporal overlap among the dictionary atoms. The dictionaries learned by this method contain atoms which accurately resemble the original notes and sources which comprise the input signal. While our method is based on matrix factorization, it imposes an additional harmonic constraint that restricts each atom to represent at most one pitch. Furthermore, our method is flexible by allowing the size of the dictionary to grow based upon the complexity of the input signal, unlike other methods which fix the dictionary size a priori. Our method consistently outperforms other dictionary learning methods such as nonnegative matrix factorization with multiplicative updates (NMF-MU) [6], K-SVD [7], nonnegative K-SVD (NN-K-SVD) [8], and the method of optimal directions (MOD) [9], as measured by the recall and precision of learned dictionary atoms.

2.1.1 Problem Formulation

Dictionary learning methods based upon matrix factorization accept a time-frequency representation of the musical signal as the input. Although there exist many different time-frequency representations, we will simply use the magnitude spectrogram of the input signal.

Given a discrete-time single-channel music signal $x(n)$, the magnitude spectrogram of the input signal is a real-valued nonnegative matrix $\mathbf{X} \in \mathbb{R}_+^{M \times N}$, where $\mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2 \ \ldots \ \mathbf{x}_N]$, whose N columns are the discrete Fourier transform (DFT) magnitudes of consecutive, possibly overlapping, frames of the input signal. Given the matrix $\mathbf{X}$, our primary goal is to find two matrices, the dictionary $\mathbf{A} \in \mathbb{R}_+^{M \times K}$ and gain matrix $\mathbf{S} \in \mathbb{R}_+^{K \times N}$, which minimize some distance between $\mathbf{X}$ and $\mathbf{AS}$. If we denote $||\mathbf{X}||_F$ as the Frobenius norm of $\mathbf{X}$, where $||\mathbf{X}||_F^2 = \mathrm{tr}(\mathbf{X}^T\mathbf{X}) = \sum_{i,j} x_{ij}^2$, then we can describe the problem as follows:

$$\min_{\mathbf{A},\mathbf{S}} ||\mathbf{X} - \mathbf{AS}||_F^2 \qquad \text{such that} \quad \mathbf{A} \in \mathbb{R}_+^{M \times K}, \mathbf{S} \in \mathbb{R}_+^{K \times N}. \tag{2.1}$$

The columns of the matrix $\mathbf{A} = [\mathbf{a}_1 \ \mathbf{a}_2 \ \ldots \ \mathbf{a}_K]$ correspond to the individual atoms of the dictionary. In this musical context, these atoms resemble the spectra of individual sources or notes found in the musical mixture. The gain matrix $\mathbf{S} = [\mathbf{s}_1\ \mathbf{s}_2\ ...\ \mathbf{s}_K]^T$ represents the contribution of each dictionary atom in the spectrogram $\mathbf{X}$, i.e., the element s_{kn} indicates the amount of atom $\mathbf{a}_k$ present in observation $\mathbf{x}_n$. We refer to the row vector $\mathbf{s}_k^T$ as the k^{th} row of $\mathbf{S}$, i.e., $\mathbf{s}_k$ indicates the activity of atom $\mathbf{a}_k$ across time.

2.1.2 Dictionary Learning: Existing Methods

To motivate our proposed algorithm, we discuss existing dictionary learning procedures based upon singular value decomposition (SVD), including K-SVD and its nonnegative variant, NN-K-SVD [7, 8]. SVD computes the matrix factorization $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ where $\mathbf{U} = [\mathbf{u}_1\ \mathbf{u}_2\ ...\ \mathbf{u}_M]$ and $\mathbf{V} = [\mathbf{v}_1\ \mathbf{v}_2\ ...\ \mathbf{v}_N]$ are both orthnormal matrices and the diagonal matrix $\mathbf{\Sigma}$ is such that for any choice of K, the difference $||\mathbf{X} - \sum_{k=1}^{K} \sigma_{kk}\mathbf{u}_k\mathbf{v}_k^T||_F$ is minimized. In our context, the dictionary $\mathbf{A}$ corresponds to the first K columns of $\mathbf{U}$, and the gain matrix $\mathbf{S}$ corresponds to the first K rows of $\mathbf{\Sigma}\mathbf{V}^T$. Intuitively, through SVD, we find the K dictionary atoms and their associated gains which best represent the magnitude spectrogram $\mathbf{X}$.

However, SVD does not guarantee sparsity or nonnegativity of the factorization. On the other hand, K-SVD is an iterative algorithm that learns a dictionary that can be overcomplete and whose gain coefficients are sparse. Instead of immediately solving for $\mathbf{A}$ and $\mathbf{S}$ jointly, this algorithm solves the minimization in (2.1) one dictionary atom at a time, ignoring the nonnegativity constraints, while the other atoms remain constant. In other words, for a given k, each iteration of K-SVD solves the minimization

$$\min_{\mathbf{a}_k,\mathbf{s}_k} ||\mathbf{X} - \mathbf{A}\mathbf{S}||_F^2 \,. \tag{2.2}$$

Note that

$$\begin{aligned} ||\mathbf{X} - \mathbf{A}\mathbf{S}||_F &= \left|\left|\mathbf{X} - \sum_{j=1}^{K} \mathbf{a}_j \mathbf{s}_j^T\right|\right|_F \\ &= \left|\left|\left(\mathbf{X} - \sum_{j\neq k} \mathbf{a}_j \mathbf{s}_j^T\right) - \mathbf{a}_k \mathbf{s}_k^T\right|\right|_F \,. \end{aligned} \tag{2.3}$$

For convenience, denote

$$\mathbf{E}_k = \mathbf{X} - \sum_{j\neq k} \mathbf{a}_j \mathbf{s}_j^T. \tag{2.4}$$

Then, the solution to (2.2) is the rank-one approximation of the SVD $\mathbf{E}_k = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, specifically, $\mathbf{a}_k = \mathbf{u}_1$ and $\mathbf{s}_k = \sigma_{11}\mathbf{v}_1$. K-SVD adjusts $\mathbf{a}_k$ and $\mathbf{s}_k$ accordingly in each iteration and moves on to the next dictionary atom in the next iteration. The entire process is repeated until convergence of the dictionary occurs. Sparse coding is applied to update the gain matrix before each set of K iterations.

While K-SVD encourages sparsity and accommodates overcompleteness, it still does not influence the nonnegativity of either the dictionary $\mathbf{A}$ or the gain matrix $\mathbf{S}$. On the other hand, nonnegative K-SVD (NN-K-SVD) retains the same flavor of K-SVD while maintaining nonnegativity of the matrix elements. Consider the following constrained minimization:

$$\min_{\mathbf{a}_k} ||\mathbf{E}_k - \mathbf{a}_k \mathbf{s}_k^T||_F^2 \qquad \text{such that } \; \mathbf{a}_k \in \mathbb{R}_+^M. \tag{2.5}$$

Here, we keep $\mathbf{s}_k$ constant and enforce the nonnegativity of $\mathbf{a}_k$. By differentating the objective function, it can be shown that the optimal solution for $\mathbf{a}_k$ (similarly, for $\mathbf{s}_k$ by keeping $\mathbf{a}_k$ constant) is

$$\mathbf{a}_k = \left[\frac{\mathbf{E}_k\mathbf{s}_k}{\mathbf{s}_k^T\mathbf{s}_k}\right]_+ \qquad \mathbf{s}_k = \left[\frac{\mathbf{E}_k^T\mathbf{a}_k}{\mathbf{a}_k^T\mathbf{a}_k}\right]_+ , \tag{2.6}$$

where $[\cdot]_+$ denotes a matrix or vector whose negative elements are set to zero. By observing that the Hessian of the objective function with respect to $\mathbf{a}_k$ is proportional to the identity matrix, the optimal projection from the unconstrained minimum to the constrained minimum is performed simply by setting all negative elements of the unconstrained solution to zero, hence the solution in (2.6). Each iteration of NN-K-SVD uses these rules to update $\mathbf{a}_k$ and $\mathbf{s}_k$. While we no longer minimize $\mathbf{a}_k$ and $\mathbf{s}_k$ jointly, the updates still guarantee a decrease in the objective function while maintaining nonnegativity of the matrices $\mathbf{A}$ and $\mathbf{S}$.

2.1.3 Proposed Algorithm

While NN-K-SVD can find numerically acceptable solutions to (2.1), some problems remain. First, there is no guarantee that the individual atoms of the learned dictionary will each correspond to only one musical source. In particular, when multiple atoms coincide in time (e.g., $\mathbf{s}_1$ and $\mathbf{s}_2$ are highly correlated), the aforementioned algorithms will learn a single atom that contains information from both $\mathbf{a}_1$ and $\mathbf{a}_2$. For example, consider the learned atoms in Fig. 2.1. We fabricate a dictionary $\mathbf{A}$ with two atoms whose gain coefficients $\mathbf{S}$ have significant overlap in time, and then construct the spectrogram $\mathbf{X} = \mathbf{AS}$. The dictionaries learned by K-SVD, NN-K-SVD, and NMF with multiplicative updates yield output dictionaries which do not match the input dictionary. However, the dictionary learned by our proposed method, discussed below, does accurately resemble the original dictionary.

The second problem deals with the size of the dictionary, K. For the popular existing algorithms, the dictionary size K must be chosen before the algorithm begins. If the chosen value of K is too low, then the learned dictionary cannot accurately represent the input spectrogram. If K is too high, computation and memory requirements can increase dramatically and unnecessarily. When executing eigendecompositions and/or matrix multiplications, these requirements can become overwhelming.

In order to solve these problems, we propose a novel approach to dictionary learning that emphasizes the presence of at most *one pitch per dictionary atom*. Our method builds upon the technical foundations of NN-K-SVD mentioned earlier. As illustrated in Fig. 2.1, existing dictionary learning algorithms are intended for general purposes, i.e., they do not enforce any perceptual constraints on the learned dictionary atoms. Under the assumption that individual musical sources do not overlap in time or frequency, existing algorithms can learn dictionaries accurately. However, this assumption is not necessarily true for musical contexts where individual sources are highly correlated.

Motivated by the observation that music contains a series of pitched and unpitched sounds, we enforce a harmonic constraint on the learned atom by filtering the spectrum represented by $\mathbf{a}_k$ through a comb filter, thus preserving the spectral

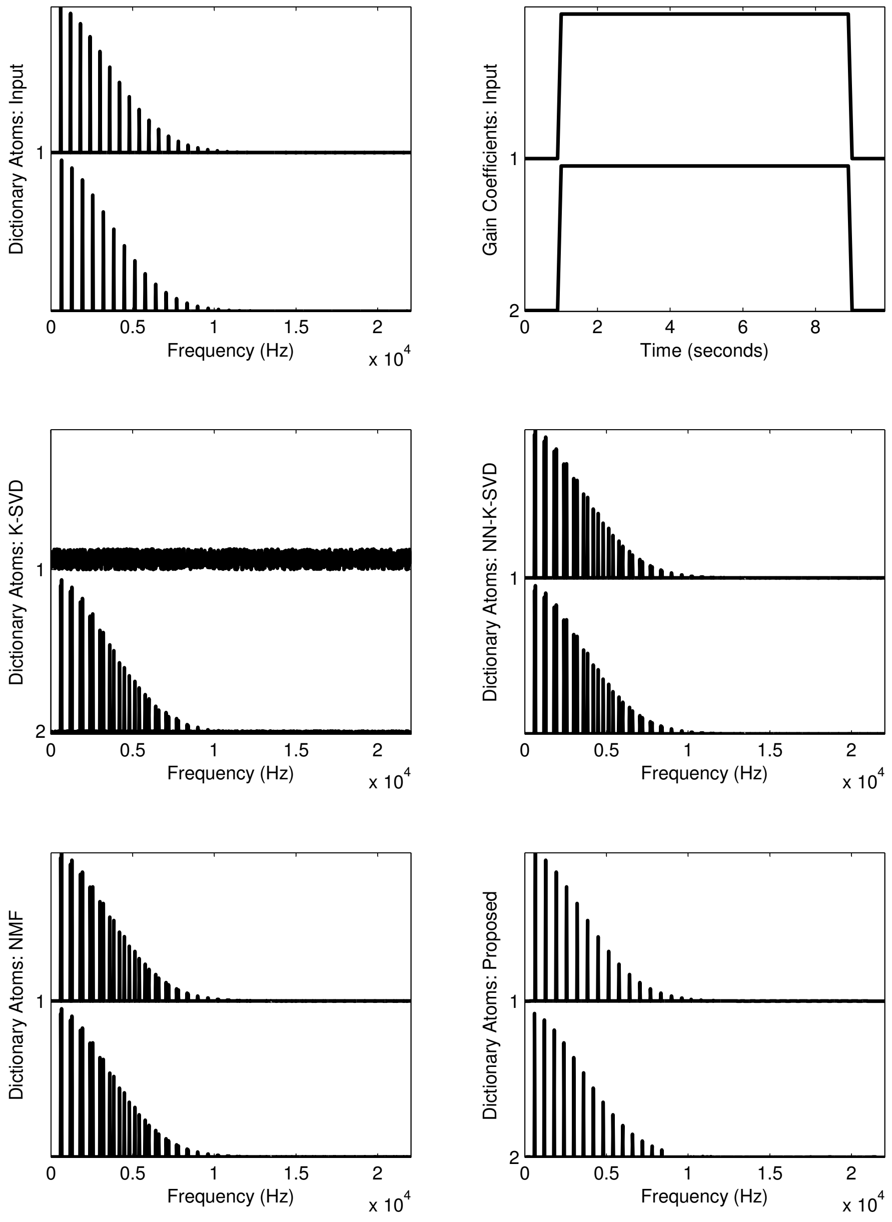


Figure 2.1: Two dictionary atoms (top left) and their gain coefficients (top right) were used to construct a spectrogram. Using either K-SVD (middle left), NN-K-SVD (middle right), or NMF-MU (bottom left), the learned dictionary atoms do not resemble the original atoms. Using the proposed algorithm (bottom right), the original and learned atoms match.

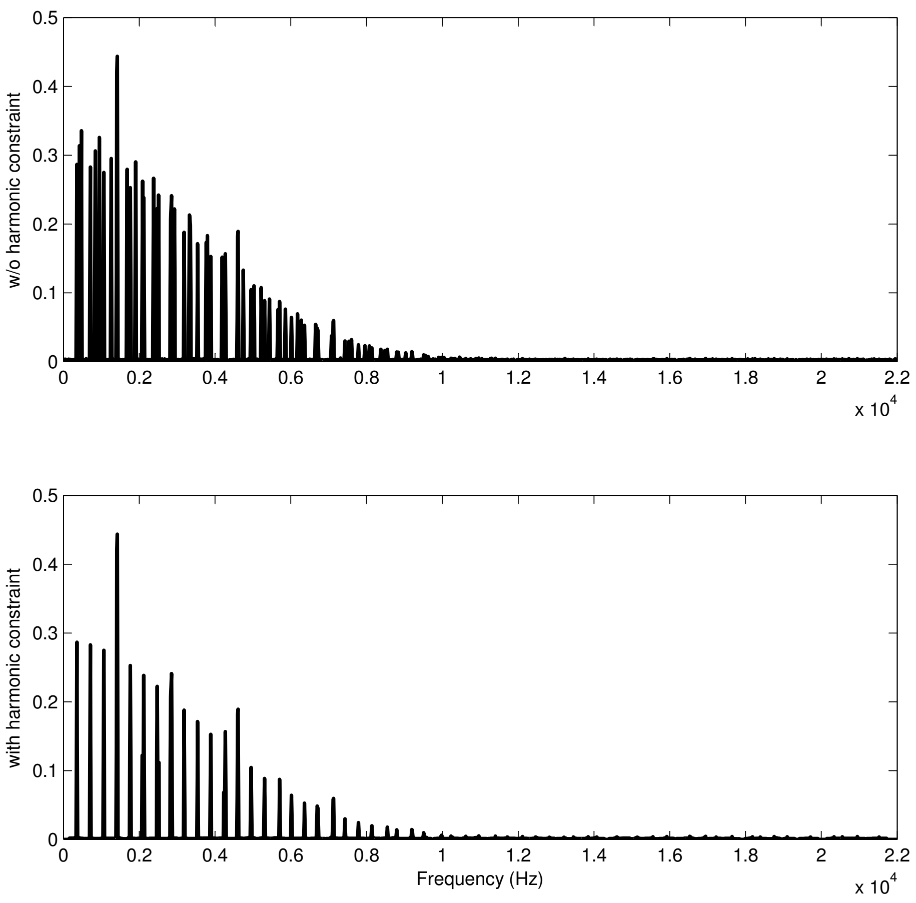


Figure 2.2: Dictionary atom of original spectrum (top) and atom after filtering spectrum through a comb filter (bottom).

energy around the harmonic frequencies and eliminating the energy at other frequencies as shown in Fig. 2.2. To estimate the fundamental frequency, we simply compute the harmonic product spectrum [16] from the first five harmonics for candidate pitches. Other frequency-domain pitch estimation algorithms can work, as well.

The other notable feature of our algorithm is the initialization and growth of the dictionary. For the best $\mathbf{A} \in \mathbb{R}^{M \times K}$, if $||\mathbf{X} - \mathbf{AS}||_F$ is still not low enough, we increment K and add another column vector $\mathbf{a}_K$ to $\mathbf{A}$ and another row vector $\mathbf{s}_K^T$ to $\mathbf{S}$. There are many reasonable ways to initialize $\mathbf{a}_K$. One could randomly generate $\mathbf{a}_K$, or $\mathbf{a}_K$ could equal the mean of the columns of $\mathbf{X} - \mathbf{AS}$. For this work, we simply set $\mathbf{a}_K$ to equal a column of $\mathbf{E}$, $\mathbf{e}_n$, where n is chosen such that $\mathbf{e}_n$ has high energy. Then, we initialize $\mathbf{s}_K = \left[\frac{\mathbf{E}_K^T \mathbf{a}_K}{\mathbf{a}_K^T \mathbf{a}_K}\right]_+$ as shown in (2.6).

With each of the basic building blocks described, we now summarize the proposed algorithm.

1. Set the dictionary size K to equal 1.

2. Initialize $\mathbf{a}_K$ and $\mathbf{s}_K$ as desired.

3. For each $k \in \{K, K-1, ..., 2, 1\}$,

   (a) Compute $\mathbf{E}_k$:

   $$\mathbf{E}_k = \mathbf{X} - \sum_{j \neq k} \mathbf{a}_j \mathbf{s}_j^T.$$

   (b) Find $\mathbf{a}_k$:

   $$\mathbf{a}_k = \left[\frac{\mathbf{E}_k \mathbf{s}_k}{\mathbf{s}_k^T \mathbf{s}_k}\right]_+.$$

   (c) Estimate the fundamental frequency, f_0, for the spectrum $\mathbf{a}_k$ using the harmonic product spectrum.

   (d) Filter $\mathbf{a}_k$ through a comb filter tuned to f_0 to emphasize its harmonicity.

   (e) Find $\mathbf{s}_k$:

   $$\mathbf{s}_k = \left[\frac{\mathbf{E}_k^T \mathbf{a}_k}{\mathbf{a}_k^T \mathbf{a}_k}\right]_+.$$

Repeat step 3 until the dictionary $\mathbf{A}$ converges.

4. If $||\mathbf{X} - \mathbf{AS}||_F^2$ is low enough, stop. Otherwise, increment K, and go to step 2.

2.1.4 Experiments

For our experiments, we synthesize a dictionary $\mathbf{A}_{in}$ of harmonic atoms similar to the atoms in Fig. 2.1 having a fixed envelope on the order of $\exp(-m^2)$, where $m \in \{1, 2, ..., M\}$ is the frequency bin index, and $M = 2048$ corresponds to the the Nyquist frequency. We also synthesize the corresponding gain coefficients to be a $K \times N$ matrix with $N = 100$ and with L ones randomly placed in each column and zero otherwise. These two matrices are multiplied to obtain $\mathbf{X}$, the input to each dictionary learning algorithm. Six dictionary learning algorithms are tested: the proposed algorithm, NMF-MU [6], K-SVD and NN-K-SVD [17], the method of optimal directions [9, 17], and basic SVD.

The output dictionary $\mathbf{A}_{out}$ from each algorithm is compared against the input dictionary in terms of hits, misses, and false alarms. A hit occurs if both dictionaries contain corresponding atoms whose normalized correlation exceeds 0.9. A miss occurs if an atom from $\mathbf{A}_{in}$ does not correlate with any atom in $\mathbf{A}_{out}$, and a false alarm occurs if an atom from $\mathbf{A}_{out}$ does not correlate with any atom in $\mathbf{A}_{in}$. Two measures are used to measure performance: recall and precision. Recall is equal to hits/(hits + misses), and precision is equal to hits/(hits + false alarms). These measures are averaged over ten trials of each experiment.

First, we illustrate the effects of the dictionary size K and the number of simultaneously active atoms L on dictionary learning. For each trial, we generate a dictionary with K harmonic atoms, each with a randomly-selected fundamental frequency that is uniformly distributed over the MIDI interval $[48, 84]$. Fig. 2.3 illustrates results for $K = 5$ and $L \in \{1, 2, 3, 4\}$, while Fig. 2.4 illustrates results for $K = 20$ and $L \in \{1, 2, ..., 19\}$. Because the existing algorithms are initialized to strictly contain K atoms, each miss must accompany a false alarm, thus making their recall and precision is equal. On the other hand, the proposed algorithm must infer the proper value for K as described earlier. When the estimated and true values of K differ, then misses and false alarms can occur independently.

As shown in Figs. 2.3 and 2.4, the recall and precision for the proposed algorithm is better than the other algorithms for most combinations of K and L, particularly when L is high. The performance of all methods degrades as L increases because the amount of spectral-temporal overlap also increases. However, the proposed method learns more accurate atoms when L is high because of the additional harmonic constraints. The two existing methods with nonnegative constraints, NMF-MU and NN-K-SVD, both perform well except when L is high because of their inability to resolve the spectral-temporal overlap. The remaining methods – K-SVD, MOD, and SVD – all fail because of the lack of a nonnegativity constraint.

Next, we show results when the pitches of the input dictionary atoms have overlapping harmonics. To ensure a high amount of overlap, we fix $K = 5$ and $L = 3$. The five chosen pitches are 200, 300, 400, 500, and 600 Hz. The gain matrix is once again randomly generated by assigning L ones to each column of $\mathbf{S}$

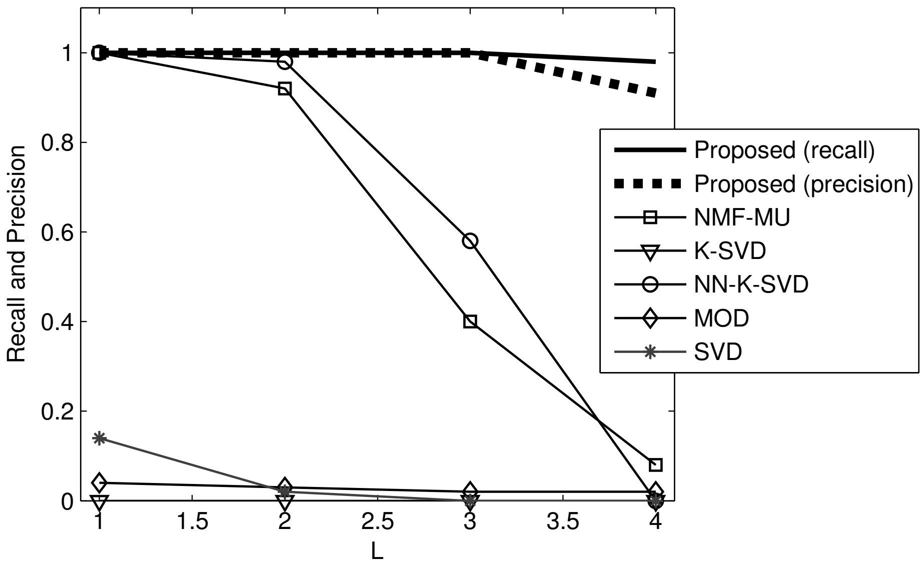


Figure 2.3: Recall and precision when $K = 5$ for $L \in \{1, 2, 3, 4\}$. Ground-truth pitches are initialized randomly over ten trials.

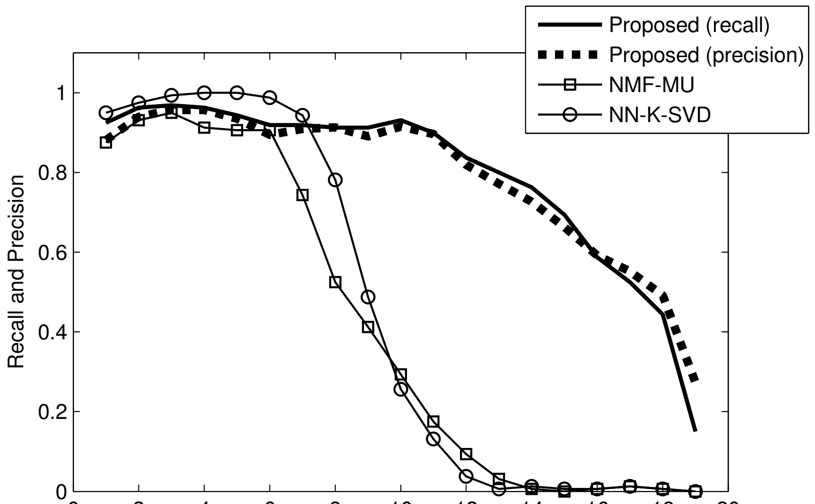


Figure 2.4: Recall and precision when $K = 20$ for $L \in \{1, 2, ..., 19\}$. Ground-truth pitches are initialized randomly over ten trials.

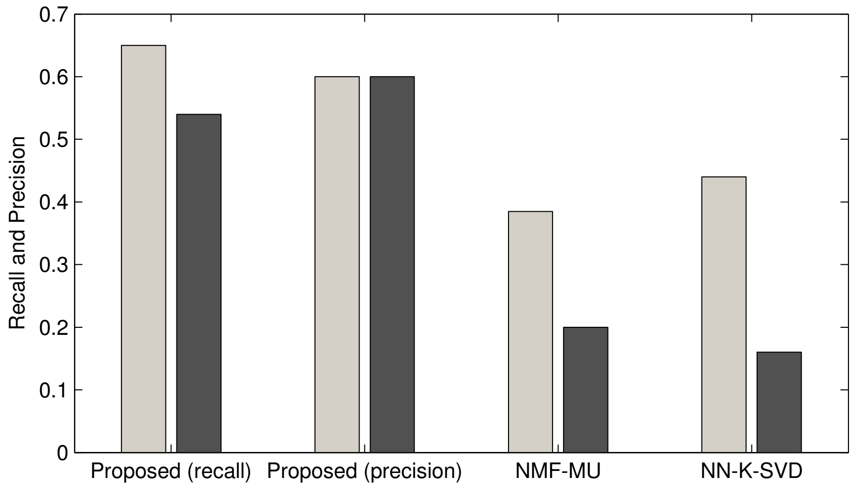


Figure 2.5: Recall and precision when $K = 5$ and $L = 3$ (light gray) and when $K = 10$ and $L = 5$ (dark gray). Pitches are chosen such that large spectral-temporal overlap occurs.

as described earlier. Fig. 2.5 shows that the best recall and precision is achieved by the proposed algorithm. Again, recall and precision and equal for each of the existing methods because K is fixed to its correct value thus creating a one-to-one correspondence between misses and false alarms. Finally, we fix $K = 10$ and $L = 5$ where $f_0 \in \{200, 300, 400, 500, 600, 800, 900, 1000, 1200, 1500\}$. Fig. 2.5 again shows that the best recall and precision is achieved by the proposed algorithm.

2.2 Co-occurrence Constraints

In this section, we discuss another type of constraint, co-occurrence constraints, that adds structure to factorizations such as nonnegative matrix factorization (NMF). In the traditional NMF formulation, given a nonnegative matrix $\mathbf{X}$, the objective of

NMF is to find two nonnegative matrices, $\mathbf{A}$ and $\mathbf{S}$, that minimizes some divergence between $\mathbf{X}$ and $\mathbf{AS}$. The NMF problem was originally popularized by Paatero and Tapper [4], and Lee and Seung later proposed algorithms for solving the NMF problem using multiplicative update rules [5, 6].

When used for source separation, the basic formulation of NMF has notable disadvantages. Objects may require more than a single dictionary atom in order to be approximated accurately. For example, Fig. 2.6 illustrates the decomposition of a spectrogram of an audio signal containing one note played by a violin. Although only one note is played, the vibrato induced by the performer causes the pitch to modulate. As a result, a rank-one approximation is not sufficient to capture this pitch modulation. A user can select multiple dictionary atoms to represent one note. However, in the presence of many other sources, it is unclear which atoms to select. In other words, after learning is complete, there is no straightforward way to cluster multiple dictionary atoms that belong to the same source.

One solution that addresses these problems is to enforce dependence among sets of dictionary atoms by introducing *co-occurrence constraints* – constraints that specify which dictionary atoms are dependent, or co-occur. These co-occurrence constraints have shown to be useful for describing sources with multiple, co-occurring dictionary atoms. Smaragdis et al. [18] proposed the use of cross entropy to enforce the similarity between dictionary atoms belonging to the same source. The atoms are then easily grouped into sets. By decomposing a spectrogram of a drums recording, Smaragdis et al. illustrate that co-occurrence constraints allow each drum sound to be represented more accurately by two dictionary atoms instead of one.

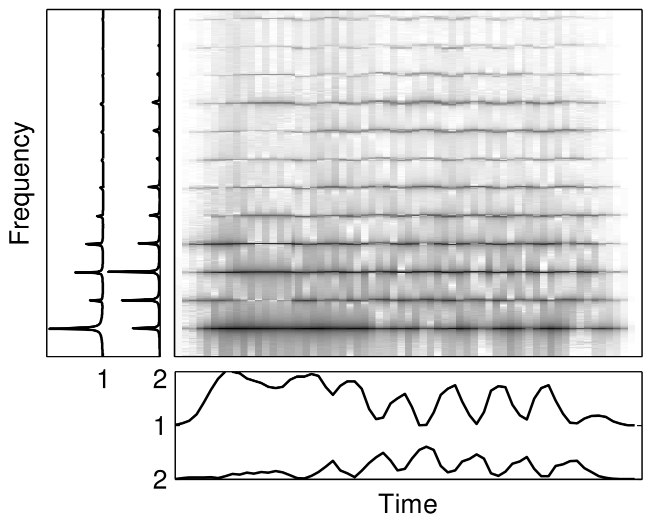


Figure 2.6: Nonnegative matrix factorization of the spectrogram **X** (top right) into **A** (top left) and **S** (bottom right) for one violin note. Two atoms are required to capture the pitch modulation due to vibrato.

In this section, we introduce three new update rules to enforce dependence among dictionary atoms by incorporating co-occurrence constraints into NMF. These rules are conceptually simple, easy to implement, and effective for describing sources using multiple dictionary atoms. First, we formulate the NMF problem with co-occurrence constraints. Then, we derive new update rules for minimizing three common divergence metrics. Finally, we illustrate the use of these update rules in the context of music transcription.

2.2.1 Problem Formulation

The basic NMF problem is formulated as follows. Given a nonnegative matrix $\mathbf{X} \in \mathbb{R}_+^{M\times N}$, we must find nonnegative matrices $\mathbf{A} \in \mathbb{R}_+^{M\times K}$ and $\mathbf{S} \in \mathbb{R}_+^{K\times N}$ that minimize some divergence metric, $\mathbf{d}(\mathbf{X}, \mathbf{AS})$. For any two matrices $\mathbf{X}$ and $\mathbf{Y}$, we define $\mathbf{d}(\mathbf{X}, \mathbf{Y}) = \sum_{m,n} \mathbf{d}(x_{mn}, y_{mn})$. The three divergence metrics we consider are the Euclidean distance,

$$\mathrm{d}_{\mathrm{EUC}}(x, y) = |x - y|^2 \ , \tag{2.7}$$

the Kullback-Leibler divergence,

$$\mathrm{d}_{\mathrm{KL}}(x, y) = x \log \frac{x}{y} - x + y \ , \tag{2.8}$$

and the Itakura-Saito divergence,

$$\mathrm{d}_{\mathrm{IS}}(x, y) = \frac{x}{y} - \log \frac{x}{y} - 1 \ . \tag{2.9}$$

These three divergences are special instances of the generalized Bregman divergence [19].

In this dissertation, for conciseness of notation, we will use $\mathbf{X} \cdot \mathbf{Y}$ to denote element-wise multiplication of matrices $\mathbf{X}$ and $\mathbf{Y}$, $\frac{\mathbf{X}}{\mathbf{Y}}$ to denote element-wise division, and $\mathbf{X}^2$ to denote element-wise exponentiation. Also, we use $\mathbf{1}$ to denote a matrix of ones of appropriate dimension.

Multiplicative update rules for $\mathbf{A}$ and $\mathbf{S}$ have been derived to minimize each of the three divergences [6, 19]. The basic update rules are as follows for the Euclidean distance,

$$\mathbf{A} \leftarrow \mathbf{A} \cdot \frac{\mathbf{X}\mathbf{S}^T}{\mathbf{A}\mathbf{S}\mathbf{S}^T} \qquad \mathbf{S} \leftarrow \mathbf{S} \cdot \frac{\mathbf{A}^T\mathbf{X}}{\mathbf{A}^T\mathbf{A}\mathbf{S}} \ , \tag{2.10}$$

Kullback-Leibler divergence,

$$\mathbf{A} \leftarrow \mathbf{A} \cdot \frac{\frac{\mathbf{X}}{\mathbf{AS}}\mathbf{S}^T}{\mathbf{1}\mathbf{S}^T} \qquad \mathbf{S} \leftarrow \mathbf{S} \cdot \frac{\mathbf{A}^T\frac{\mathbf{X}}{\mathbf{AS}}}{\mathbf{A}^T\mathbf{1}} \ , \tag{2.11}$$

and Itakura-Saito divergence,

$$\mathbf{A} \leftarrow \mathbf{A} \cdot \frac{\frac{\mathbf{X}}{(\mathbf{AS})^2}\mathbf{S}^T}{\frac{\mathbf{1}}{\mathbf{AS}}\mathbf{S}^T} \qquad \mathbf{S} \leftarrow \mathbf{S} \cdot \frac{\mathbf{A}^T\frac{\mathbf{X}}{(\mathbf{AS})^2}}{\mathbf{A}^T\frac{\mathbf{1}}{\mathbf{AS}}} \ . \tag{2.12}$$

Given a choice of divergence metric, the update rules for $\mathbf{A}$ and $\mathbf{S}$ are usually applied alternately. Because $\mathbf{d}(\mathbf{X}, \mathbf{AS})$ is not jointly convex in $(\mathbf{A}, \mathbf{S})$, the global minimum may not necessarily be achieved. However, it is convex in $\mathbf{A}$ and $\mathbf{S}$ individually, this guaranteeing decrease at each iteration.

To introduce co-occurrence constraints, we must influence the value of the inner product $\mathbf{s}_i^T\mathbf{s}_j$, where $\mathbf{s}_k^T$ is the k^{th} row of $\mathbf{S}$. For instance, using the musical examples in Fig. 2.6, a large value for $\mathbf{s}_1^T\mathbf{s}_2$ would indicate that dictionary atoms 1 and 2 co-occur heavily in time, while $\mathbf{s}_1^T\mathbf{s}_2 = 0$ would indicate that the atoms do not co-occur at all. This problem can be formulated as follows:

$$\min_{\mathbf{S}} \mathbf{d}(\mathbf{Q}, \mathbf{S}\mathbf{S}^T) \qquad \text{such that} \ \ \mathbf{S} \in \mathbb{R}_+^{K\times N} \ , \tag{2.13}$$

where $\mathbf{Q} \in \mathbb{R}_+^{K\times K}$ is a pre-defined symmetric matrix such that q_{ij} is low when atoms i and j are not dependent and q_{ij} is high when the atoms are desired to be highly dependent. First, choosing $q_{ii} = 1$ for all i performs normalization upon each row of $\mathbf{S}$. Then we can set $0 \ll q_{ij} \leq 1$ for all pairs of atoms i and j that we desire to be dependent and $0 \leq q_{ij} \ll 1$ for all other pairs of atoms. For d_{KL} and d_{IS}, q_{ij} must be strictly greater than zero for all i and j.

2.2.2 Proposed Update Rules

Following the derivations by Lee and Seung [6], we derive multiplicative update rules for $\mathbf{S}$ to minimize $\mathbf{d}(\mathbf{Q}, \mathbf{S}\mathbf{S}^T)$ for each of the three divergences. Using the Euclidean distance as an example, first we explicitly define $\mathrm{d}_{\mathrm{EUC}}(\mathbf{Q}, \mathbf{S}\mathbf{S}^T)$:

$$\begin{aligned} \mathrm{d}_{\mathrm{EUC}}(\mathbf{Q}, \mathbf{S}\mathbf{S}^T) &= ||\mathbf{Q} - \mathbf{S}\mathbf{S}^T||_F^2 \qquad (2.14) \\ &= \mathrm{tr}((\mathbf{Q} - \mathbf{S}\mathbf{S}^T)^T(\mathbf{Q} - \mathbf{S}\mathbf{S}^T)) \;, \qquad (2.15) \end{aligned}$$

where $||\mathbf{X}||_F^2$ is the squared Frobenius norm of $\mathbf{X}$, and $\mathrm{tr}(\mathbf{X})$ is the trace of $\mathbf{X}$. Next, we differentiate $\mathrm{d}_{\mathrm{EUC}}(\mathbf{Q}, \mathbf{S}\mathbf{S}^T)$ with respect to $\mathbf{S}$:

$$\frac{\partial}{\partial \mathbf{S}} \mathrm{d}_{\mathrm{EUC}}(\mathbf{Q}, \mathbf{S}\mathbf{S}^T) \propto \mathbf{S}\mathbf{S}^T\mathbf{S} - \mathbf{Q}\mathbf{S} \;. \qquad (2.16)$$

Finally, as illustrated by Lee and Seung [6], we construct the multiplicative update term by placing the negative part of $\frac{\partial}{\partial \mathbf{S}}$ in the numerator and the positive part of $\frac{\partial}{\partial \mathbf{S}}$ in the denominator as follows:

$$\mathbf{S} \leftarrow \mathbf{S} \cdot \frac{\mathbf{Q}\mathbf{S}}{\mathbf{S}\mathbf{S}^T\mathbf{S}} \;. \qquad (2.17)$$

In practice, a small positive number ε is added to the numerator and denominator for three reasons. First, ε prevents division by zero. Second, as long as ε is large enough, this update rule guarantees a decrease in $\mathrm{d}_{\mathrm{EUC}}(\mathbf{Q}, \mathbf{S}\mathbf{S}^T)$ at each iteration by restricting $\mathbf{S}$ to lie within a local region around the previous instance of $\mathbf{S}$. Third, including ε in this manner does not alter the divergence metric being minimized.

Update rules for $\mathbf{A}$ can be constructed in similar fashion by minimizing $\mathbf{d}(\mathbf{Q}, \mathbf{A}^T\mathbf{A})$. The choice to impose co-occurrence constraints on $\mathbf{A}$ versus $\mathbf{S}$ depends upon the context of the application.

Therefore, we arrive at the finalized update rule for either $\mathbf{A}$ or $\mathbf{S}$ to minimize the Euclidean distance:

$$\boxed{\mathbf{A} \leftarrow \mathbf{A} \cdot \frac{\mathbf{A}\mathbf{Q} + \varepsilon}{\mathbf{A}\mathbf{A}^T\mathbf{A} + \varepsilon} \quad \text{or} \quad \mathbf{S} \leftarrow \mathbf{S} \cdot \frac{\mathbf{Q}\mathbf{S} + \varepsilon}{\mathbf{S}\mathbf{S}^T\mathbf{S} + \varepsilon}} \, . \tag{2.18}$$

Similar derivations using the Kullback-Leibler and Itakura-Saito divergences yield the following two update rules, respectively:

$$\boxed{\mathbf{A} \leftarrow \mathbf{A} \cdot \frac{\mathbf{A}\frac{\mathbf{Q}}{\mathbf{A}^T\mathbf{A}} + \varepsilon}{\mathbf{A}\mathbf{1} + \varepsilon} \quad \text{or} \quad \mathbf{S} \leftarrow \mathbf{S} \cdot \frac{\frac{\mathbf{Q}}{\mathbf{S}\mathbf{S}^T}\mathbf{S} + \varepsilon}{\mathbf{1}\mathbf{S} + \varepsilon}} \tag{2.19}$$

$$\boxed{\mathbf{A} \leftarrow \mathbf{A} \cdot \frac{\mathbf{A}\frac{\mathbf{Q}}{(\mathbf{A}^T\mathbf{A})^2} + \varepsilon}{\mathbf{A}\frac{\mathbf{1}}{\mathbf{A}\mathbf{A}^T} + \varepsilon} \quad \text{or} \quad \mathbf{S} \leftarrow \mathbf{S} \cdot \frac{\frac{\mathbf{Q}}{(\mathbf{S}\mathbf{S}^T)^2}\mathbf{S} + \varepsilon}{\frac{\mathbf{1}}{\mathbf{S}\mathbf{S}^T}\mathbf{S} + \varepsilon}} \, . \tag{2.20}$$

To incorporate these co-occurrence constraints into NMF, we first formulate the modified minimization problem, using the Euclidean distance again as an example:

$$\min_{\mathbf{A},\mathbf{S}} \mathrm{d}_{\mathrm{EUC}}(\mathbf{X}, \mathbf{AS}) + \lambda\, \mathrm{d}_{\mathrm{EUC}}(\mathbf{Q}, \mathbf{SS}^T) \ , \tag{2.21}$$

where $\lambda > 0$ is a regularization parameter that controls the relative emphasis between $\mathrm{d}_{\mathrm{EUC}}(\mathbf{X}, \mathbf{AS})$ and $\mathrm{d}_{\mathrm{EUC}}(\mathbf{Q}, \mathbf{SS}^T)$. The proper value for λ depends both upon the data as well as the divergence metric used. Then, using the same method of derivation shown earlier by Lee and Seung [6], we can modify the original update rule for minimizing $\mathrm{d}_{\mathrm{EUC}}$ as follows:

$$\mathbf{S} \leftarrow \mathbf{S} \cdot \frac{\mathbf{A}^T\mathbf{X} + \lambda\mathbf{QS} + \varepsilon}{\mathbf{A}^T\mathbf{AS} + \lambda\mathbf{SS}^T\mathbf{S} + \varepsilon} \ . \tag{2.22}$$

Another valid method involves alternating between the updates in Eqs. (2.10) and (2.18). Our experiments have shown both options to be roughly equal in accuracy and execution time.

2.2.3 Experiments

First, we illustrate that the three proposed multiplicative update rules do guarantee decrease in the three divergence metrics at each iteration and that $\mathbf{S}$ does eventually converge. We initialize $\mathbf{S}$ and $\mathbf{Q}$ using the values shown in Fig. 2.7. We use the update rules in (2.18), (2.19), and (2.20) for 200 iterations to solve the minimization problem in (2.13) for each of the three corresponding divergence metrics. We see from Fig. 2.7 that $\mathbf{SS}^T$ does resemble $\mathbf{Q}$ after minimizing any of the three divergence metrics. By examining $\mathbf{S}$ after convergence, for every pair of rows $(\mathbf{s}_i, \mathbf{s}_j)$ such that

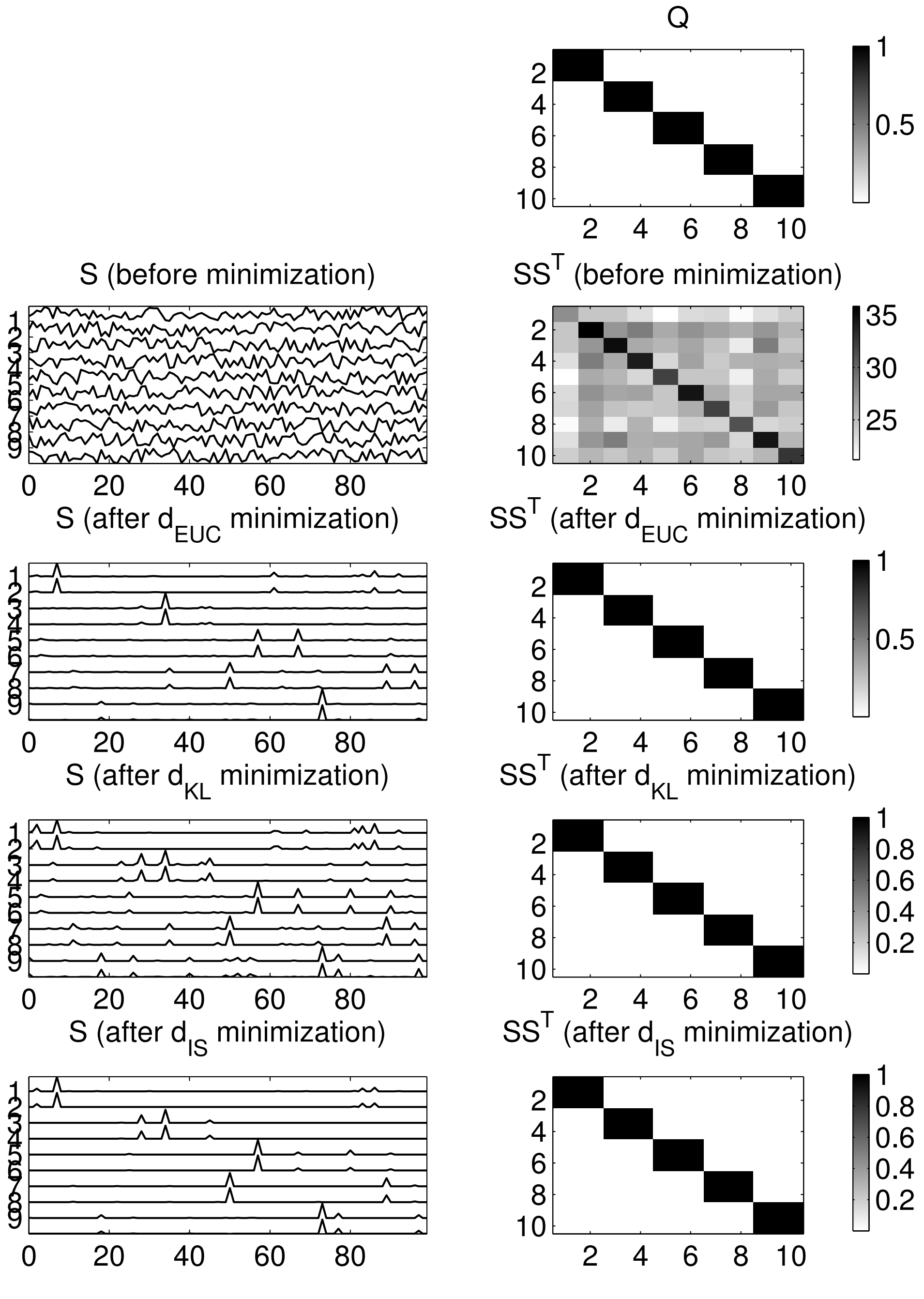


Figure 2.7: Minimization of $\mathbf{d}(\mathbf{Q}, \mathbf{SS}^T)$ for three divergence metrics. Top row: $\mathbf{Q}$. Left column: $\mathbf{S}$ before and after minimization. Right column: $\mathbf{SS}^T$ before and after minimization.

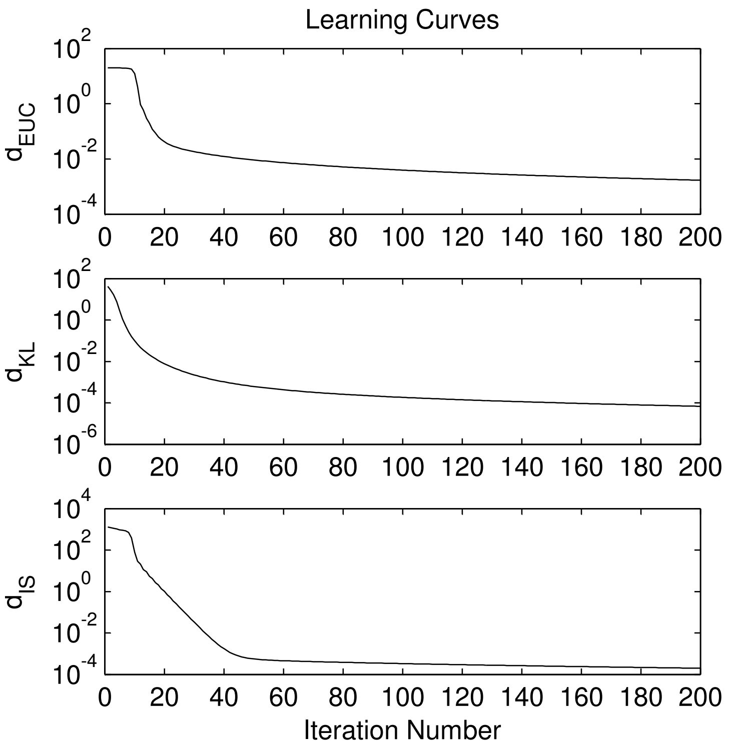


Figure 2.8: Learning curves for the examples in Fig. 2.7. If ε is sufficiently large, then descent of the divergence metrics is guaranteed at each iteration.

$q_{ij} = 1$, we find that $\mathbf{s}_i$ and $\mathbf{s}_j$ are nearly equal.

Fig. 2.8 illustrates the learning curves for each of the three minimizations depicted in Fig. 2.7. The values of $\mathrm{d}_{\mathrm{EUC}}(\mathbf{Q}, \mathbf{SS}^T)$, $\mathrm{d}_{\mathrm{KL}}(\mathbf{Q}, \mathbf{SS}^T)$, and $\mathrm{d}_{\mathrm{IS}}(\mathbf{Q}, \mathbf{SS}^T)$ decrease monotonically as a function of the iteration number, thus confirming that a decrease in the divergence metrics is guaranteed after each iteration of the corresponding update rule as long as ε is sufficiently large. For these experiments, we used $\varepsilon = 0.2$ when minimizing $\mathrm{d}_{\mathrm{EUC}}$, $\varepsilon = 0.2$ when minimizing d_{KL}, and $\varepsilon = 0.6$ when minimizing d_{IS}.

Next, we use the proposed update rules incorporated with NMF to decompose the spectrogram in Fig. 2.9 containing three notes played by a violin. Because each note is pitch-modulated due to vibrato, multiple atoms are required to accurately represent each note. We initialize a dictionary of six atoms into three groups of two. For the following experiments, we minimize d_{KL} which has qualitatively shown to provide better separation than the other divergence metrics.

Following a co-occurrence model by Wang et al. [20], we define pairs of atoms as "must co-occur", "can co-occur", or "cannot co-occur". We set $q_{ij} = 1$ for atoms that must co-occur, $q_{ij} = 10^{-8}$ for atoms that cannot co-occur, and set $q_{ij} = \mathbf{s}_i^T \mathbf{s}_j$ at each iteration for atoms that can co-occur. In this example, we claim that the atoms representing the second note can co-occur with any of the other atoms, while atoms representing the first note cannot co-occur with atoms of the third note. Atoms within the same group must co-occur by definition. Along with the co-occurrence constraints, to improve the likelihood of co-occurrence within groups, we also impose a smoothness constraint on $\mathbf{S}$ using established NMF modifications [21, 22]. Fig. 2.9

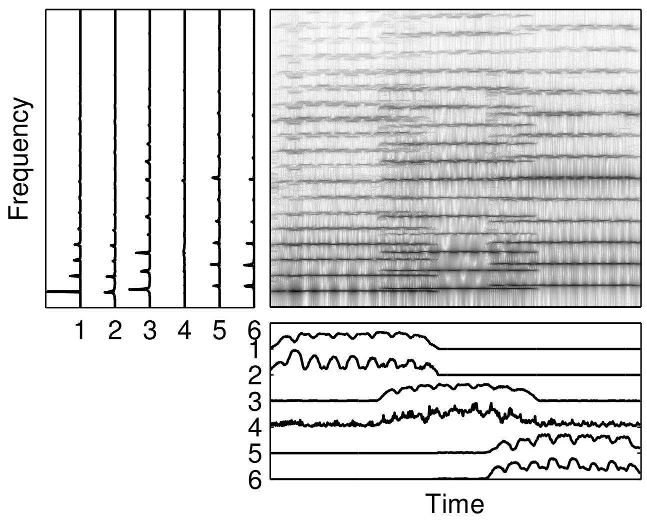


Figure 2.9: Factorization of spectrogram with co-occurrence constraints on $\mathbf{S}$ for three violin notes. Six dictionary atoms are grouped into three sets of two atoms.

shows the results of this procedure after minimization. We see that the algorithm does correctly cluster each pair of atoms belonging to the same note. Fig. 2.10 shows the value of $\mathbf{Q}$ and $\mathbf{SS}^T$ after minimization.

Finally, we perform the same procedure on the spectrogram in Fig. 2.11 containing five drum beats produced by two drums. However, to enforce dependence in the frequency domain among atoms, we now impose co-occurrence constraints on

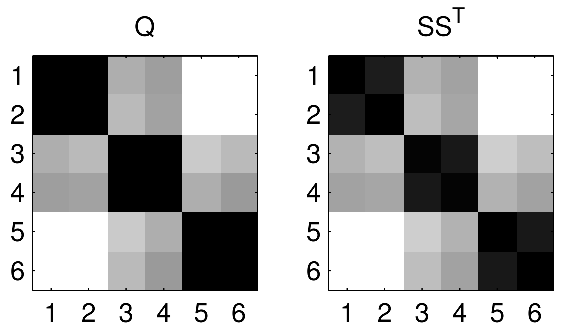


Figure 2.10: $\mathbf{SS}^T$ versus $\mathbf{Q}$ for the example in Fig. 2.9.

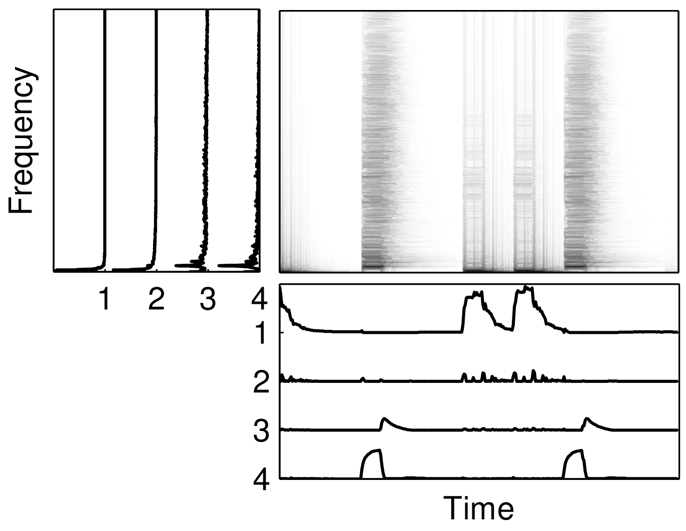


Figure 2.11: Factorization of spectrogram with co-occurrence constraints on $\mathbf{A}$ for five drum beats from kick and snare drums. Four dictionary atoms are grouped into two sets of two atoms.

the columns of $\mathbf{A}$. In a musical context, this constraint is useful whenever there is a percussive sound that emits an initial transient sound followed by a steady-state decaying sound. While the transient and steady-state portions do not co-occur in time, they do overlap considerably in frequency. This behavior is also a property of the piano, xylophone, and similar pitched instruments whose sounds are produced in a percussive manner.

Fig. 2.11 shows a decomposition using two sets of two atoms each. Each pair of atoms are similar in frequency, as shown in Fig. 2.12. The transient and steady-state portions of each beat are visible in the matrix $\mathbf{S}$, particularly for the snare drum which occupies a wider frequency bandwidth. Fig. 2.12 shows the resemblance between $\mathbf{Q}$ and $\mathbf{A}^T\mathbf{A}$ after minimization.

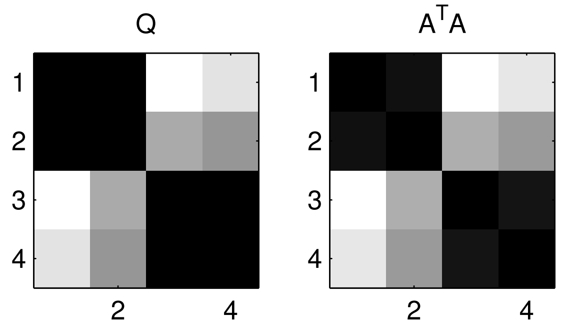


Figure 2.12: $\mathbf{A}^T\mathbf{A}$ versus $\mathbf{Q}$ for the example in Fig. 2.11.

2.3 Summary

First, we presented a novel method of dictionary learning based upon nonnegative K-SVD which can separate sources that are otherwise inseparable using common methods. Despite the simplicity of our algorithm, it performs well for a variety of musical scenarios involving pitched sounds with spectral-temporal overlap. In the future, we plan to investigate the robustness proposed algorithm under different acoustic conditions, particularly for music that includes additive noise or unpitched sources, along with decomposition of time-frequency representations of natural music signals.

Second, we proposed novel multiplicative update rules that impose co-occurrence constraints on either of the matrices produced through NMF. These update rules can minimize different divergence metrics, and they integrate easily with the basic NMF multiplicative updates. The constraints are useful when representing objects with multiple atoms, and they provide a natural way to cluster co-occurring atoms. Examples involving music transcription show that these constraints are operate successfully either in the frequency or time domains.

In the future, we believe that these constraints will become useful in many applications addressed by NMF beyond music transcription and source separation. For either of these tasks, after learning a dictionary of perceptually meaningful atoms, the next stage involves clustering of the dictionary atoms according to their musical source. While some clustering methods already exist, difficulties remain when doing this in an unsupervised manner. If combined with a successful atom clustering method, we believe that the proposed algorithm can offer state-of-the-art accuracy and robustness in music transcription and source separation tasks.

Chapter 3

Spectral-Temporal Instrument Recognition

Musical instrument recognition is a central problem in music information retrieval (MIR). Classification of a musical signal by its instruments can facilitate other MIR tasks such as transcription [23, 24], source separation [25], segmentation [26, 27], genre recognition [28], and search [26].

Most of the research in instrument recognition has focused on classifying isolated, monophonic, instrumental sounds. For these types of sounds, researchers have reported successful results [29, 30, 31, 32, 33, 34, 35]. However, melodic phrases (i.e., multiple notes played in sequence) present a greater challenge. Because musical notes can overlap in both time and frequency, recognition of their features becomes more difficult. Researchers have been able to classify instruments in melodies with moderate success. Published accuracy rates vary widely [36, 35, 37, 13, 10] and depend heavily upon the instrument taxonomy, number of instrument classes, and data set.

Further progress in musical instrument recognition may depend upon recent advances in signal processing such as sparse coding and dictionary learning. Musical signals contain a high amount of spectral and temporal redundancy. Sparse coding methods remove these redundancies in order to efficiently represent or accurately classify the signal. Other MIR tasks such as pitch estimation [38, 39, 40, 41], transcription [14, 15, 40], source separation [21, 18, 42], and genre recognition [43] have already benefited from these methods.

There already exist sparse coding and dictionary learning methods that exploit the *spectral* redundancy among sounds in a musical signal [44, 45, 21, 22, 14, 43]. Many of these methods depend upon nonnegative matrix factorization (NMF) – a popular, convenient, and effective method for decomposing matrices – to obtain low-rank approximations of audio spectrograms of the signal [4, 5, 6]. NMF yields a set of vectors, spectral atoms, which approximately span the frequency space of the spectrogram, and another set of vectors, temporal atoms, which correspond to the temporal activation of each spectral atom.

While these methods are effective in exploiting the spectral redundancy in a signal, redundancy remains in the *temporal* domain. Psychoacoustic studies have shown that spectral information and temporal information are equally important in the definition of acoustic timbre [46]. For example, one widely accepted timbral model, the cortical representation, estimates the spectral *and* temporal modulation content of the auditory spectrogram. This mathematical representation models the primary carrier of timbral information in the early cortical stage of human auditory processing [46].

Classification methods that only utilize spectral information are discarding the potentially useful temporal information that could be used to improve classification performance. Although there are works that use temporal information for instrument recognition [47, 12, 11, 33, 37, 13, 10], the temporal features investigated are often heuristically defined quantities motivated only by mathematical convenience and not by the abundance of research on biological auditory systems that recognize timbre so easily. As a result, temporal features that appear reliable in isolated notes quickly become fragile in melodic music.

In this chapter, we combine advances in dictionary learning, auditory modeling, and music information retrieval to propose a timbral representation that combines a new method of temporal feature extraction with already proven spectral features. The temporal feature, the *multiresolution gamma filterbank response* (MGFR), is computed from the temporal atoms extracted from spectrograms using NMF. Feature extraction and classification is simple, because it only requires linear filtering and a flat classifier. By combining the beneficial elements of NMF, multiresolution analysis, and supervised classification, this algorithm is rigorous and accurate yet without too many "moving parts." Other methods may require preprocessing such as note segmentation [10], pitch estimation [11], or classifiers such as hierarchical clustering [12, 13]. To our knowledge, no other work has attempted to use temporal information extracted from NMF to classify instruments in a systematic manner. Because NMF has become widely popular for facilitating audio classification, this work is useful to practitioners in the field.

We summarize our novel contributions in this chapter as follows:

1. An algorithm for musical instrument recognition that uses the spectral and temporal information produced by NMF in a simple yet principled manner.

2. Analysis of the spectral and temporal properties of the multiresolution gamma

filterbank.

3. Comprehensive experiments on isolated notes and melodic phrases from a diverse set of instruments. We show that our spectral-temporal method is competitive with the state of the art.

Preliminary results of this work were presented earlier by the authors [48]. Here, we present a comprehensive discussion on NMF, spectral analysis of the multiresolution gamma filterbank, illustrative examples on actual musical signals, new experimental results that cover the more difficult task of classifying instruments in melodic phrases, and an extensive comparison against other works in musical instrument recognition, plus further analytical discussions.

We begin in Section 3.1 by motivating the use of the proposed feature extraction method. Unlike existing classification methods that use traditional features such as statistical moments, we extract spectral and temporal features from the input signal in a biologically motivated manner similar to that of the cortical representation. For this, we need an accurate spectral-temporal decomposition. In Section 3.2, we discuss how NMF can be used to obtain such a decomposition. Next, in Section 3.3, we provide a complete mathematical analysis of the multiresolution gamma filterbank, which operates on temporal NMF atoms, and examples of its usage upon musical sounds. In Section 3.4, we define the proposed feature extraction and classification method and formulate the instrument recognition problem.

Finally, in Section 3.5, we test the hypothesis that the proposed method has the capability to improve upon the accuracy achievable by the state of the art in musical instrument recognition. We evaluate the classification accuracy of this feature over many instrumental sounds and contexts. For isolated sounds, when combining spectral and temporal features, the proposed classifier can achieve an accuracy of **92.3%** when tested among 24 instrument classes. For solo melodic phrases, we can achieve an accuracy of **96.2%**, or when using family classifications, **97.4%**. Section 3.6 provides a discussion about the desirable characteristics of our method compared to other methods, and we conclude in Section 3.7.

3.1 Representations of Timbre

In order to successfully classify musical sounds, we must identify the qualities that distinguish musical instruments. Those qualities are encoded through *timbre* – a perceptual property of sound that distinguishes musical sounds of the same pitch [46]. Because timbre is itself an ambiguously defined quality, there exist many valid timbral representations. Most timbral representations characterize the spectral quality of sound. For example, linear predictive coding (LPC) models the spectral envelope using an all-pole model, while mel-frequency cepstral coefficients (MFCCs) model the envelope of the log-spectrum using a filterbank. Other timbral spectral descriptors include spectral flatness, crest, centroid, and spread [49]. However, as mentioned earlier, studies have shown that temporal characteristics of sound influence timbre as much as spectral characteristics [46]. By incorporating temporal information, it is possible that instrument recognition can be improved.

Temporal information can be expressed in different ways. Many attempts to incorporate temporal information use features such as the temporal centroid, spread, skewness, kurtosis, attack time, decay time, slope, and locations of maxima and minima [49, 50, 10]. For isolated notes, these features work well, but for melodic sequences of notes, these features require additional processing such as note onset detection and segmentation which introduces the potential for error propagation. For example, Joder et al. use spectral and temporal features including statistical moments to classify notes that have been segmented from a melodic musical signal using onset detection [10].

Beyond MIR, temporal information has also been used as a feature in automatic speech recognition (ASR). Works by Hermansky and Ellis have explored the use of temporal patterns (TRAPs) of spectral energies to classify phonemes [51, 52, 53, 54, 55, 56]. These TRAPs are features that cover a large window size, e.g. one second long, from a single frequency band. The idea is to capture the temporal evolution of band-limited spectral energy in a vicinity of the underlying phonetic class [52]. Results showed that TRAPs perform slightly better than spectral features such as PLP cepstral coefficients for frame-level classification and slightly worse for word-level classification. However, when the spectral and temporal features were combined, classification improved noticeably over either method individually.

One timbral representation, the *cortical representation*, incorporates both spectral and temporal information. This representation has been utilized to describe the perception of timbre from signals such as speech [57] and music [58, 59]. Essentially, the cortical representation embodies the output of cortical (i.e., from the cortex) cells as sound is processed by earlier stages in the auditory system. Fig. 3.1 illus-

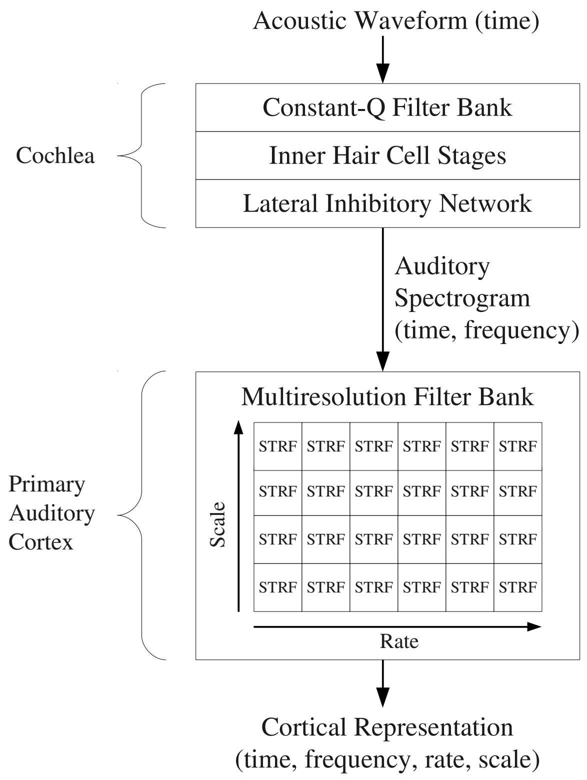

Acoustic Waveform (time)
Cochlea
Constant-Q Filter Bank
Inner Hair Cell Stages
Lateral Inhibitory Network
Auditory Spectrogram (time, frequency)
Primary Auditory Cortex
Multiresolution Filter Bank
STRF
Scale
Rate
Cortical Representation (time, frequency, rate, scale)

Figure 3.1: The cochlear and early cortical stages of the auditory system, shown here, inspire our proposed method. The auditory spectrogram is convolved across time and frequency with STRFs of different rates and scales to produce the four-dimensional cortical representation. This multiresolution representation is believed to carry timbral information.

trates the relationship between the cochlear and early cortical stages of processing in the mammalian auditory system. The cochlear stage models the transformation by the cochlea of an acoustic input signal into a neural representation known as the auditory spectrogram, while the early cortical stage models the analysis of the auditory spectrogram by the primary auditory cortex.

One property of cortical cells, the spectrotemporal receptive field (STRF), summarizes the way a single cortical cell responds to a stimulus. Mathematically, the STRF is like a two-dimensional impulse response defined across time and frequency.

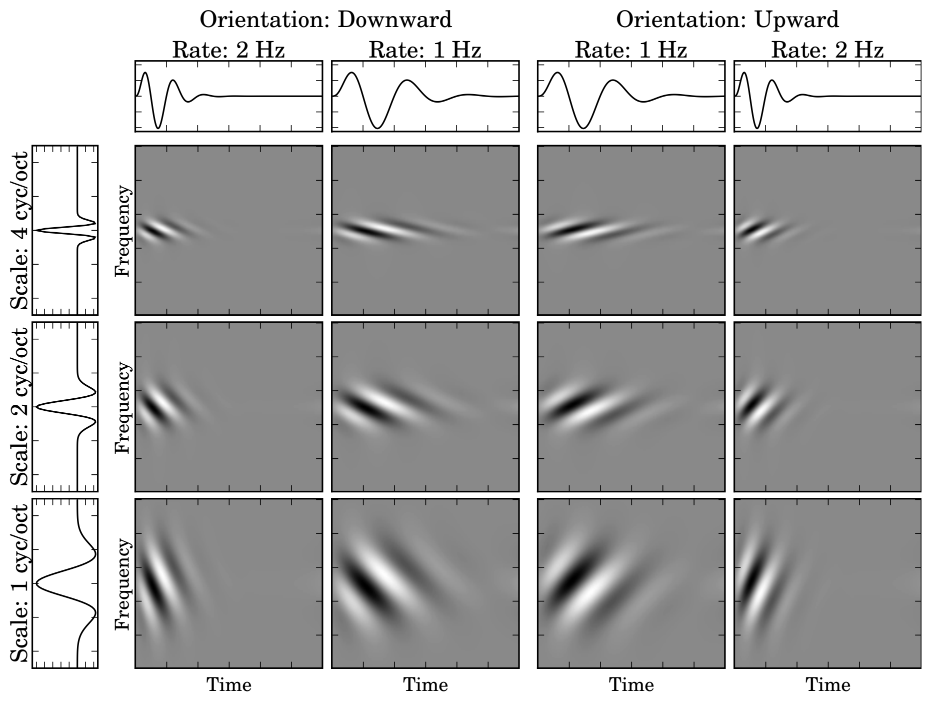


Figure 3.2: Twelve example STRFs. Together, they constitute a filterbank similar to the one proposed in Section 3.3. The left six STRFs select downward-modulating frequencies, and the right six STRFs select upward-modulating frequencies. Top row: seed functions for rate determination. Left column: seed functions for scale determination.

Each STRF has three parameters: scale, rate, and orientation. Scale defines the spectral resolution of an STRF, rate defines its temporal resolution, and orientation determines if the STRF selects upward or downward frequency modulations. Fig. 3.2 illustrates the STRF as a function of these three parameters. Each cortical cell can be interpreted as a filter whose impulse response is an STRF with a particular rate, scale, and orientation. Therefore, a collection of cortical cells constitutes a filterbank. In fact, the cortical representation is mathematically equivalent to a multiresolution wavelet filterbank [60].

Despite the biological relationship between the cortical representation and timbre, this representation has disadvantages for classification. First, because the cortical representation is a complex-valued four-dimensional filterbank output, it is massively redundant. Like many types of redundant data, the cortical representation could benefit from some form of coding, decomposition, or dimensionality reduction. However, proper application of these tools to the cortical representation for engineering purposes such as speech recognition and MIR is not yet well understood. Therefore, these are ongoing areas of research [61, 59]. Second, the STRF is not time-frequency separable [60]. In other words, computation of the cortical representation cannot be decomposed into two procedures that operate on the time and frequency dimensions separately. Because spectral and temporal information require different classification methods, this obstacle impedes classification.

Like the cortical representation, the spectrogram computed via short-time Fourier transform (STFT) reveals spectral and temporal redundancies in a musical signal. Although the spectrogram is still redundant, it does not have the disadvantages in classification mentioned earlier for the cortical representation. Despite the lack of a direct physiological analogy, the spectrogram computed via STFT is easily decomposed into a set of spectral and temporal basis vectors, particularly for musical signals [14].

3.2 Nonnegative Matrix Factorization

Among the decomposition methods used upon time-frequency representations, one of the most popular is nonnegative matrix factorization (NMF) [4, 5, 6]. Given an element-wise nonnegative matrix $\mathbf{X} \in \mathbb{R}_+^{M \times N}$, NMF attempts to find two nonnegative matrices, $\mathbf{A} \in \mathbb{R}_+^{M\times K}$ and $\mathbf{S} \in \mathbb{R}_+^{K\times N}$, that minimize some divergence between $\mathbf{X}$ and $\mathbf{AS}$. Among the algorithms that perform this minimization, one of the most convenient algorithms uses a multiplicative update rule during each iteration in order to maintain nonnegativity of the matrices $\mathbf{A}$ and $\mathbf{S}$ [6]. When the divergence measure is the Euclidean distance:

$$d_{EUC}(\mathbf{X},\mathbf{Y}) = \sum_{m,n} |x_{mn} - y_{mn}|^2 \ , \tag{3.1}$$

then the update rule is

$$\mathbf{A} \leftarrow \mathbf{A} \cdot \frac{\mathbf{X}\mathbf{S}^T}{\mathbf{A}\mathbf{S}\mathbf{S}^T} \ , \tag{3.2}$$

$$\mathbf{S} \leftarrow \mathbf{S} \cdot \frac{\mathbf{A}^T\mathbf{X}}{\mathbf{A}^T\mathbf{A}\mathbf{S}} \ , \tag{3.3}$$

where $\mathbf{X}\cdot\mathbf{Y}$ and $\mathbf{X}/\mathbf{Y}$ denotes element-wise multiplication and division, respectively. When the divergence measure is the Kullback-Leibler divergence:

$$d_{KL}(\mathbf{X},\mathbf{Y}) = \sum_{m,n} x_{mn} \log \frac{x_{mn}}{y_{mn}} - x_{mn} + y_{mn} \ , \tag{3.4}$$

then the update rule is

$$\mathbf{A} \leftarrow \mathbf{A} \cdot \frac{\frac{\mathbf{X}}{\mathbf{AS}}\mathbf{S}^T}{\mathbf{1}\mathbf{S}^T} \ , \tag{3.5}$$

$$\mathbf{S} \leftarrow \mathbf{S} \cdot \frac{\mathbf{A}^T\frac{\mathbf{X}}{\mathbf{AS}}}{\mathbf{A}^T\mathbf{1}} \ , \tag{3.6}$$

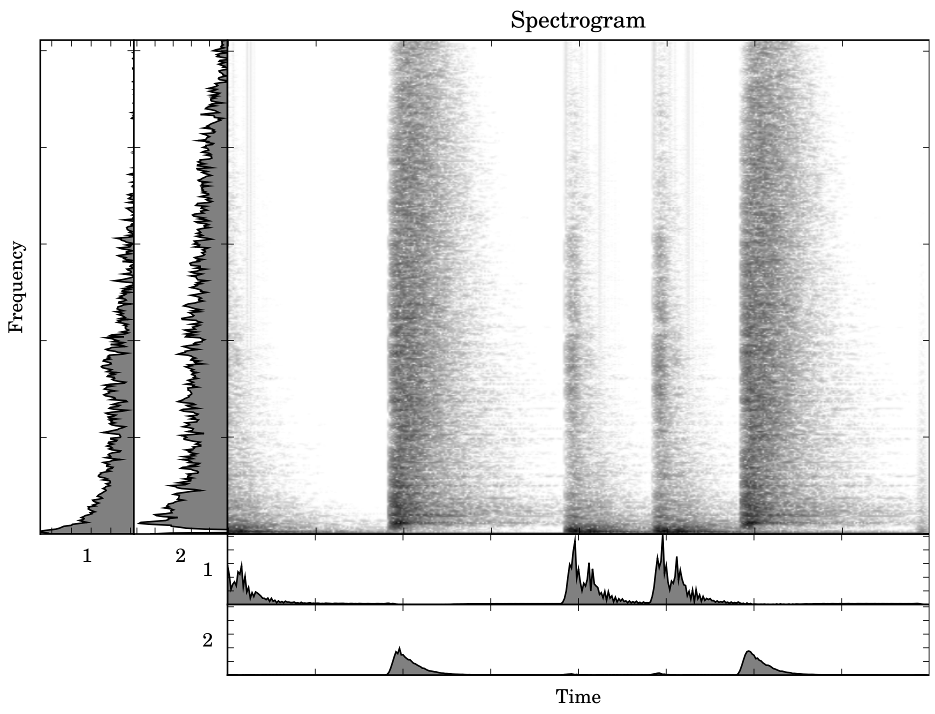


Figure 3.3: The NMF of a spectrogram drum beats. Component 1: kick drum. Component 2: snare drum. Top right: **X**. Left: **A**. Bottom: **S**.

where **1** is a matrix of ones. Other divergences require different update rules [19]. Although there is no explicit sparsity cost to be minimized, NMF and sparse coding algorithms achieve similar objectives. Research has shown that the nonnegativity constraint does enforce sparsity of the gain coefficients under certain conditions, including prewhitening of the observations and the absence of noise [62].

Many researchers have already demonstrated the usefulness of NMF for separating a musical signal into individual notes [38, 18, 43, 14, 21]. By first expressing a time-frequency representation of the signal as a matrix, these methods decompose the matrix into a summation of a few individual *atoms*, each corresponding to one musical source or one note. The nonnegativity constraint makes sense considering that sources are either present or absent; a source is never "subtracted" from a signal in which it is already absent. As a result, algorithms that were previously only successful for isolated sounds can be successfully applied to a mixture of sounds by operating on each atom individually.

Fig. 3.3 illustrates the use of NMF upon the spectrogram of a musical signal. We define each column of $\mathbf{A}$ as a *spectral atom* and each row of $\mathbf{S}$ as a *temporal atom*. The factorization reveals the presence of spectral atoms that reoccur over several time units as well as temporal atoms that reoccur over several frequency bands. These temporal atoms usually resemble envelopes of known sounds, particularly in musical signals. For example, observe the difference between the profiles of the temporal atoms in Fig. 3.3. The three beats generated by the kick drum share the same temporal profiles, and the two beats generated by the snare drum share the same profiles. This general observation motivates the hypothesis that the energy profile of temporal NMF atoms is a valid timbral representation that can be used to classify instruments.

In the next section, we propose one technique that extracts timbral information from temporal NMF atoms similar to that of the cortical representation. Our technique uses a *multiresolution gamma filterbank* to perform multiresolution analysis upon the factorized spectrogram. However, unlike the cortical representation, this multiresolution analysis is particularly suited to the energy profiles contained in the temporal NMF atoms.

3.3 Multiresolution Gamma Filterbank

We propose the use of a multiresolution gamma filterbank, a collection of gamma filters, to extract information from temporal NMF atoms. For this work, we define the gamma kernel to be

$$g(t;n,b) = \alpha t^{n-1} e^{-bt} u(t) \tag{3.7}$$

where $b > 0$, $n \geq 1$, $u(t)$ is the unit step function, and

$$\alpha = \sqrt{\frac{(2b)^{2n-1}}{\Gamma(2n-1)}} \tag{3.8}$$

ensures that $\int |g(t;n,b)|^2 dt = 1$ for any value of n and b, where $\Gamma(n)$ is the Gamma function. Let I be the total number of gamma filters in the filterbank. For each $i \in \{1, ..., I\}$, define the correlation kernel (i.e., time-reversed impulse response) of each gamma filter to be

$$g_i(t) = g(t; n_i, b_i). \tag{3.9}$$

The set of kernels $\{g_1, g_2, ..., g_I\}$ defines the *multiresolution gamma filterbank*. Fig. 3.4 illustrates some example kernels of the filterbank.

For each i, let the filter output be the cross-correlation between the input atom, $s(t)$, and the kernel, $g_i(t)$:

$$y_i(\tau) = \int_{-\infty}^{\infty} s(t) g_i(t-\tau) dt. \tag{3.10}$$

The set of outputs $\{y_1, y_2, ..., y_I\}$ from the filterbank is called the *multiresolution*

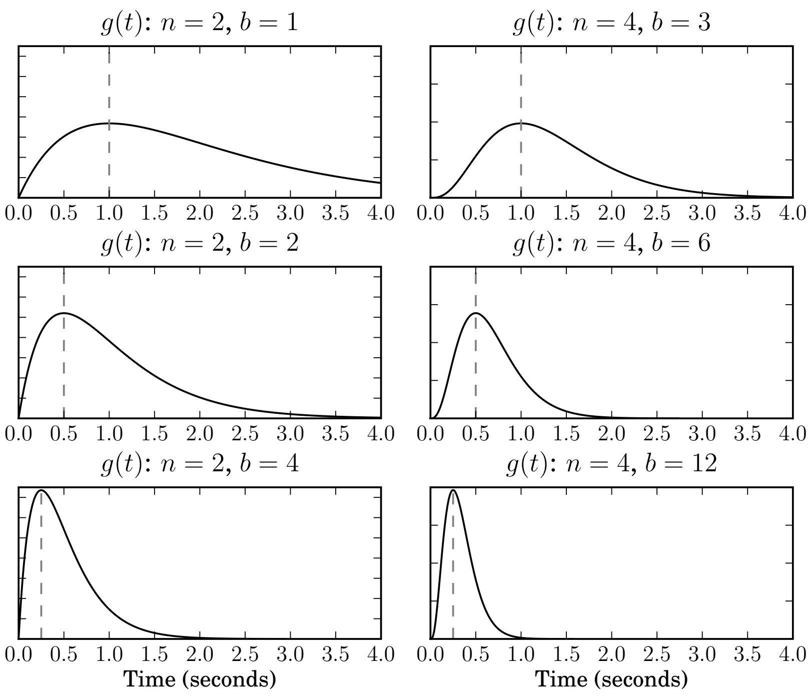


Figure 3.4: Example kernels of gamma filters. The dashed vertical line indicates the location of the maxima. Left column: $n = 2$. Right column: $n = 4$.

gamma filterbank response (MGFR).

First, we examine temporal properties of the gamma filter. We define the *attack time* of the kernel $g(t)$ to be the time elapsed until the kernel achieves its maximum. By differentiating $\log g(t)$, we determine the attack time to be

$$t_a = (n-1)/b \text{ seconds.} \tag{3.11}$$

Fig. 3.4 illustrates the relationship between the attack time and the parameter b. Also, as t becomes large, $\log g(t) \approx -bt$ plus a constant. Therefore, b is the decay parameter of $g(t)$, where we define the *decay rate* of $g(t)$ to be

$$r_d = 20b \log_{10} e \approx 8.7b \text{ dB per second.} \tag{3.12}$$

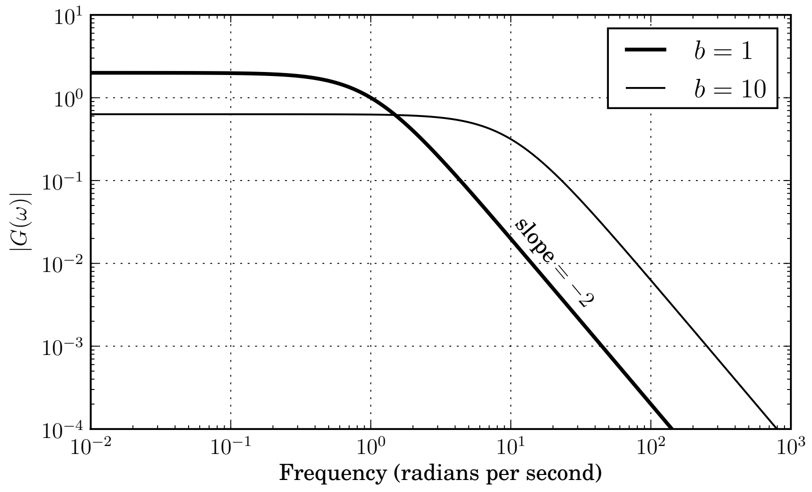


Figure 3.5: Log-log magnitude responses of two gamma filters, $b = 1$ and $b = 10$, for $n = 2$.

Together, these two temporal properties imply that a gamma kernel with *any* attack time and decay rate can be created from the proper combination of n and b.

Next, we examine spectral properties. The Fourier transform, $G(\omega)$, of $g(t; n, b)$ is

$$G(\omega) \propto \left(\frac{1}{b + j\omega}\right)^{n}, \tag{3.13}$$

and the magnitude response is

$$|G(\omega)| \propto \left(\frac{1}{b^2 + \omega^2}\right)^{n/2}. \tag{3.14}$$

Figs. 3.5 and 3.6 illustrate the magnitude responses of the gamma filterbank for different values of n and b. The cutoff frequency for each filter is $\omega_c = b$ radians per second, and the stopband slope is $-n$ on the log-log scale, or $20n$ dB attenuation per decade.

Fig. 3.7 illustrates the operation of the multiresolution gamma filterbank.

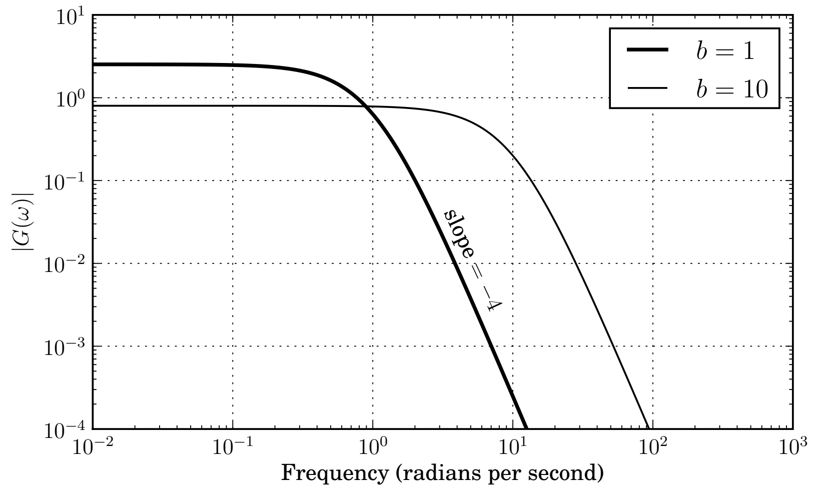


Figure 3.6: Log-log magnitude responses of two gamma filters, $b = 1$ and $b = 10$, for $n = 4$.

When a temporal NMF atom is sent through the multiresolution gamma filterbank, the MGFR reveals the strength of the attacks and decays of the atom's envelope for different values for n and b. Observe how the filterbank response is largest for those filters whose attack time matches that of the input atom.

The multiresolution gamma filterbank behaves like a set of STRFs. Both systems perform multiresolution analysis on the input data. Each STRF passes a different spectral-temporal pattern depending upon the rate and scale. In fact, the seed function used to determine the rate of an STRF is a gammatone kernel – a sinusoid whose envelope is a gamma kernel. By altering the parameters of the gammatone kernel, STRFs can select different rates. Similarly, in the multiresolution gamma filterbank, each filter passes different envelope shapes depending upon the parameters n and b which completely characterize the attack and decay of the envelope. Intuitively, the filter with kernel $g_i(t)$ passes envelopes with attack times equal to $(n_i - 1)/b_i$ seconds and envelopes with decay rates equal to $8.7b_i$ dB per

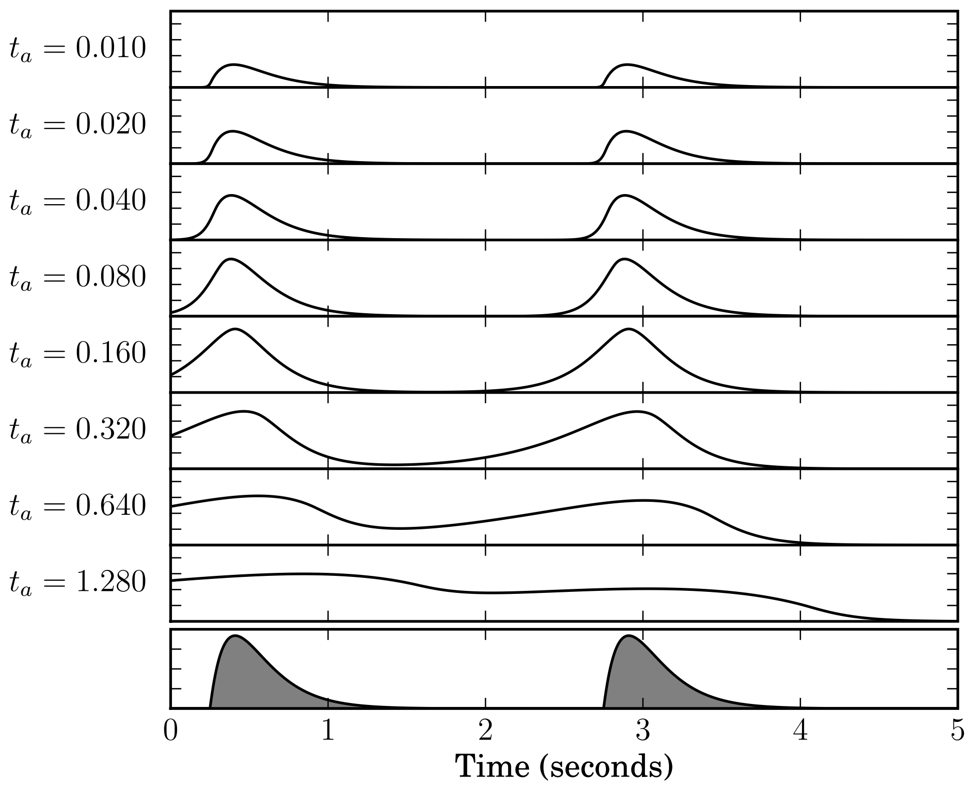


Figure 3.7: Top: MGFR as a function of time for $n = 2$. Bottom: input temporal atom containing two pulses with attack times of 160 ms.

second.

As we see from Fig. 3.7, the MGFR is a function of time. For classification, we must still extract a single feature vector from the MGFR without respect to time. In the next section, we describe how to extract and classify features from the MGFR to perform musical instrument recognition, and we also describe how to extract features from spectral NMF atoms.

3.4 Proposed Feature Extraction and Classification

Here, we describe how to extract and classify spectral and temporal features. First, to obtain a feature vector from spectral NMF atoms, we compute the mel-frequency cepstral coefficients (MFCCs) of each spectral atom. For each atom, these MFCCs form the spectral feature vector, $\mathbf{z}_S$. The MFCCs have been frequently used to describe the timbral quality of speech and music [26], and other researchers have also used MFCCs to classify spectral NMF atoms [63]. For more about the MFCC, we refer the reader to other sources [64, 65].

Next, we describe how to extract a shift-invariant temporal feature from the MGFR introduced in the previous section. For each filter response, we compute a norm:

$$z_i = \left(\int_{-\infty}^{\infty} |y_i(t)|^p dt \right)^{1/p} . \tag{3.15}$$

The vector $\mathbf{z}_T = [z_1, z_2, ..., z_I]$ is the temporal feature vector. To eliminate scaling ambiguities among the input atoms, every feature vector $\mathbf{z}_T$ is normalized to have unit Euclidean norm. Fig. 3.8 illustrates the feature vector $\mathbf{z}_T$ when $p = \infty$ for four instrumental sounds. The feature vector $\mathbf{z}_T$ reveals the sharp attacks of the kick and snare drums and the slower attacks of the trumpet and violin. The feature also reveals the fast decays of the kick drum and trumpet and the relatively slower decays of the snare drum and violin.

Different choices of p provide different interpretations of $\mathbf{z}_T$. When $p = \infty$,

$$z_i = \max_{\tau} y_i(\tau) = \max_{\tau} \int_{-\infty}^{\infty} s(t) g_i(t - \tau) dt, \tag{3.16}$$

i.e., z_i indicates the maximum value of the inner product between $s(t)$ and $g_i(t - \tau)$ for all possible lags τ, and therefore $\mathbf{z}_T$ describes the shape of the loudest note or

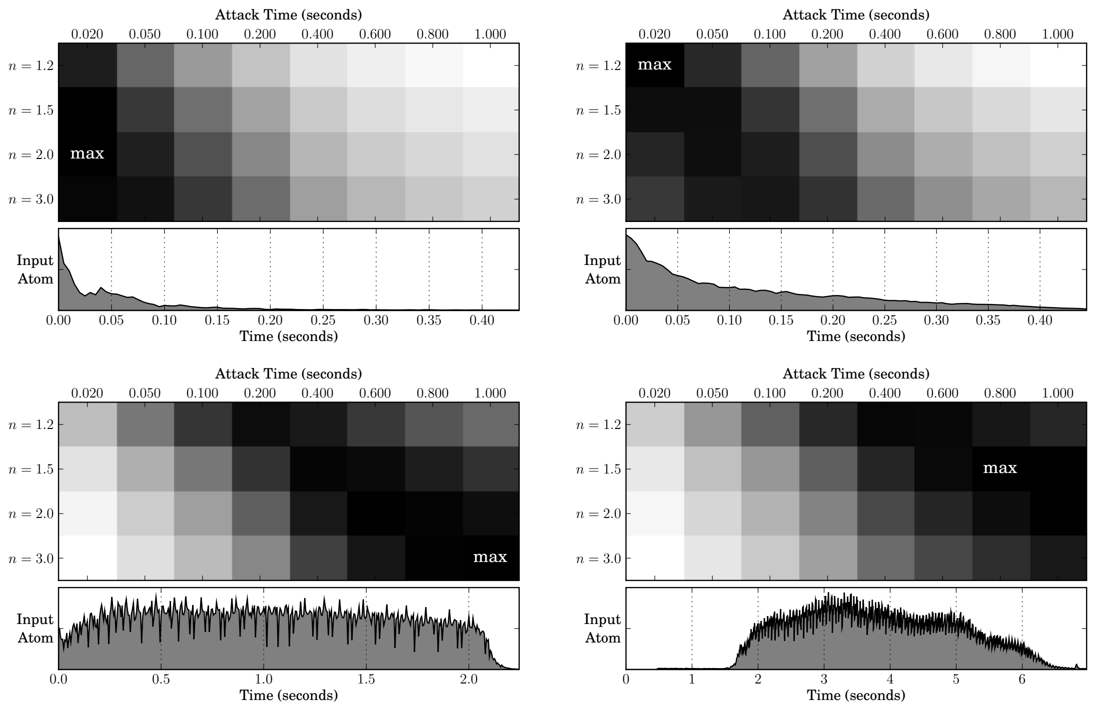


Figure 3.8: Feature vector $\mathbf{z}_T$ for $p = \infty$, where $\mathbf{z}_T \in \mathbb{R}^{32}$, and the corresponding temporal atom. Top row: kick drum and snare drum. Bottom row: trumpet and violin.

beat in the temporal atom regardless of the presence of any quieter notes. When $p = 1$,

$$z_i = \int_{-\infty}^{\infty} y_i(\tau) d\tau = \int_{-\infty}^{\infty} s(t) dt \int_{-\infty}^{\infty} g_i(\tau) d\tau, \tag{3.17}$$

i.e., z_i does not depend upon the shape of $s(t)$. After normalization, $\mathbf{z}_T$ will be identical for any sample. Therefore, $p = 1$ is not a good choice. When $p = 2$, $\mathbf{z}_T$ contains information from the entire temporal atom including both loud and quiet notes, but it weighs louder notes more heavily. As Section 3.5 will show, p has little impact on classification performance; both $p = 2$ and $p = \infty$ perform approximately the same.

The *proposed feature extraction algorithm* is summarized below.

1. Perform NMF on the magnitude spectrogram, $\mathbf{X}$, to obtain $\mathbf{A}$ and $\mathbf{S}$ using updates in (3.2) and (3.3) or (3.5) and (3.6).

2. For each spectral atom (i.e., row of $\mathbf{A}$), compute the MFCCs, $\mathbf{z}_S$.

3. For each temporal atom (i.e., row of $\mathbf{S}$), compute the MGFR in (3.10).

4. From the MGFR, compute the feature vector $\mathbf{z}_T$ in (3.15).

5. Form the feature vector $\mathbf{z} = [\mathbf{z}_S \; \mathbf{z}_T]$.

Finally, we formulate the instrument recognition problem as a typical supervised classification problem: given a set of training features extracted from signals of known musical instruments, identify the instrument present in a test signal. To perform supervised classification, spectral and temporal atoms are extracted from training signals of known musical instruments using NMF. The feature vector $\mathbf{z}$ plus its instrument label are used for training. To predict the label of an unknown sample, $\mathbf{z}$ is extracted from the unknown sample and classified using the trained model.

A major advantage of the proposed feature extraction and classification procedure is its *simplicity*. The proposed system requires no rule-based preprocessing. Unlike other systems that contain safeguards, thresholds, and hierarchies, the proposed system uses straightforward filtering and a flat classifier. As the next section shows, this simple procedure can achieve state-of-the-art accuracy for instrument recognition.

n	b	t_a	n	b	t_a
1.2	0.200	1.000	1.5	0.500	1.000
1.2	0.250	0.800	1.5	0.625	0.800
1.2	0.333	0.600	1.5	0.833	0.600
1.2	0.500	0.400	1.5	1.25	0.400
1.2	1.00	0.200	1.5	2.50	0.200
1.2	2.00	0.100	1.5	5.00	0.100
1.2	4.00	0.050	1.5	10.0	0.050
1.2	10.0	0.020	1.5	25.0	0.020
2.0	1.00	1.000	3.0	2.00	1.000
2.0	1.25	0.800	3.0	2.50	0.800
2.0	1.67	0.600	3.0	3.33	0.600
2.0	2.50	0.400	3.0	5.00	0.400
2.0	5.00	0.200	3.0	10.0	0.200
2.0	10.0	0.100	3.0	20.0	0.100
2.0	20.0	0.050	3.0	40.0	0.050
2.0	50.0	0.020	3.0	100	0.020

Table 3.1: Parameters for the 32-filter gamma filterbank used in the following experiments.

3.5 Experiments

We perform three sets of experiments on two data sets: isolated sounds and solo melodic phrases. From each input signal, $x(t)$, we obtain the magnitude spectrogram, $\mathbf{X}$, via STFT using frames of length 46.4 ms (i.e., 2048/44100) windowed using a Hamming window and a hop size of 10.0 ms. Then, we perform NMF using the Kullback-Leibler update rules in (3.5) and (3.6) to obtain $\mathbf{A}$ and $\mathbf{S}$. Research has shown these update rules to achieve good separation among instruments in musical signals [14, 63, 21]. For all experiments, we use a multiresolution gamma filterbank of thirty-two filters with the parameters shown in Table 3.1. By empirical observation, these attack times and decay rates cover a wide range of sounds produced by common musical instruments.

We initially tested two supervised classifiers: support vector machines (SVM) and gaussian mixture models (GMM). Although both tools have been successfully used for many pattern classification tasks, all of our tests have shown the SVM to be more accurate, and therefore we only report those results. For the SVM, we use the LIBSVM implementation [66] with the radial basis kernel. For multiple classes, LIBSVM uses the one-versus-one classification strategy by default. The remaining programs and simulations were written entirely in Python using the SciPy package [67].

3.5.1 Isolated Sounds

First, we compare the abilities of spectral and temporal features to classify isolated, monophonic sounds by instrument. Portions of the experiments in this subsection were presented earlier by the authors [48]. The data set for these experiments combines samples from the University of Iowa database of Musical Instrument Samples [68], McGill University Master Samples [69], the OLPC Samples Collection [70], and the Freesound Project [71]. All of these samples consist of isolated sounds generated by real musical instruments. We have parsed the audio files such that each file consists of a single musical note (for harmonic sounds) or beat (for percussive sounds).

The magnitude spectrogram is then decomposed using NMF with an inner dimension of $K = 1$. For this data set, our experiments have shown that $K = 1$ suffices to represent a majority of the information in isolated notes. The user is welcome to use a larger choice for K; the result is simply a greater number of atoms produced by NMF. In such a case, a few of the atoms will retain most of the information while the other atoms will be insignificant. Subsequent methods can be used to select which atoms are meaningful. There are existing works devoted to the proper choice of K [72, 73], but this topic is beyond the scope of this dissertation.

In total, there are 3907 feature vectors collected among twenty-four instrument classes: bassoon, cello, cello pizzicato, B-flat clarinet, flute, glockenspiel, acoustic guitar, french horn, kick (bass) drum, marimba, oboe, piano, alto saxophone, snare drum, timpani, tom-toms, trombone, trumpet, tuba, viola, viola pizzicato, violin, violin pizzicato, and xylophone. These twenty-four instrument classes represent each of the common instrument families: strings, woodwinds, brass, and percussion. Table 3.2 summarizes this data set. With few exceptions [31], this selection of instruments is more comprehensive than any existing work on isolated instrument recognition. Recognition accuracy for class c is defined to be the percentage of the feature vectors whose true class is c that are correctly classified by the SVM as belonging in class c. Overall recognition accuracy is the average of the accuracy rates for each class.

As a control experiment, Experiment A1 evaluates the classification ability using only spectral features computed from MFCCs. From each column of $\mathbf{A}$, we extract 32 MFCCs as described in Section 3.4 with center frequencies logarithmically spaced over 5.3 octaves between 110 Hz and 3951 Hz. These MFCCs form the feature vector $\mathbf{z}_S$. From the 3907 32-dimensional feature vectors, we evaluate classification performance through ten-fold cross validation. Fig. 3.9 illustrates the confusion

Instrument	#	A1	A2(2)	A2(∞)	A3(2)	A3(∞)
Bassoon	131	99.2	57.3	75.6	97.7	96.9
Clarinet	145	80.7	62.1	73.1	84.8	86.2
Flute	236	84.7	55.9	60.6	87.7	89.0
Oboe	118	72.0	81.4	77.1	87.3	91.5
Saxophone	196	93.4	62.2	65.8	91.3	86.7
Horn	92	80.4	66.3	62.0	80.4	85.9
Trombone	99	93.9	65.7	53.5	96.0	89.9
Trumpet	236	97.5	81.8	82.2	97.9	97.9
Tuba	111	98.2	75.7	75.7	97.3	99.1
Cello	349	94.8	84.5	89.7	96.0	97.4
Viola	309	94.2	67.0	67.6	91.9	90.9
Violin	390	97.2	82.3	86.2	96.4	96.2
Cello Pizz.	321	98.1	71.7	87.5	98.1	98.4
Viola Pizz.	254	99.6	60.6	81.9	100.0	99.6
Violin Pizz.	315	97.5	70.8	85.4	98.4	99.0
Glockensp.	10	100.0	80.0	90.0	100.0	100.0
Guitar	27	51.9	29.6	29.6	66.7	63.0
Marimba	39	46.2	25.6	25.6	82.1	79.5
Piano	260	95.0	63.8	89.2	98.5	98.5
Xylophone	13	61.5	61.5	53.8	76.9	84.6
Kick	90	98.9	97.8	95.6	100.0	100.0
Snare	86	96.5	95.3	88.4	98.8	98.8
Timpani	47	85.1	27.7	61.7	85.1	87.2
Toms	33	100.0	100.0	90.9	100.0	100.0
Total	3907	88.2	67.8	72.9	92.1	**92.3**

Table 3.2: Experiments A1, A2, and A3: sample sizes and accuracy rates.

matrix for Experiment A1, and Table 3.2 shows the accuracy rates for each class. The average of the 24 accuracy rates is 88.2%. We notice some understandable misclassifications. For example, 18.5% of guitar samples are misclassified as cello pizzicato and 14.8% are misclassified as piano. 5.5% of clarinet samples and 13.6% of oboe samples are misclassified as flute. 10.3% of marimba samples are misclassified as xylophone. In general, these spectral features can accurately classify the drums, brass, and string instruments. However, accuracy is poor among the woodwinds and pitched percussive instruments. Some of these misclassifications are due to an imbalance in the sample size of each class [74]. Despite its ability to improve the average accuracy rate, the reduction of class imbalance in supervised classification is beyond the scope of this dissertation.

Experiment A2 evaluates the classification ability using only temporal features computed from the MGFR with the parameters shown in Table 3.1. One temporal feature vector $\mathbf{z}_T$ is computed for each temporal NMF atom as described in Section V. Like Experiment A1, we evaluate classification performance through ten-fold cross validation among the 3907 32-dimensional feature vectors. We perform the experiment for $p = 2$ and again for $p = \infty$. The average accuracy rate is 67.8% when $p = 2$ and 72.9% when $p = \infty$. Fig. 3.10 illustrates the confusion matrix for Experiment A2 when $p = \infty$, and Table 3.2 shows the accuracy rates for each class. We observe that temporal features alone do not classify instruments as well as spectral features. Nevertheless, for 11 out of the 24 classes, accuracy remains above 80% when $p = \infty$. In particular, there are few misclassifications between percussion instruments and non-percussion instruments. Most misclassifications occur within

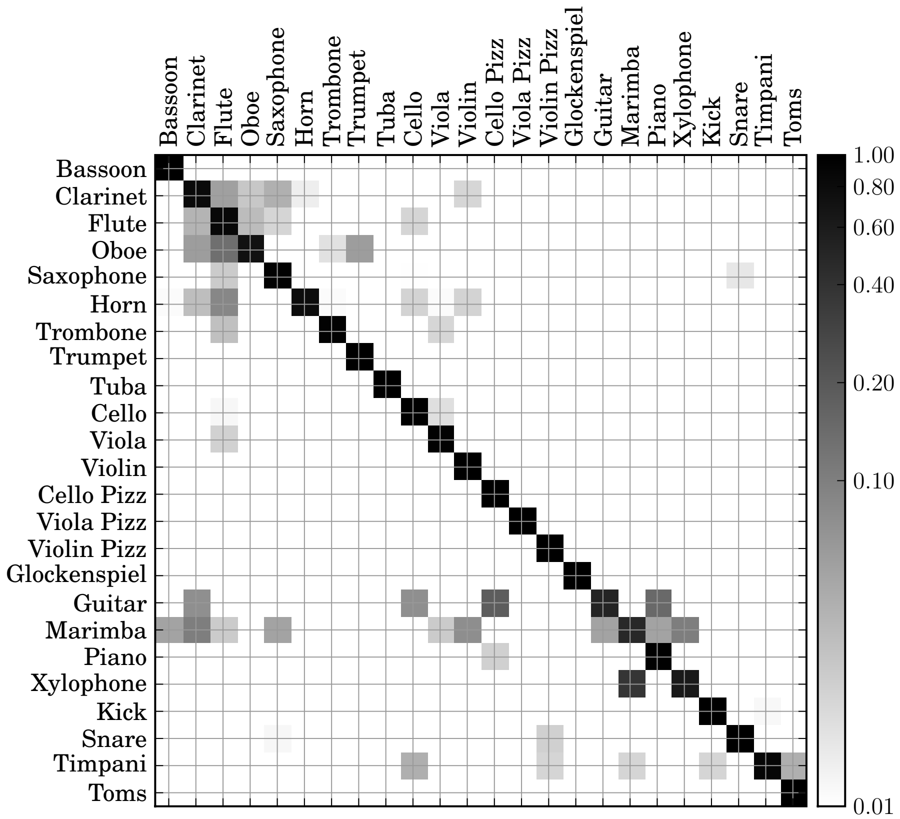


Figure 3.9: Experiment A1: Accuracy of spectral classification of isolated sounds using ten-fold cross validation. Row labels: True class. Column labels: Estimated class. Average accuracy: 88.2%.

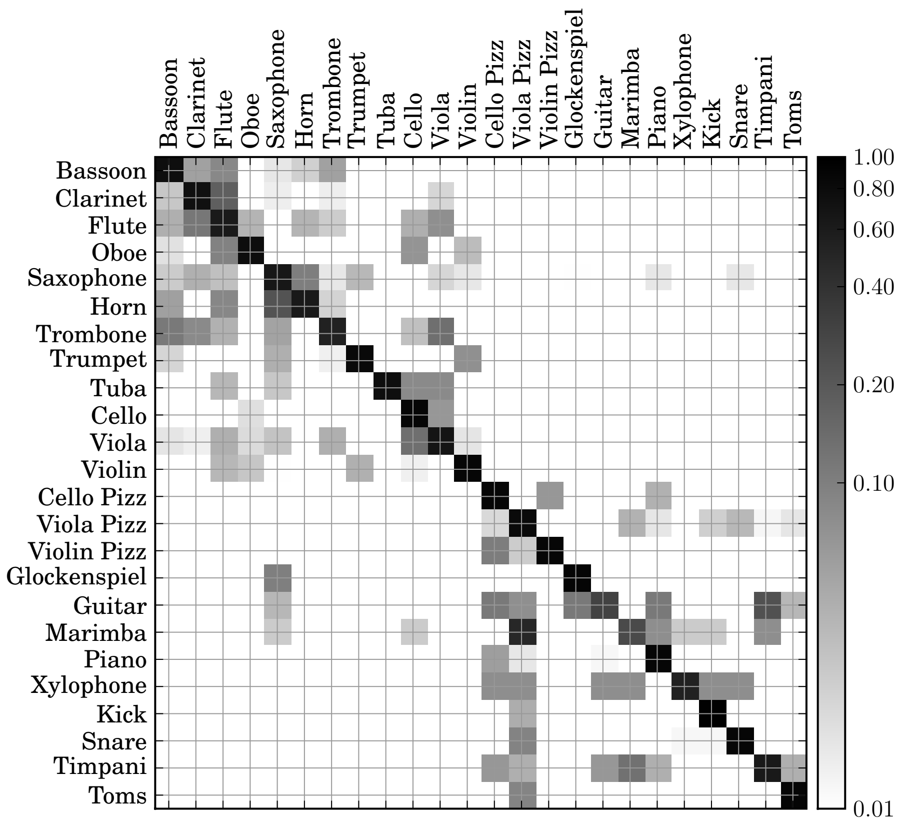


Figure 3.10: Experiment A2: Accuracy of temporal classification ($p = \infty$) of isolated sounds using ten-fold cross validation. Row labels: True class. Column labels: Estimated class. Average accuracy: 72.9%.

instrument families, e.g., cello and viola, bassoon and clarinet, and guitar and piano.

Experiment A3 evaluates the classification performance when combining spectral and temporal features. The feature vectors, $\mathbf{z}_S$ and $\mathbf{z}_T$, extracted from each spectral-temporal atom pair during Experiments A1 and A2, are concatenated to form 3907 64-dimensional feature vectors as described in Section 3.4. Again, the experiment is performed once for $p = 2$ and again for $p = \infty$. Table 3.2 shows the accuracy rates, and Fig. 3.11 illustrates the confusion matrix when $p = \infty$. The total accuracy rate is 92.1% when $p = 2$ and **92.3%** when $p = \infty$. Temporal information improves classification accuracy for 18 of the 24 instrument classes along with the

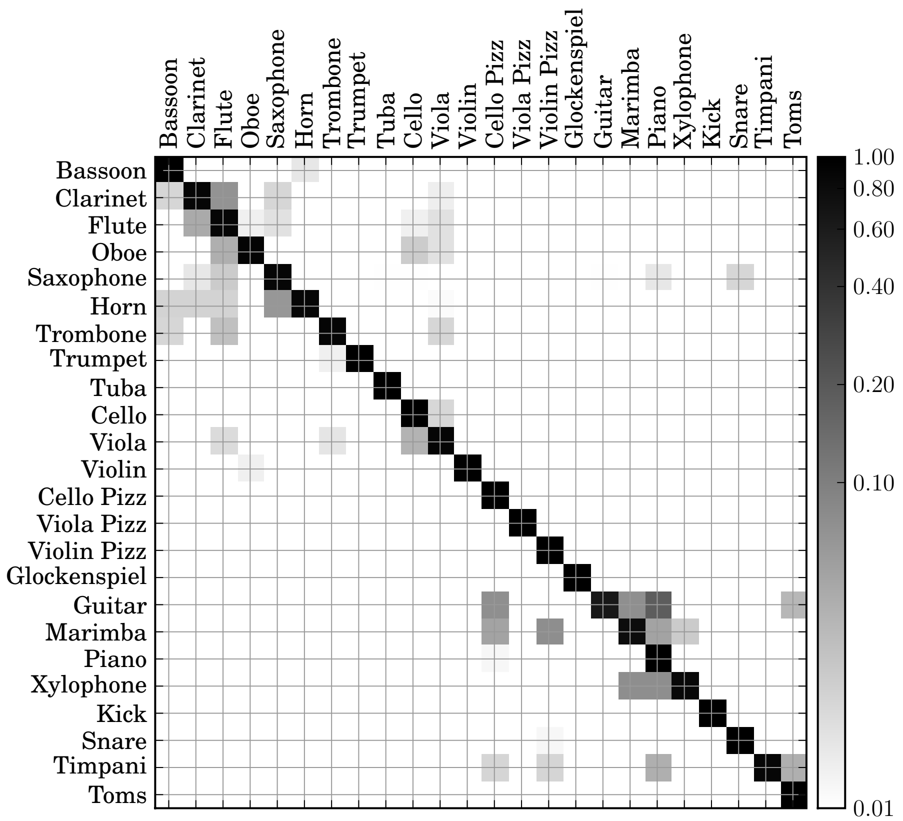


Figure 3.11: Experiment A3: Accuracy of spectral-temporal classification ($p = \infty$) of isolated sounds using ten-fold cross validation. Row labels: True class. Column labels: Estimated class. Average accuracy: **92.3%**.

overall accuracy. Accuracy improves most for the string pizzicato, percussion, brass, and certain woodwind instruments. The remaining misclassifications occur mostly within families, e.g., clarinet and flute, and guitar and piano. For isolated sounds, this experiment verifies the hypothesis that a combination of spectral and temporal information can improve instrument recognition accuracy over methods that use information from one domain alone.

At 92.3% among 24 classes, our method achieves state-of-the-art performance for isolated instrument recognition. Eronen [31] achieved an average accuracy of 35% among 29 classes from a mixed data set that included the McGill University

Master Samples [69], University of Iowa Database [68], IRCAM Studio Online, and private recordings of acoustic guitar and a Roland XP-30 synthesizer. The classifier was k-nearest neighbor classifier and features included MFCCs, warped LPCs, fundamental frequency, attack time, amplitude envelope, and spectral centroid. Kitahara et. al. [32] achieved 79.7% accuracy among 19 classes from standard MIDI files using Bayesian classification after PCA and a variety of spectral, temporal, and modulation features. Kostek et. al. [34] achieved 71.3% accuracy among 12 classes from a private data set by using neural networks, genetic algorithms, wavelet energy bands, and MPEG-7 descriptors. Chétry et. al. [35] achieved 83.2% among 10 classes from the University of Iowa Database and the RWC Database [75] by using a K-means codebook and line spectrum frequencies.

3.5.2 Feature Vector Classification of Solo Melodic Phrases

Next, we evaluate the proposed method's ability to recognize instruments in solo phrases. These experiments use data collected from over forty hours of solo melodic phrases from eleven instruments performing well-known musical repertoire such as concertos, sonatas, and orchestral excerpts. Composers include Mozart, Beethoven, Richard Strauss, Ferdinand David, Haydn, Vaughan Williams, Brahms, and Tchaikovsky. The phrases were performed by amateur musicians from across the United States and were recorded by staff from the School of Music at the University of Maryland. Each of the signals in this data set contains a musical excerpt performed by a single instrument. Signal duration varies approximately between 10 and 30 seconds. This database contains the following eleven instrument classes: bassoon, cello, clarinet, flute, french horn, oboe, trombone, trumpet, tuba, viola, and violin.

We make a brief remark about data. Availability of public data sets is a widespread problem in MIR, particularly due to copyright restrictions that prevent some excellent databases from being shared. One consequence is the difficulty in making quantitative comparisons among experiments. As MIR research evolves, old public data sets may not fulfill the evolving needs of researchers. Then, researchers are forced to use private data sets, e.g. musical instrument recognition [37, 10]. As a result, one cannot make an indisputable comparison across works. Synthetic data such as MIDI files provide standardization, but they cannot capture the acoustic traits of natural recordings. We decided to use our aforementioned data set because it is more comprehensive, more standardized, and more appropriate than any currently available data set of solo instrumental melodic signals. Therefore, the results presented here only suggest that our method is within the same realm as the state of the art and perhaps better in some scenarios, but not universally better.

For this experiment set, each signal is decomposed using NMF with an inner dimension of $K = 12$. For this work, we have decided to keep this parameter constant over all tested signals. Although such a decomposition may not be sufficient for tasks such as transcription and source separation, the atoms are decomposed well enough for a timbre-specific task such as instrument recognition. There exist methods in the literature that adapt the decomposition rank, K, to the input signal [72, 73]. While such methods may improve the following results, they remain beyond the scope of this dissertation.

In this subsection, we evaluate the performance of classifying each of the feature vectors through ten-fold cross validation, i.e., one classification is made per feature vector. (In the following subsection, we merge these classifications to make one classification per signal.) As a control, Experiment B1 evaluates the classification ability of spectral features using MFCCs. From each of the K spectral atoms, one feature vector $\mathbf{z}_S$ is extracted using MFCCs. The parameters are the same as those in Experiment A1: 32 MFCCs with center frequencies between 110 Hz and 3951 Hz. Fig. 3.12 illustrates the confusion matrix for Experiment B1, and Table 3.3 shows the accuracy rates for each class. The average of the 11 accuracy rates is 72.4%. There is some misclassification within instrument families, e.g., cello/viola, and horn/trombone/trumpet, but the other errors are scattered among all classes.

Experiment B2 evaluates the classification ability of temporal features. Like Experiment A2, the proposed feature extraction algorithm uses the parameters shown in Table 3.1. One feature vector $\mathbf{z}_T$ is computed for each temporal NMF atom. Here, we only perform the experiment for $p = \infty$. Table 3.3 shows the accuracy rates for each class. The average accuracy rate is 31.2%. Fig. 3.13 illustrates the confusion matrix for Experiment B2. Perhaps as expected, the results show that temporal features alone do not classify instruments as well as spectral features, and this fact is emphasized in melodic phrases where adjacent notes have the ability to overlap in time and frequency.

Experiment B3 evaluates the classification performance when concatenating spectral and temporal features. Table 3.3 shows the accuracy rates, and Fig. 3.14 illustrates the confusion matrix. The total accuracy rate is 71.1%. Overall, the

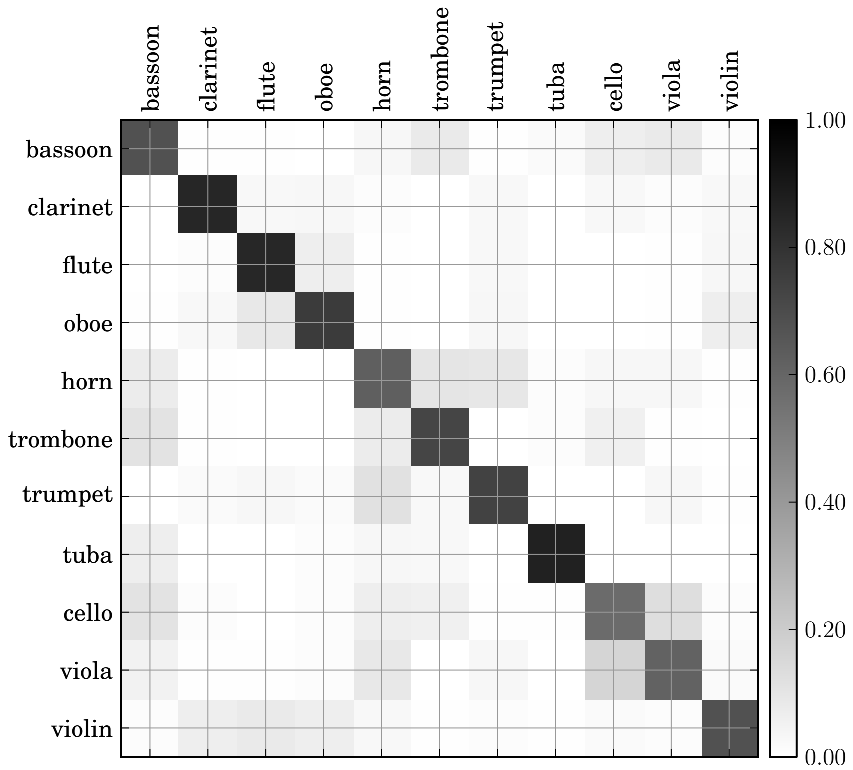


Figure 3.12: Experiment B1: spectral features. Classification accuracy of solo excerpts using ten-fold cross validation. Row labels: True class. Column labels: Estimated class. Average accuracy: 72.4%.

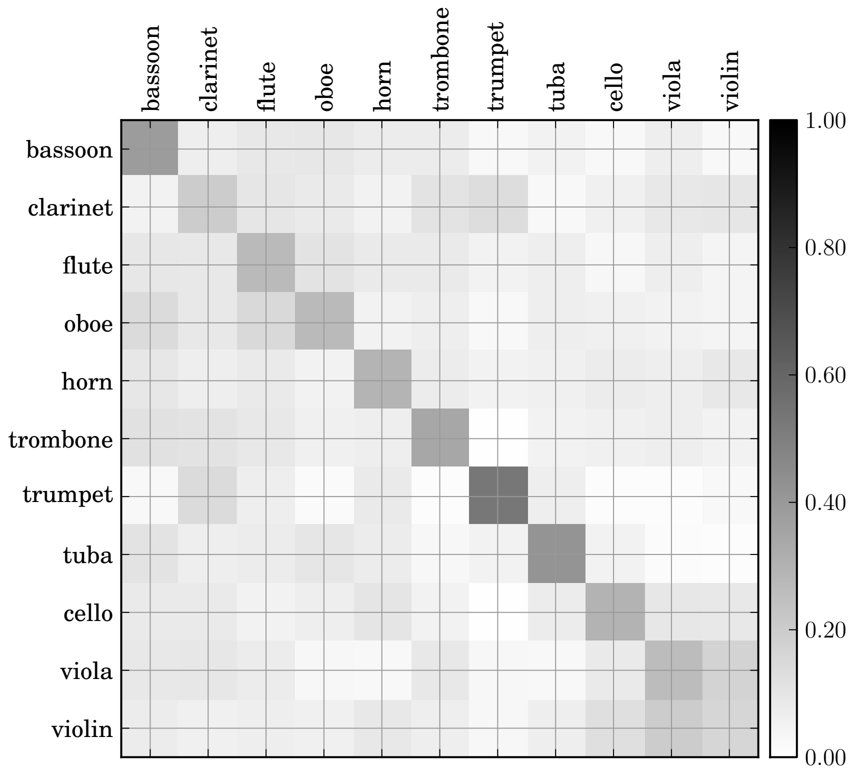


Figure 3.13: Experiment B2: temporal features, $p = \infty$. Classification accuracy of solo excerpts using ten-fold cross validation. Row labels: True class. Column labels: Estimated class. Average accuracy: 31.2%.

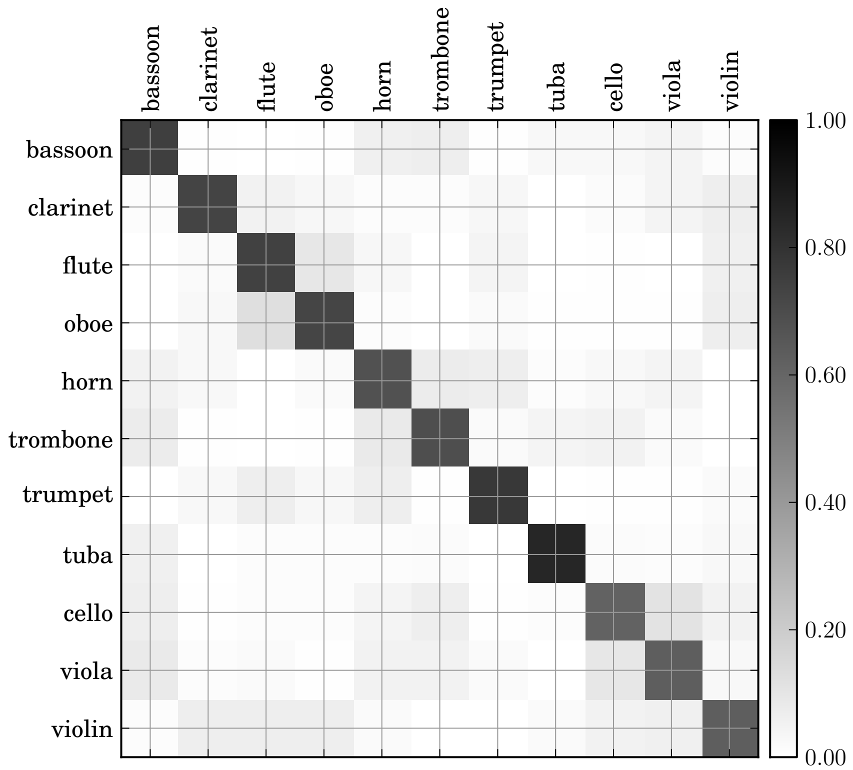


Figure 3.14: Experiment B3: spectral-temporal features, $p = \infty$. Classification accuracy of solo excerpts using ten-fold cross validation. Row labels: True class. Column labels: Estimated class. Average accuracy: 71.1%.

Instrument	#	B1	B2	B3
Bassoon	144	68.1	38.9	75.0
Clarinet	120	85.0	20.0	73.3
Flute	144	84.7	27.1	74.3
Oboe	144	76.4	27.1	72.9
Horn	144	62.5	29.2	68.1
Trombone	144	72.9	34.0	69.4
Trumpet	144	73.6	52.8	77.1
Tuba	120	86.7	41.7	85.0
Cello	144	57.6	29.9	61.1
Viola	144	61.1	26.4	63.2
Violin	144	68.1	16.0	63.2
Total	1536	72.4	31.2	71.1

Table 3.3: Experiments B1, B2, B3: sample sizes and accuracy rates.

results are similar to Experiment B1. Misclassifications are scattered widely among all classes.

Finally, we repeat the same three experiments by classifying each feature vector by instrument family instead of individual instrument. For this work, we define the *wind* family to include bassoon, clarinet, flute, and oboe. The *brass* family includes french horn, trombone, trumpet, and tuba. The *strings* include cello, viola, and violin. Each feature vector obtained during Experiment Set B is relabeled by family instead of individual instrument. Otherwise, the classification procedure remains the same. In Table 3.4, we show the results of these classifications. Spectral classification yields an accuracy of 78.2%, temporal classification yields 57.7%, and spectral-temporal classification yields 79.4%. The string family has the worst classification but also appears to benefit the most from the introduction of temporal information.

Instrument	#	B1	B2	B3
Winds	552	82.2	59.4	83.2
Brass	552	83.5	56.9	82.2
Strings	432	69.0	56.7	72.9
Total	1536	78.2	57.7	79.4

Table 3.4: Experiments B1, B2, B3 with family classifications: sample sizes and accuracy rates.

3.5.3 Signal Classification of Solo Melodic Phrases

In the previous subsection, one classification was made for each feature vector, $\mathbf{z}$. Here, we merge classifications of multiple feature vectors to arrive at one final classification for each signal. We accomplish this task through majority voting. Each input signal is decomposed into K spectral atoms and K temporal atoms, and each of these atoms are classified as shown in Experiment Set B. Then, among the K spectral (or temporal) atoms, a majority vote is taken from the K classifications to provide one classification for the entire signal. The data is the same as that used in Experiment Set B. Instead of ten-fold cross validation, we now use leave-one-out cross validation, i.e., for each SVM instance, the test vectors include the K feature vectors from one signal, and the remaining feature vectors are used for training. This process is repeated for each signal.

In Experiment C1, we perform majority voting over the MFCC classifications obtained from the spectral atoms of each signal. In Experiment C2, we perform majority voting over the MGFR classifications obtained from the temporal atoms of each signal. Finally, in Experiment C3, we perform majority voting over the classifications from the spectral-temporal feature vectors obtained in Experiment

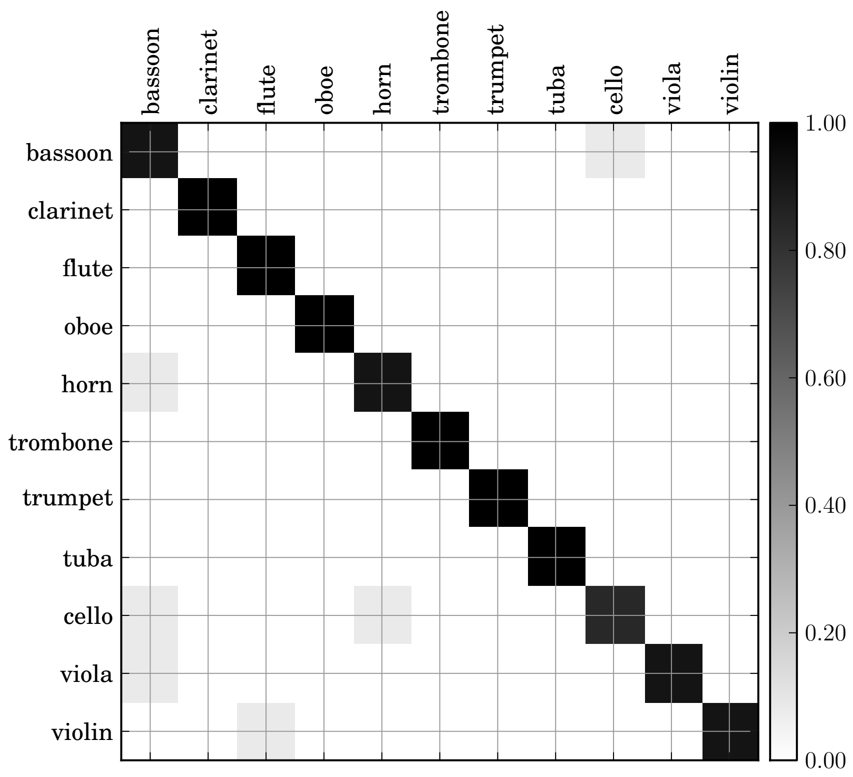


Figure 3.15: Experiment C1: spectral features. Classification accuracy of solo excerpts using leave-one-out cross validation. Row labels: True class. Column labels: Estimated class. Average accuracy: 95.5%.

Instrument	#	C1	C2	C3
Bassoon	12	91.7	91.7	100.0
Clarinet	10	100.0	70.0	100.0
Flute	12	100.0	41.7	100.0
Oboe	12	100.0	50.0	91.7
Horn	12	91.7	41.7	100.0
Trombone	12	100.0	50.0	100.0
Trumpet	12	100.0	83.3	100.0
Tuba	10	100.0	50.0	100.0
Cello	12	83.3	41.7	83.3
Viola	12	91.7	33.3	91.7
Violin	12	91.7	8.3	91.7
Total	128	95.5	51.1	**96.2**

Table 3.5: Experiments C1, C2, C3: One decision per song. Sample sizes and accuracy rates.

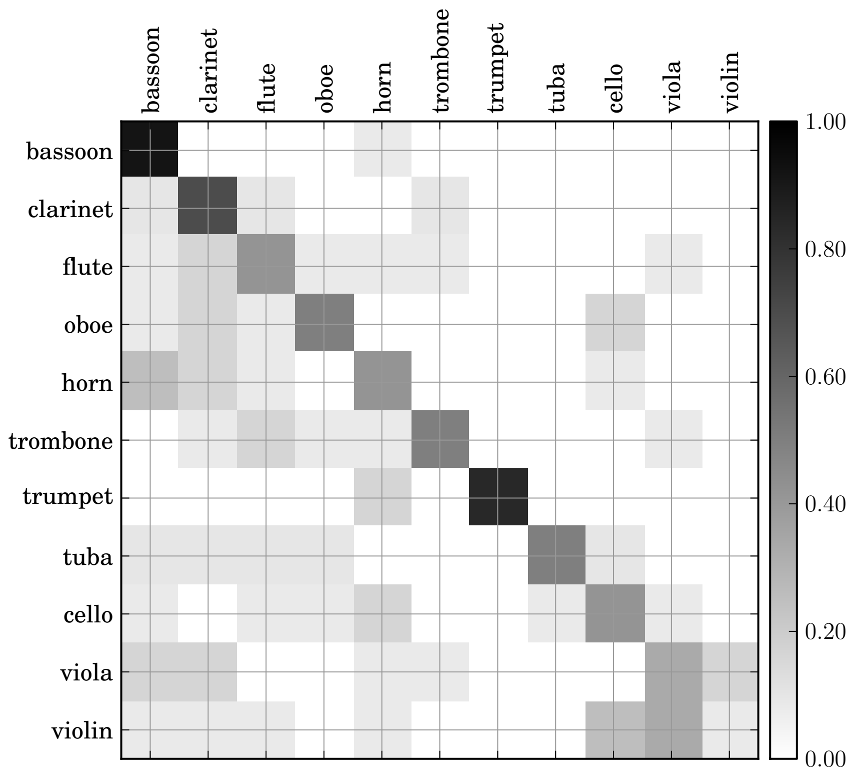


Figure 3.16: Experiment C2: temporal features, $p = \infty$. Classification accuracy of solo excerpts using leave-one-out cross validation. Row labels: True class. Column labels: Estimated class. Average accuracy: 51.1%.

Instrument	#	C1	C2	C3
Winds	46	97.8	84.8	100.0
Brass	46	97.8	73.9	97.8
Strings	36	83.3	77.8	94.4
Total	128	93.0	78.8	**97.4**

Table 3.6: Experiments C1, C2, C3 with family classifications. One decision per song. Sample sizes and accuracy rates.

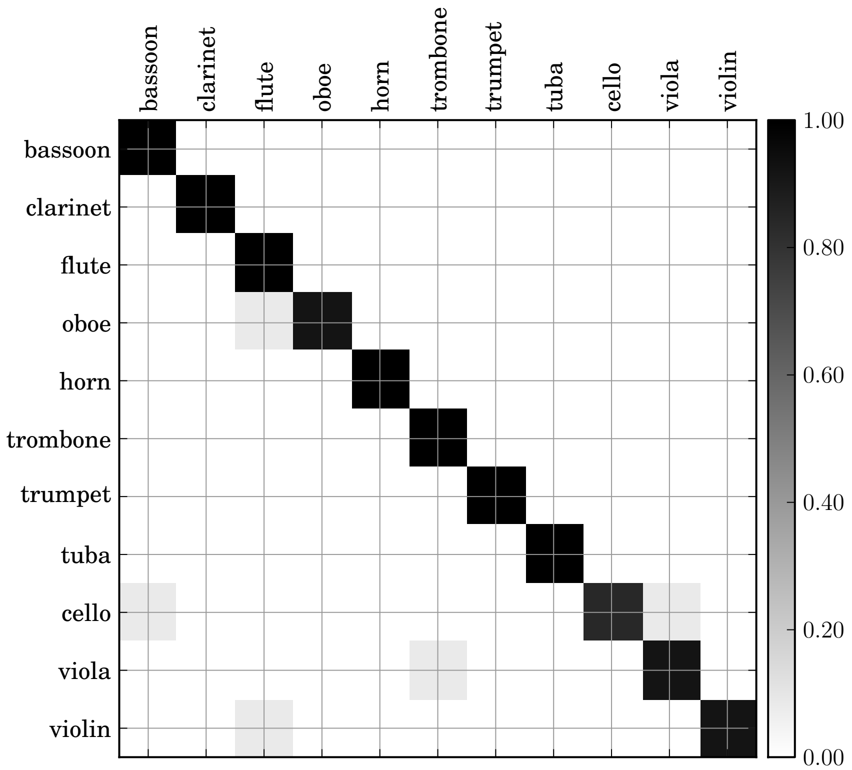


Figure 3.17: Experiment C3: spectral-temporal features, $p = \infty$. Classification accuracy of solo excerpts using leave-one-out cross validation. Row labels: True class. Column labels: Estimated class. Average accuracy: **96.2%**.

B3. The results are summarized in Table 3.5 which shows that the best accuracy, **96.2%**, is achieved by the use of spectral-temporal features. Confusion matrices are shown in Figs. 3.15, 3.16, and 3.17.

The accuracy rates achieved by our system reflect state-of-the-art performance for instrument recognition in solo melodic excerpts among as many as 11 classes. Vincent and Rodet [36] achieve 90% accuracy among five classes from a data set containing ten commercial CDs by using nonlinear independent subspace analysis and a GMM classifier. Chétry et. al [35] achieves 78% accuracy among six classes from a set of ten different data sources by using line spectrum frequencies and a SVM. Essid et. al. [37] achieve 93% accuracy among ten classes from data collected from commercial CDs by using features such as octave band intensities, temporal, spectral, cepstral, and modulation features, and a SVM. Joder et. al [10] achieve 84.7% accuracy among eight classes from commercial CDs and the RWC database by using temporal, spectral, cepstral, and wavelet features, and a SVM.

We repeat these experiments by using family classifications mentioned earlier. Majority voting is now performed upon the classifications of each feature vector by instrument family instead of individual instrument. The results are summarized in Table 3.6. The combination of spectral and temporal information results in a family classification accuracy of **97.4%**, an improvement over classification performend with either spectral or temporal information alone.

3.6 Discussion

One characteristic of this algorithm is the projection of a temporal atom down to a single feature vector, i.e., the conversion of a function $s(t)$ to the vector $\mathbf{z}_T \in \mathbb{R}^{32}$. This projection is *shift-invariant* – regardless of where the musical event is located in the temporal atom, the projection will produce the same feature vector. In other words, if $\mathbf{z}_T$ is extracted from atom $s(t)$, then $\mathbf{z}_T$ will also be extracted from atom $s(t - \tau)$ for any τ. Other methods of dimensionality reduction may not be shift-invariant. For example, linear discriminant analysis (LDA) can reduce the dimensionality of a set of vectors such that projected vectors from the same class (e.g., musical instrument) are near each other while projected vectors from different classes are far apart. However, in its unmodified form, LDA would have trouble ignoring shifts in the temporal atom, i.e., LDA will detect *when* a musical event occurs rather than which instrument it represents. The feature selection process should discard any idea of temporal locality – when a musical note begins and ends – and instead focus on the atom's shape. This issue reinforces the need for a shift-invariant projection method such as ours.

That being said, it could be possible to learn different kernels, $\{g_1, g_2, ..., g_I\}$, with which to convolve the temporal atoms. For rigor and simplicity, we have chosen to use gamma kernels because their spectral and temporal properties are easy to deduce and their use is well established in the signal processing community. Works in other areas have attempted to learn kernels for other tasks, e.g., source separation or genre classification, by performing sparse coding and dictionary learning in the time domain [76, 77, 78, 79]. Given a signal $x(t)$, these methods attempt to find temporal dictionary atoms $s_i(t)$ and their corresponding coefficients $a_{i,j}$ and delays $\tau_{i,j}$ that solve the following minimization:

$$\min_{a,s,\tau} \int \left| x(t) - \sum_{i=1}^{I} \sum_{j=1}^{J_i} a_{i,j} s_i(t - \tau_{i,j}) \right|^2 dt \ . \tag{3.18}$$

In doing so, these methods learn a dictionary of temporal atoms from which the input signal can be approximated along with their coefficients. This time-relative collection of data can then be used for classification. For example, Manzagol et al. use the coefficients $a_{i,j}$ in (3.18) as features for genre classification [76]; they reported similar results using learned kernels versus using a fixed set of gammatone kernels. In fact, the gammatone kernels performed slightly better for an encoding task. Other works have used this procedure to solve coding problems rather than classification problems. For example, Smith and Lewicki evaluate the coding efficiency of such a sparse code upon vocal and environmental sounds [77, 78]. In any event, each of these formulations are usually used to decompose a signal into temporal atoms of a fine scale, i.e., less than one millisecond. In this work, we must decompose a signal into atoms of a coarser scale, i.e., several hundreds of milliseconds. This requirement led to our choice of parameters in Table 3.1.

Finally, we compare our work with those works by Hermansky et al. [51, 52] which employ a similar temporal feature extraction method. Their method is similar to ours in that it captures temporal evolution over coarse windows. However, such temporal evolution is isolated within a single frequency band. Because we employ

NMF, we can capture the temporal evolution over an entire range of frequencies, weighted appropriately by the NMF algorithm. In other words, NMF can automatically tell us the most relevant frequency bands in which to observe temporal evolution. No manual selection of frequency bands is required. Like these works, our results have also shown that augmenting spectral features with temporal features noticeably improves performance over either feature set indivudally.

3.7 Summary

We have shown a method for extracting information from NMF atoms at multiple resolutions for the purpose of musical instrument recognition. Inspired by the early cortical stage of the human auditory system, this method performs multiresolution analysis upon NMF atoms through the use of MFCCs and a *multiresolution gamma filterbank*. The filters that compose this filterbank are capable of describing any combination of attack time and decay rate. Although there are other works that explore the combination of spectral and temporal information for instrument recognition, to our knowledge, this work is the first to parameterize the profiles of temporal NMF atoms through multiresolution analysis.

Our original hypothesis was that the combination of spectral and temporal information extracted from NMF atoms would improve instrument recognition over systems that use spectral information alone. The results support this hypothesis, but the conclusion is stronger for isolated sounds than it is for solo melodic phrases where the improvement in performance is less significant.

While the experimental results are encouraging, there is still room for improvement. First, the data used in this chapter comes entirely from monophonic music. Obviously, an important direction for future research is to test this method on polyphonic music. Although we expect the task to become more difficult, we believe that the primary obstacle to successful classification is NMF. Ideally, if NMF can learn perfect dictionary atoms, then the polyphonic nature of music is not an obstacle. Unfortunately, NMF is not perfect. If NMF cannot accurately learn dictionary atoms, then any following feature extraction may fail. However, NMF and its variants are being thoroughly investigated by the community, including those algorithms that impose additional constraints such as harmonicity [40, 39, 80, 41, 81, 42], co-occurrence [82, 18], sparsity [22], and smoothness [22, 41].

Second, fusing spectral and temporal information in another manner may improve performance. In Experiments A3, B3, and C3, we fused the spectral and temporal feature vectors by simply concatenating corresponding atoms. However, there are other methods of fusing SVMs from spectral and temporal data, e.g., committees or boosting.

Finally, modification to the classification may improve performance. The results shown earlier illustrate that classification by instrument family was more accurate than classification by individual instrument. Therefore, a hierarchical SVM may improve performance, where the feature vectors are first classified by instrument family and then by individual instrument. However, this approach also introduces system complexity as well as the potential for error propagation.

Chapter 4

Approximate Matching Pursuit

Music transcription is an important and well-studied task in music information retrieval (MIR). The ability to convert musical signals into a labeled set of discrete musical events, if performed successfully, would substantially impact all areas of MIR, including source separation, segmentation, genre classification, human-computer interfaces, and more. However, transcription is not easy. Because most musical signals of interest are polyphonic, sounds from separate sources (e.g., voices or instruments) often overlap in time and frequency, making it more difficult for algorithms to decompose.

Over the past decade, the emergence of constrained factorization algorithms such as sparse coding and nonnegative matrix factorization (NMF) have revolutionized the way we perform music transcription. Sparse coding attempts to represent an input signal as a sparse linear combination of atoms from a large, overcomplete dictionary. NMF attempts to represent an input signal in matrix form as a product of two low-rank nonnegative matrices. Both of these methods provide an automatic way to decompose polyphonic music into individual musical events such as notes and beats.

Both sparse coding and NMF are closely related to dictionary learning, where the input signal is used to learn a concise dictionary of musical atoms that collectively represent the input signal. Many researchers have reported success when decomposing simple musical signals using NMF [14] or methods based upon sparse coding such as K-SVD [8, 7]. Unfortunately, problems remain for intricate, polyphonic musical signals. When musical notes overlap in time and frequency, the separation and transcription performance of these basic dictionary learning methods diminishes rapidly. In such a case, the algorithm will usually learn a dictionary where each individual atom contains information from multiple musical sources, thus hindering our attempts at decomposition.

Researchers have slowly improved upon the original dictionary learning methods by adding constraints to the learning process. By restricting the dictionary atoms to reside within a predetermined feasible set, we can ensure that the learned atoms will be useful at the conclusion of the learning process. For example, existing solutions include adding constraints to the dictionary learning process such as harmonicity [42, 41] or smoothness [21, 41].

Another solution to the problem of overlapping sounds is to add structure to the dictionary. For example, one can construct and use a large, predefined, overcomplete dictionary where each atom is already labeled and assumed to contain information from only one musical source. Instead of learning an optimal dictionary for a given musical signal, it may suffice to match the signal to this large set of precomputed, labeled dictionary atoms. Then, by decomposing a signal with respect to this fixed dictionary, classification is easily achieved by simply reading the label of the atom. As musical databases become more available, construction of predefined dictionaries will become easier, thus reducing the need for adaptive dictionary learning. Many of the decomposition methods that involve dictionary learning such as NMF can easily be modified to eliminate the learning step, leaving only the task of computing how much each dictionary atom contributes to the input signal, i.e., the atoms' coefficients.

Of course, the performance of such an algorithm depends upon the breadth of the dictionary. When atoms from more musical sources are added to the dictionary, the dictionary's ability to decompose polyphonic music will improve. However, dictionary growth introduces concerns related to scalability and computational complexity. While the aforementioned algorithms have significantly advanced the state of the art, they remain slow and difficult to scale as the dictionary size increases. Most of the original sparse coding methods such as matching pursuit (MP) [1] and NMF with multiplicative updates [5, 6] have complexity that is linear in the size of the dictionary. As a result, when dictionary sizes grow, the transcription efficiency of these algorithms diminishes.

To summarize the problem: how can we make use of a large, precomputed, overcomplete dictionary to accurately perform music transcription in a scalable and computationally feasible manner?

In this chapter, we address this problem by proposing a variant of MP called *approximate matching pursuit* (AMP). Unlike MP and NMF, AMP can decompose signals into a sparse combination of atoms with complexity that is *sublinear* in the dictionary size while maintaining accuracy. To do this, AMP uses an approximate nearest neighbor (ANN) method to find approximate matches to the signal residual at each iteration. The ANN method that we choose in this work is locality sensitive hashing (LSH), a probabilistic hash algorithm that places similar, yet not identical, observations into the same bin. LSH can retrieve near neighbors with a complexity that is sublinear in the dictionary size. Not only is LSH fast, but it is also scalable – as the dictionary grows, reorganizations of the data structure are unnecessary. We simply add the new dictionary atom into its respective bin in the hash table.

Our experiments demonstrate that AMP is as accurate and robust as MP variants such as OMP [2] and STOMP [3] under a wide variety of scenarios related to sparsity, dimensionality, and additive noise. At the same time, AMP requires less computation than OMP and STOMP. We show that AMP requires fewer inner products to reach convergence than the other algorithms. Finally, we show the usefulness of AMP for music transcription by decomposing musical signals as combinations of atoms from a large dictionary of over 170,000 labeled musical spectra.

We summarize our contributions as follows:

1. We propose AMP and illustrate that it is as accurate and robust as other pursuit methods while requiring fewer computations;
2. we illustrate the usefulness of AMP for music transcription when a large dictionary of musical spectra is provided.

First, in Section 4.1, we discuss existing work related to MP for music transcription. In Section 4.2, we formulate the problem and provide the system model and assumptions. In Section 4.3, we propose the AMP algorithm and compare it to other pursuit methods. In Section 4.4, we discuss LSH, the ANN method that we choose to use inside AMP for this work. In Section 4.5, we provide many experiments that measure the accuracy and robustness of AMP and other pursuit methods. In Section 4.6, we describe how to construct a large dictionary of musical atoms, and we show how AMP can use this dictionary to transcribe polyphonic musical signals. We conclude in Section 4.7.

4.1 Related Work

Mallat and Zhang first proposed the matching pursuit (MP) algorithm in 1993 [1]. This greedy algorithm directly addresses the issue of sparsity by decomposing a signal, $\mathbf{x}$, into a linear expansion of waveforms that are selected from a redundant dictionary of functions. When stopped after a few iterations, this algorithm yields a signal approximation using only a few atoms. After each iteration of the MP algorithm, the residual, $\mathbf{r}$, is orthogonal to the previously selected vector, $\mathbf{a}_k$, but not necessarily orthogonal to the dictionary vectors selected earlier.

In response, Pati et al. proposed an improvement called orthogonal matching pursuit (OMP) which ensures that the residual is orthogonal to all previously selected dictionary vectors [2]. The OMP algorithm is shown in Appendix A. After dictionary atoms are selected for inclusion into the decomposition, an extra orthogonalization step is performed by solving a least-squares problem. Researchers have shown that OMP provides a dramatic improvement over MP [2]. In many cases, when an input signal is known to be k-sparse, OMP converges in k iterations, while MP will require many more iterations to converge.

Later, Chen and Donoho introduced basis pursuit (BP) [83, 84, 85], an optimization principle for finding sparse coefficients by solving the following linear program:

$$\min_{\mathbf{s}} ||\mathbf{s}||_1 \text{ such that } \mathbf{x} = \mathbf{A}\mathbf{s}. \tag{4.1}$$

Minimizing the L_1 norm of the vector $\mathbf{s}$ induces its sparsity. Any algorithm from the linear programming literature, e.g. simplex method or interior-point method, can be used to solve this problem. However, BP does introduce issues related to the problem size, parameter settings, and sparsity of the signal with respect to the dictionary [85]. For example, in [85], it is assumed that "fast implicit algorithms" exist for computing $\mathbf{A}\mathbf{s}$ or $\mathbf{A}^T\mathbf{x}$ which is possible when the dictionary represents a family of atoms derived from a kernel, e.g., Fourier or Wavelet bases. When such fast algorithms do not exist, BP can perform poorly in practice [3]. Our own experiments affirm this claim, and therefore we do not consider BP further in this dissertation.

Several variants of matching pursuit have since been proposed. One variant is Stagewise Orthogonal Matching Pursuit (STOMP) [3], shown in Appendix A. In OMP, the dictionary atom that is nearest to the residual is added to the active set of dictionary atoms at each iteration. In STOMP, all dictionary atoms that are sufficiently near the residual are added to the active set of dictionary atoms. For some threshold, $\tau_{\mathbf{r}}$, that depends on the residual, $\mathbf{r}$, any index k that satisfies $\mathbf{a}_k^T\mathbf{r} > \tau_{\mathbf{r}}$ is added to the active set at each iteration. As a result, STOMP runs faster than OMP and BP for large-scale problems [3].

MP algorithms have been applied to MIR in many ways. The most popular applications are music transcription and source separation. Harmonic matching pursuit (HMP) has been used to decompose an audio signal into Gabor or harmonic (i.e., sums of Gabor) atoms [86]. Dictionaries of atoms can also be adapted and learned to fit the data [79]. To resolve instances when harmonics from separate notes overlap, some algorithms impose smoothness constraints [81, 80]. Similar sparse coding methods have been used for genre recognition [76]. In the neurological signal processing literature, pursuit methods for generic acoustic signals have been applied for coding purposes [77, 78].

4.2 Problem Formulation

Given the magnitude spectrum of an input signal, $\mathbf{x} \in \mathbb{R}^M$, and a dictionary, $\mathbf{A} = [\mathbf{a}_1\ \mathbf{a}_2\ \ldots\ \mathbf{a}_K] \in \mathbb{R}^{M \times K}$, the problem is to find a vector of coefficients $\mathbf{s} \in \mathbb{R}^K$ that minimizes $||\mathbf{x} - \mathbf{A}\mathbf{s}||_2$.

When $M < K$, the dictionary is called *overcomplete*, and there are infinitely many solutions for $\mathbf{s}$. However, by imposing a sparsity constraint on $\mathbf{s}$, the solution space diminishes greatly, possibly to a unique solution. In particular, if the input is truly a sparse linear combination of dictionary atoms, i.e., $\mathbf{x} = \mathbf{A}\mathbf{s}_0$, where $\mathbf{s}_0$ is a sparse vector, then the problem becomes finding an optimal set of coefficients, $\hat{\mathbf{s}} = \operatorname{argmin}_{\mathbf{s}} ||\mathbf{x} - \mathbf{A}\mathbf{s}||_2$, that is equal to the input coefficients, i.e., $\hat{\mathbf{s}} = \mathbf{s}_0$.

An exhaustive search for the sparsest solution is NP-hard [87]. However, sub-optimal greedy algorithms such as OMP often work well in practice. Unfortunately, OMP requires at least K inner products to computed during each iteration, thus creating a complexity that is at least linear in K. Because this complexity is too slow for large dictionaries, the problem becomes solving for $\hat{\mathbf{s}} = \mathbf{s}_0$ using an algorithm that has complexity that is sublinear in the dictionary size, K.

In reality, for any dictionary $\mathbf{A}$, it is unlikely that an input musical signal is an exact sparse combination of dictionary atoms. We can model this discrepancy using a noise term, $\mathbf{x} = \mathbf{A}\mathbf{s}_0 + \mathbf{n}$. Therefore, we want to solve the sparse solution problem under additive noise, as well.

Throughout this work, we assume that the dictionary, $\mathbf{A}$, does contain atoms that accurately represent the spectrum of the input signal, $\mathbf{x}$. For example, if we choose to analyze the signal generated by a guitar, then the dictionary should contain spectra from a guitar. Otherwise, decomposition may fail. Also, we assume that the dictionary is overcomplete, $M < K$, although this assumption is not strictly necessary for AMP to operate. We also assume that the true sparsity of any input signal, $||\mathbf{s}_0||_0$, is less than the dimensionality, M. For musical signals, this assumption usually holds in practice. For example, even in highly polyphonic music, the number of simultaneous sounds will likely be less than the dimensionality of our spectra, i.e., the number of frequency bins. If not, then we increase the FFT size to produce longer spectra.

4.3 Proposed Algorithm

One drawback of the existing pursuit methods mentioned earlier is their complexity. When the dictionary size, K, becomes very large (e.g., over one million), these methods may require an unacceptably large amount of computation to find an answer. For example, in each iteration of MP, K inner products must be computed between the residual $\mathbf{r}$ and every atom in the dictionary – a complexity of order $O(MK)$.

Here, we introduce a simple variation of these pursuit methods that uses an ANN algorithm in place of computing K inner products as done in MP. As a result, we can reduce the complexity to be sublinear in K.

The approximate matching pursuit (AMP) algorithm is described in Algorithm 1. This algorithm is similar to OMP except that it addresses the main computational bottleneck for large dictionaries – nearest neighbor search – by allowing any adequately near neighbor to be selected as a component.

Algorithm 1 Approximate Matching Pursuit [Tjoa and Liu]

```
Input: $\mathbf{x} \in \mathbb{R}^M$; $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_K] \in \mathbb{R}^{M \times K}$ s.t. $||\mathbf{a}_k||_2 = 1$ for all $k$.
Output: $\hat{\mathbf{s}} \in \mathbb{R}^K$
Initialize: $\mathcal{S} \leftarrow \emptyset$; $\mathbf{s} \leftarrow \mathbf{0}$; $\mathbf{r} \leftarrow \mathbf{x}$; $\epsilon > 0$.
while $||\mathbf{r}|| > \epsilon$ do
    Find any $k$ such that $\mathbf{a}_k$ and $\mathbf{r}$ are near neighbors.
    $\mathcal{S} \leftarrow \mathcal{S} \cup k$
    Solve for $\{s_j | j \in \mathcal{S}\}$: $\min_{s_j | j \in \mathcal{S}} ||\mathbf{x} - \sum_{j \in \mathcal{S}} \mathbf{a}_j s_j||$
    $\mathbf{r} \leftarrow \mathbf{x} - \mathbf{A}\mathbf{s}$
$\hat{\mathbf{s}} \leftarrow \mathbf{s}$
```

We make the following remarks:

1. AMP resembles MP and OMP. This modification is intentionally kept simple. Like OMP, AMP is capable of providing a sparse decomposition in far fewer

iterations than MP. If the ANN retrieval method were instead changed to a nearest-neighbor (NN) method, then AMP would yield identical results to OMP.

2. AMP is flexible in the sense that any ANN method could be used as long as it performs retrieval in sublinear time. Therefore, AMP can also be considered as a modular framework of algorithms.

Despite its simplicity, this modification raises questions. For example, it is not clear how fast AMP converges, its computational expense, or if it will converge at all. We do not yet know if AMP can obtain the exact same sparse decomposition as MP or OMP. We have not described how to find near neighbors, or how a near neighbor is defined. In the next section, we show how LSH can be used inside AMP and the advantages behind LSH over other ANN algorithms.

4.4 Locality Sensitive Hashing

AMP allows the use of any ANN algorithm that can perform retrieval in sublinear time. For this work, we focus on locality-sensitive hashing (LSH), a category of algorithms that places nearby points in a high-dimensional space into the same bin in a hash table. Because of its simplicity, robustness, and low complexity, LSH has become popular for solving many high-level problems beyond MIR such as search and retrieval of text and images. The robustness of LSH is desirable for problems in MIR where queries are often distorted due to environmental or musical variation, and therefore, learned dictionary atoms will rarely match predefined dictionary atoms exactly.

For example, Ryynänen and Klapuri used LSH to perform query-by-humming (QBH) by constructing a hash table from pitch contour vectors [88]. Each pitch vector approximates a melody contour over a small time window. During retrieval, for each query pitch vector, their method uses LSH to search for nearest neighbors in a Euclidean space from the index of database melody fragments to obtain melody candidates and their matching positions in time. Yu et al. use LSH and order statistics to store chroma features in a hash table for audio content retrieval [89]. Cotton and Ellis use LSH to store landmarks in audio that correspond to meaningful acoustic events [90]. Casey and Slaney have used LSH to store features called audio shingles for computing various levels of musical similarity between songs [91, 92, 93].

However, LSH has rarely been used for signal-level problems like music transcription. To our knowledge, this work is among the first in MIR to use LSH for low-level tasks such as sparse coding and music transcription.

Several ANN algorithms have been proposed over the last forty years. Although we use LSH within AMP in this dissertation, there are other ANN methods that could be used instead. The kd-tree is one of the oldest and simplest methods for computing approximate nearest neighbors [94]. Given a set of points, the kd-tree first partitions the set at the median value of the first coordinates of all points. These two partitions, P_L and P_R, are then partitioned again separately at the median value of the second coordinates of all points already belonging to the same partition. At each iteration, each partition is partitioned into two further partitions in the same manner. The resulting data structure is a binary tree with n

leaves and depth $O(\log n)$. To find an approximate nearest neighbor, given a query point, we traverse down the tree until a desired proximity from the query point is reached. At any internal node, all of the points that are children of an internal node are considered to be neighbors of each other.

While other ANN algorithms can be used within AMP instead of LSH, such as those that use space partitioning like the kd-tree and hierarchical k-means, these algorithms do not work well in high-dimensional spaces, i.e., dimensionality over 100. In fact, all current indexing techniques based on space partitioning degrade to linear search for sufficiently high dimensions [95, 96, 97]. Therefore, we only consider LSH in this work.

There are many theoretical results for LSH that are beyond the scope of this dissertation. Here, we highlight the main properties of LSH [97]:

1. Locality: For hash function h, constant $c > 1$, probabilities $P_1 > P_2$, and points $\mathbf{q}$ and $\mathbf{r}$,

   (a) If $d(\mathbf{q}, \mathbf{r}) \leq R$, then $\Pr(h(\mathbf{q}) = h(\mathbf{r})) \geq P_1$.

   (b) If $d(\mathbf{q}, \mathbf{r}) \geq cR$, then $\Pr(h(\mathbf{q}) = h(\mathbf{r})) \leq P_2$.

2. Complexity: For dimensionality M, dictionary size K, and exponent $\rho \in (0, 1)$, query time is

$$O(MK^\rho \log K).$$

   The exponent ρ is smaller as more approximation is tolerated.

In this work, for $i \in \{1, 2, ..., k\}$ and $\ell \in \{1, 2, ..., L\}$, we define the function h_i^ℓ

to be

$$h_i^\ell(\mathbf{q}) = \operatorname{sgn}\langle \mathbf{p}_i^\ell, \mathbf{q}\rangle \tag{4.2}$$

where $\mathbf{p}_i^\ell$ is a zero-mean, unit variance, Gaussian random vector with independent elements. It has been shown that this choice of distribution on $\mathbf{p}_i^\ell$ will hash points together whose angle,

$$\theta(\mathbf{q}, \mathbf{r}) = \arccos \frac{\langle \mathbf{q}, \mathbf{r}\rangle}{||\mathbf{q}||||\mathbf{r}||}, \tag{4.3}$$

is small [98]. Specifically, it can be shown that, for any i and ℓ,

$$P(h_i^\ell(\mathbf{q}) = h_i^\ell(\mathbf{r})) = 1 - \frac{\theta(\mathbf{q}, \mathbf{r})}{\pi}. \tag{4.4}$$

We claim that two hashes are equal, $h(\mathbf{q}) = h(\mathbf{r})$, if and only if there exists an ℓ such that, for all $i \in \{1, 2, ..., k\}$, $h_i^\ell(\mathbf{q}) = h_i^\ell(\mathbf{r})$. In other words, the following events are equivalent:

$$\{h(\mathbf{q}) = h(\mathbf{r})\} = \cup_{\ell=1}^{L} \cap_{i=1}^{k} \{h_i^\ell(\mathbf{q}) = h_i^\ell(\mathbf{r})\}. \tag{4.5}$$

Then, given the probability in Eq. 4.4, it can be shown that the probability that $h(\mathbf{q}) = h(\mathbf{r})$ is equal to

$$P(h(\mathbf{q}) = h(\mathbf{r})) = 1 - \left(1 - \left(1 - \frac{\theta(\mathbf{q}, \mathbf{r})}{\pi}\right)^k\right)^L. \tag{4.6}$$

To construct the LSH table, we initialize L empty tables. For each atom $\mathbf{a}$ in

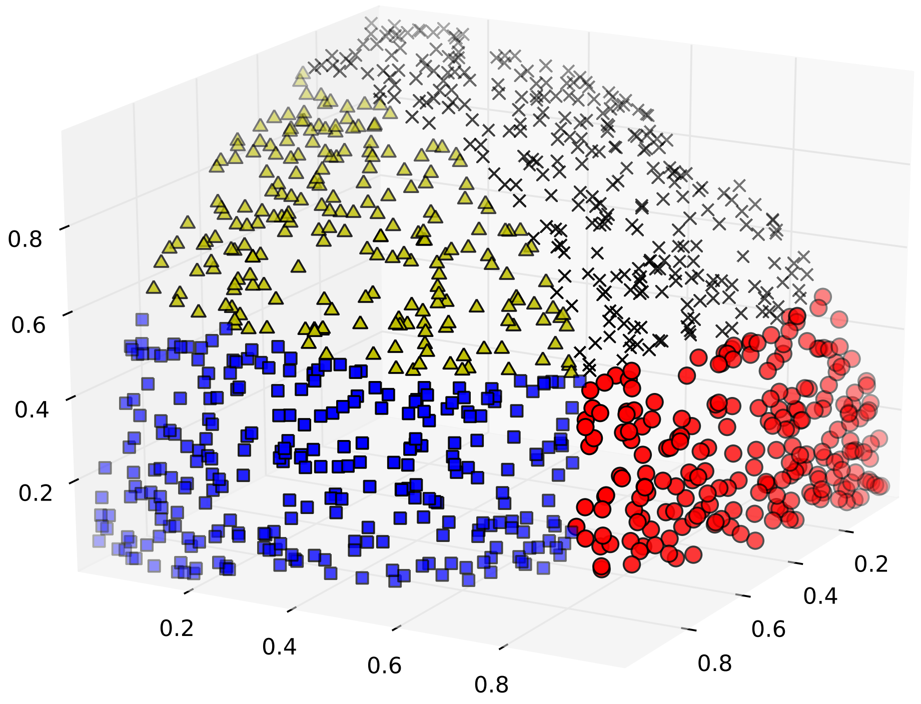


Figure 4.1: LSH example with $k = 2$. Points on the unit sphere are separated into $2^k = 4$ bins.

the dictionary $\mathbf{A}$, and for each $\ell \in \{1, 2, ..., L\}$, its hash is computed as a k-tuple:

$$h^\ell(\mathbf{a}) = (h_1^\ell(\mathbf{a}), h_2^\ell(\mathbf{a}), ..., h_k^\ell(\mathbf{a})), \tag{4.7}$$

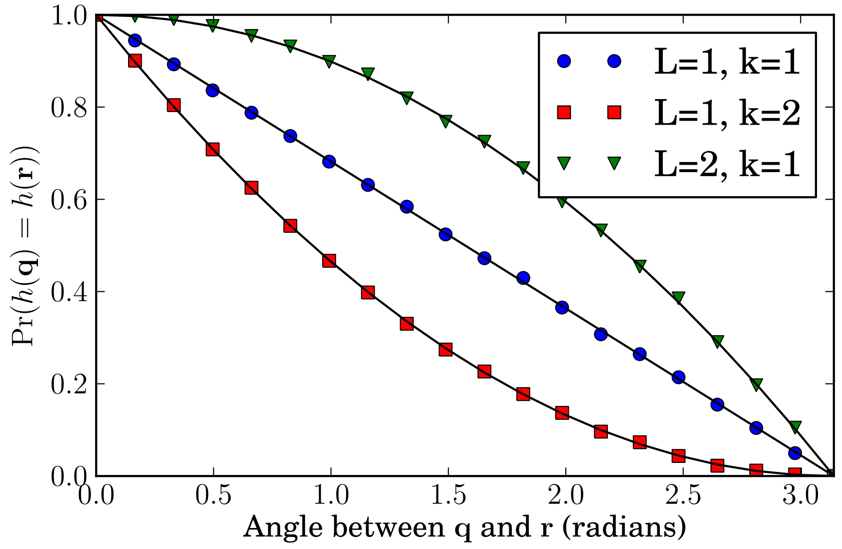
and $\mathbf{a}$ is placed into bin $h^\ell(\mathbf{a})$ of table ℓ. Finally, to perform a query for point $\mathbf{r}$, for all ℓ, we retrieve all of the points in bin $h^\ell(\mathbf{r})$ of table ℓ. Among these retrieved points that share a bin with $\mathbf{r}$, we perform exhaustive search to find the nearest neighbor among them. As indicated by Eq. 4.6, through the proper choice of k and L, one can achieve any desired amount of similarity between any two input vectors.

An example of LSH is shown in Figure 4.1 when $k = 2$. Points on the unit sphere are hashed, and those points that reside in the same bin share the same marker. We notice that points in the same bin are close together.

There are few existing implementations of LSH that are publicly available, e.g., E^{2}LSH and LSHKIT. For this work, we implement our own LSH system using Python. To confirm the validity of our implementation, we compare the numerical results generated from our implementation with theoretical probability shown in Eq. 4.6. In Figure 4.2, we plot the angle between two vectors versus the probability that their hashes are equal. For every vector pair, $\mathbf{q}$ and $\mathbf{r}$, we perform 10,000 trials, and we count the number of times that $h(\mathbf{q})$ equals $h(\mathbf{r})$ divided by 10,000. The markers indicate the proportion of pairs of vectors whose hashes are equal as a function of the angle between them. The dark lines represent the expected theoretical result. As we see from Figure 4.2, the numerical results agree with the theory. As L increases, hashes are more likely to be equal. As k increases, hashes are less likely to be equal.

4.5 Experiments

First, we show the accuracy, noise robustness, and complexity of AMP versus other methods by using synthetic data. In these experiments, we use a dictionary containing atoms that are uniformly distributed upon the unit sphere in $\mathbb{R}^M$. As a baseline, we compare AMP against OMP and STOMP. For STOMP, during each iteration, we simply return all database entries that are within 0.9 times the similarity of the nearest neighbor. While Donoho et al. use a different thresholding method, that method is merely one option under certain Gaussian assumptions [3]; under other assumptions, other thresholds may become valid. Ordinary MP and BP both perform far slower than these methods, and therefore we do not state their results here.



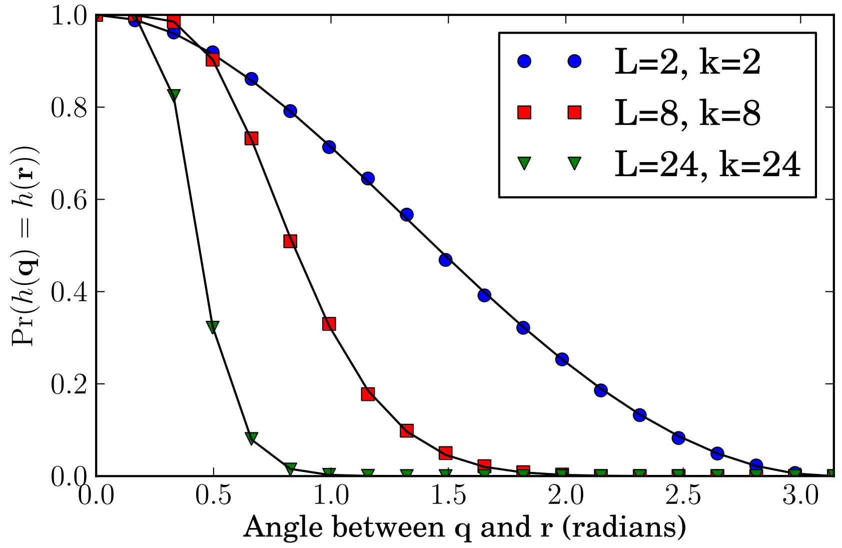


Figure 4.2: Probability of two vectors having equal hashes as a function of the angle between the vectors. Markers indicate our implementation. Dark lines indicate the expected theoretical result.

All source code is written by the authors in Python using the NumPy and SciPy packages [67].

4.5.1 Learning Curves

Figure 4.3 illustrates learning curves at different dimensionalities for OMP, STOMP, and AMP using LSH with different parameters. For each trial, the input vector $\mathbf{x}$ is generated as $\mathbf{x} = \mathbf{A}\mathbf{s}_0$, where $\mathbf{A}$ contains atoms that are uniformly distributed about the unit sphere, and the nonzero elements of $\mathbf{s}_0$ are randomly selected and are uniformly distributed over the interval [0,1]. We set the dictionary size $K = 1000$ and sparsity $||\mathbf{s}_0||_0 = 10$. We stop iterating when $||\mathbf{x} - \mathbf{A}\mathbf{s}|| < 10^{-10}||\mathbf{x}||$.

We plot the cost $||\mathbf{x} - \mathbf{A}\mathbf{s}||$ in Figure 4.3 as a function of the vector dimension, M, and the iteration number. For most cases, OMP learns a perfect representation of $\mathbf{x}$ in $||\mathbf{s}_0||_0$ iterations because of orthogonalization. Only for a few instances when M is low with respect to K does it take more iterations to learn a perfect representation. We expect this behavior because OMP is a greedy algorithm and does not guarantee an optimal solution. When M/K is low, then the K dictionary atoms are more densely compacted upon the unit sphere, and it becomes more likely that each iteration of matching pursuit may retrieve the wrong dictionary atom. For STOMP, we find similar convergence, although it converges in fewer iterations. However, each iteration of STOMP adds multiple atoms to the set of nonzero coefficients, and therefore the cost per iteration is higher than OMP.

Finally, we find that, while the convergence rate per iteration of AMP is upper

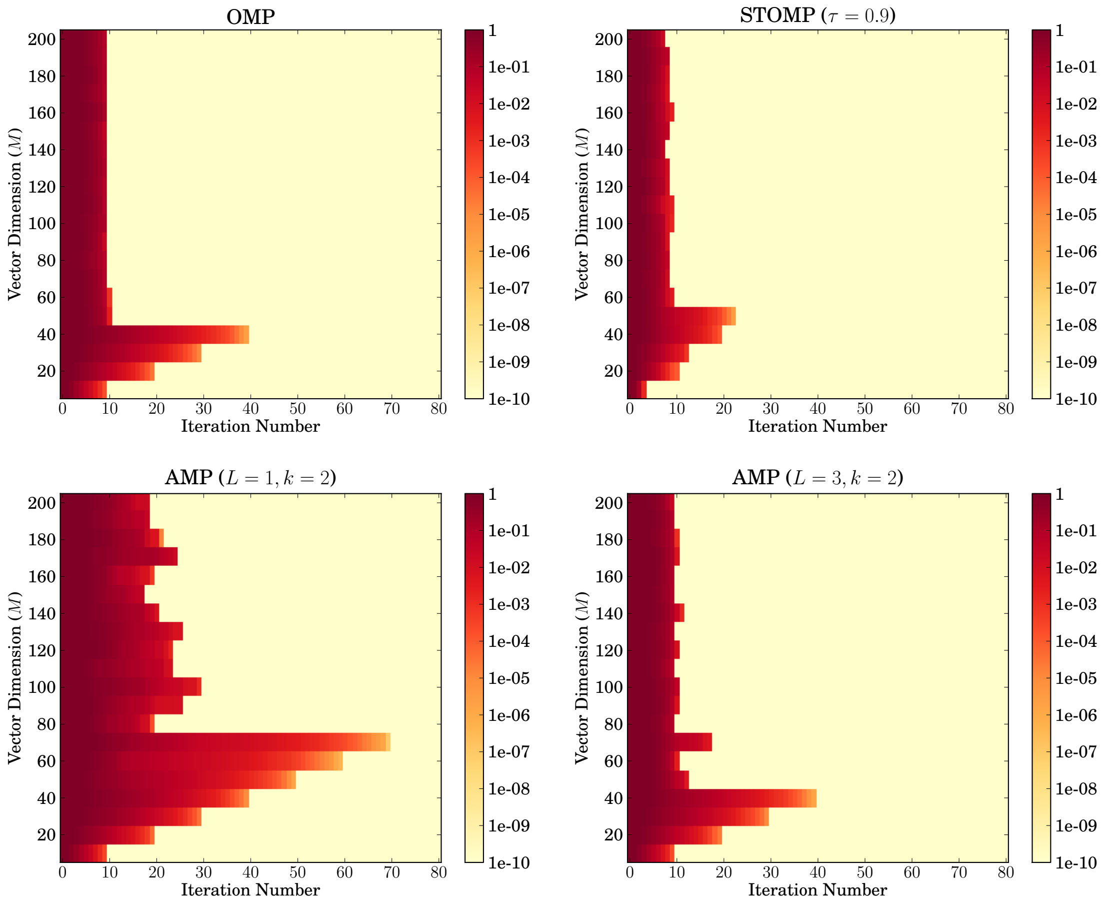


Figure 4.3: Representation cost, $||\mathbf{x}-\mathbf{As}||$, for OMP, STOMP, and AMP, as a function of the vector dimension, M, and the iteration number, averaged over ten trials. Dictionary atoms are uniformly distributed upon the unit sphere, dictionary size $K = 1000$, and sparsity $||\mathbf{s}_0||_0 = 10$.

bounded by that of OMP, AMP has convergence that is not significantly worse than OMP or STOMP. We also find that the number of tables, L, affects convergence rate. When only one hash table is used, more iterations are required to achieve convergence than when three hash tables are used. However, as we will see later, the cost per iteration is higher when L increases. In conclusion, the convergence properties of AMP are not significantly different from OMP or STOMP.

4.5.2 Phase Transition Diagrams

The plots in Figure 4.4 are called phase transition diagrams in the sparse coding literature. We plot the ratio M/K versus the ratio $||\mathbf{s}_0||_0/M$. We fix $K = 200$, and plot the number of representation errors, $||\hat{\mathbf{s}} - \mathbf{s}_0||_0$, averaged over five trials, where $\hat{\mathbf{s}}$ denotes the coefficient vector at convergence. It has been shown that increasing M while keeping these two ratios fixed will result in a phase transition diagram that depicts a sudden transition between regions of success and failure [3].

These plots suggest that, for a variety of sparsity values and dimensionalities, AMP performs nearly as well as OMP and STOMP. Each algorithm works poorly for low sparsity, and the transition from the success region to the failure region is similar for all algorithms. In conclusion, AMP works nearly as well as OMP and STOMP for a variety of sparsity values and dimensionalities.

4.5.3 Noise Robustness

In Figure 4.5, we show the robustness of these algorithms in the presence of additive Gaussian noise. We synthesize a noisy input as $\mathbf{z} = \mathbf{x} + \mathbf{n}$, where the signal $\mathbf{x}$ is equal to $\mathbf{A}\mathbf{s}_0$, and $\mathbf{n}$ is a zero-mean Gaussian random vector. The variance of the noise depends upon the signal-to-noise ratio (SNR), which is computed in decibels as $20\log_{10}(||\mathbf{x}||/||\mathbf{n}||)$ dB.

As in the previous subsection, we plot the number of representation errors as a function of the ratio $||\mathbf{s}_0||_0/M$ as well as the SNR. For all of the algorithms, reconstruction is poor when the SNR is below 50 dB or the number of nonzero

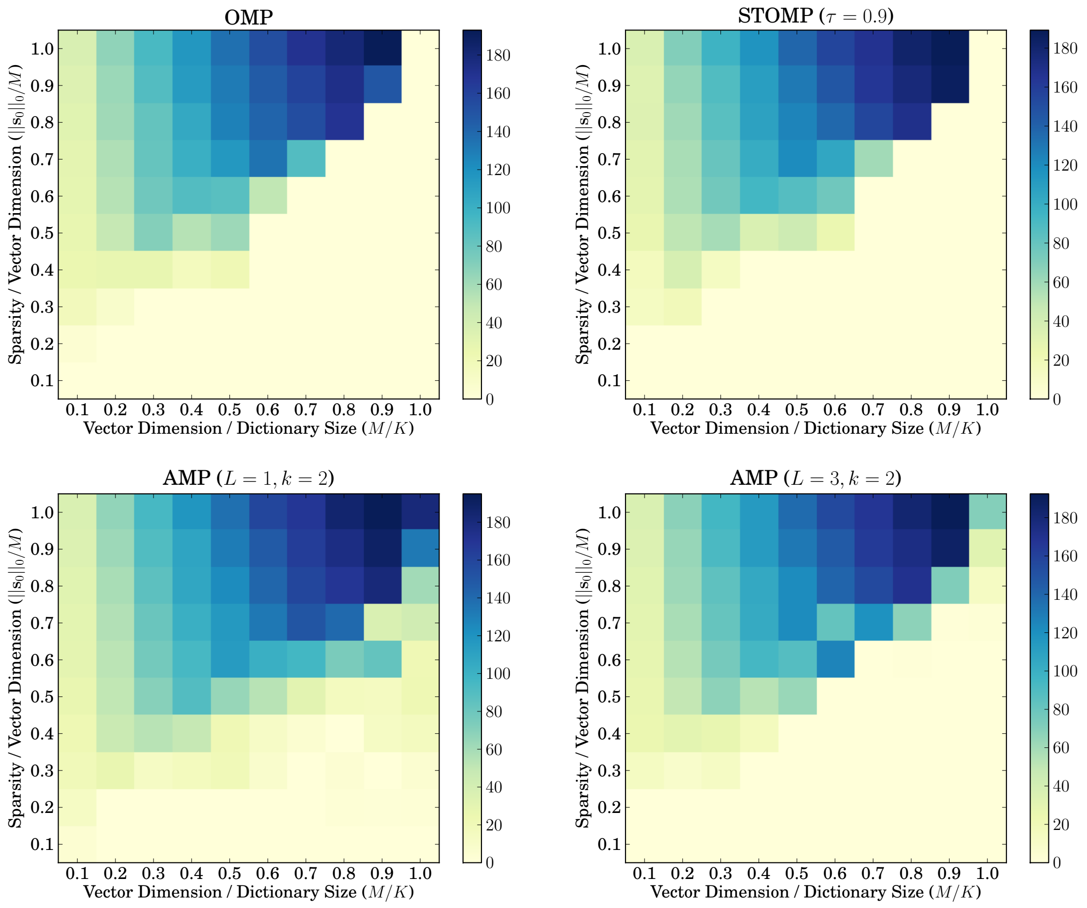


Figure 4.4: Number of representation errors, $||\hat{\mathbf{s}} - \mathbf{s}_0||_0$, for OMP, STOMP, and AMP as a function of sparsity, $||\mathbf{s}_0||_0/M$, and vector dimension, M/K, averaged over five trials. Dictionary size is $K = 200$. Dictionary atoms are uniformly distributed about the unit sphere.

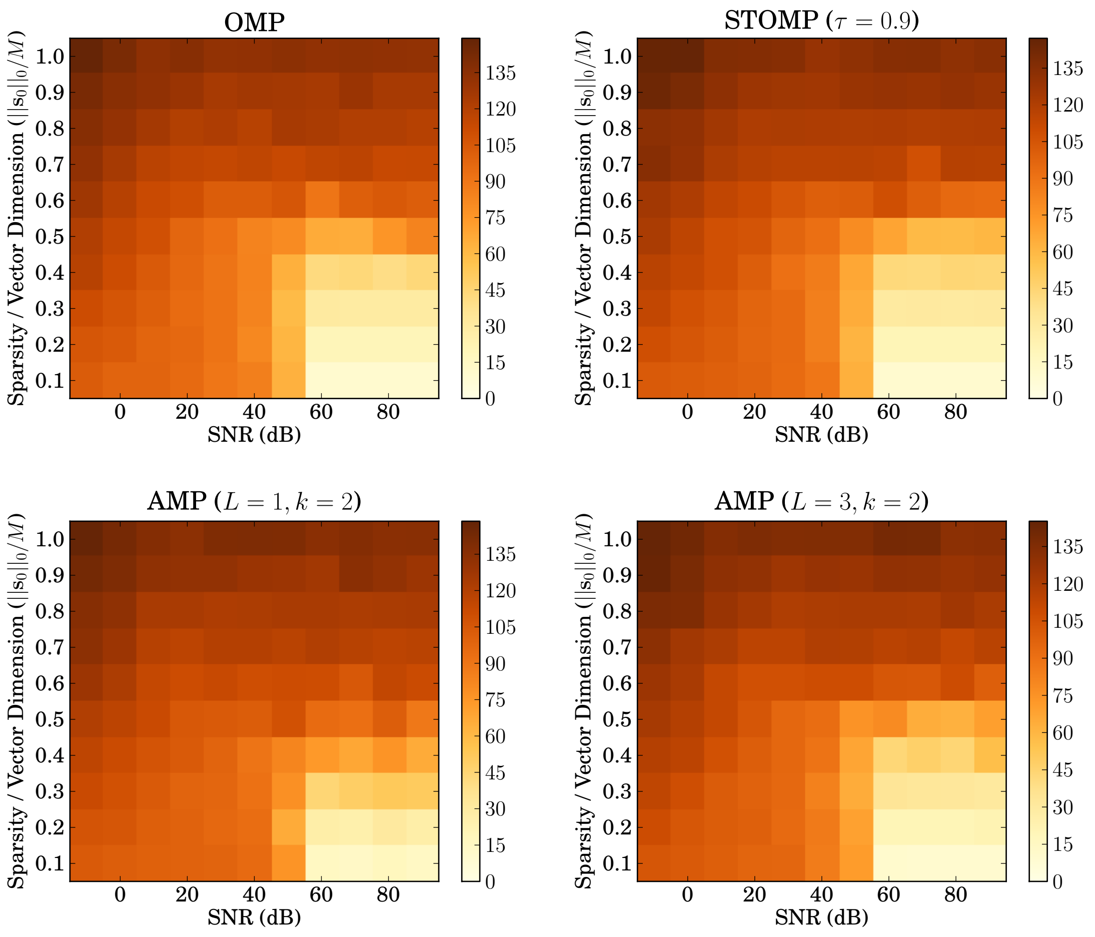


Figure 4.5: Number of representation errors, $||\hat{\mathbf{s}} - \mathbf{s}_0||_0$, for OMP, STOMP, and AMP as a function of sparsity, $||\mathbf{s}_0||_0/M$, and SNR, averaged over five trials. Dictionary size is $K = 200$, and vector dimension is $M = 100$. Dictionary atoms are uniformly distributed about the unit sphere.

coefficients is above $0.4M$. In conclusion, AMP is as robust to additive Gaussian noise as OMP and STOMP.

4.5.4 Computational Complexity

As we saw in Section 4.5.1 and Figure 4.3, AMP performs nearly as well as OMP and STOMP when measured by the residual cost per iteration. However, the amount of computation per iteration is far lower for AMP than for OMP and STOMP. In

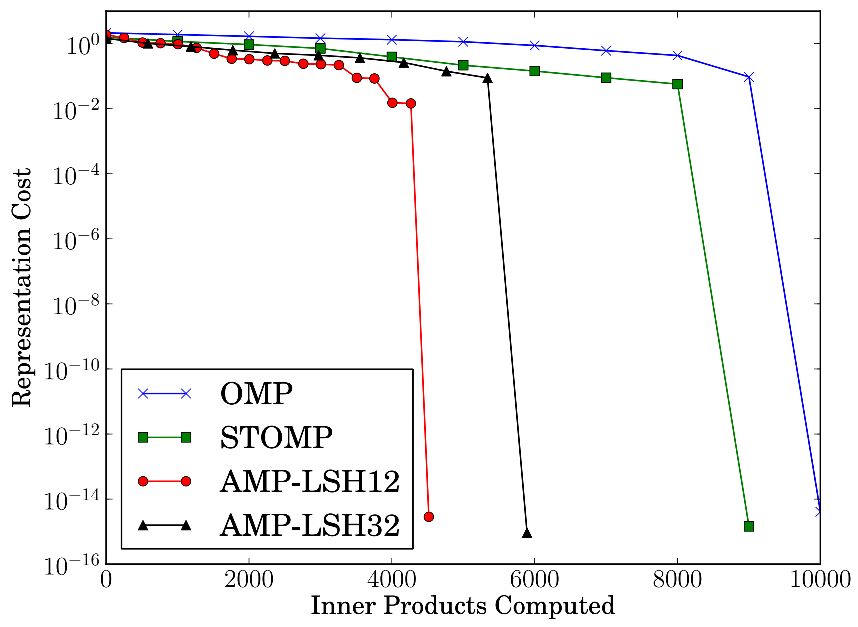


Figure 4.6: Representation cost, $||\mathbf{x} - \mathbf{A}\hat{\mathbf{s}}||$, for OMP, STOMP, and AMP, as a function of the number of inner products computed. Dictionary atoms are uniformly distributed upon the unit sphere, dictionary size $K = 1000$, dimensionality $M = 100$, and sparsity $||\mathbf{s}_0||_0 = 10$.

Figure 4.6, we again plot learning curves, but now as a function of the number of M-dimensional inner products computed. This plot illustrates the differences in complexity among the algorithms, particularly how AMP outperforms OMP and STOMP. All algorithms achieve convergence, but AMP reaches convergence using fewer inner products. The reason is evident. OMP and STOMP both require K inner products at each iteration, even though STOMP tends to achieve convergence in fewer iterations. On the other hand, AMP requires far less than K inner products per iteration. Depending upon the number of hash tables used, L, and the size of each hash, k, the number of inner products can vary. Nevertheless, whether $L = 1$ or 3, both AMP outperform OMP and STOMP in terms of complexity.

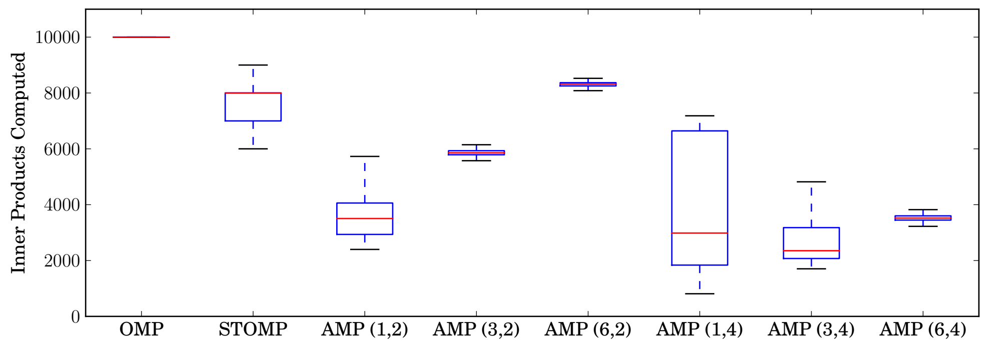


Figure 4.7: Distribution of the number of inner products performed before convergence over 300 trials. Dictionary atoms are uniformly distributed upon the unit sphere, dictionary size $K = 1000$, dimensionality $M = 100$, and sparsity $||\mathbf{s}_0||_0 = 10$.

Because LSH is a randomized algorithm, the exact number of inner products cannot be determined beforehand. Therefore, in Figure 4.7, we illustrate the distribution of the number of inner products required by each algorithm over 300 trials. Each box-and-whisker set shows the distribution of inner products divided by quartile. We plot these distributions for OMP, STOMP, and AMP using LSH with parameters $L \in \{1, 3, 6\}$ and $k \in \{2, 4\}$. This figure clearly shows that AMP requires fewer inner products over a variety of parameters. When $L = 1$, the median number of inner products is smallest because there are fewer hashes to compute. Another interesting fact is that the variation in the number of inner products is greatest when $L = 1$ and smallest when $L = 6$. This behavior can be explained using the law of large numbers; because L retrievals are made, the variance in the number of items retrieved is proportional to $1/L$. Also, the variation is greater for $k = 4$ than for $k = 2$; when k is large, the number of bins per hash table increases, and so does the variance in the number of items per bin.

Finally, in Figure 4.8, we plot the number of inner products computed as a

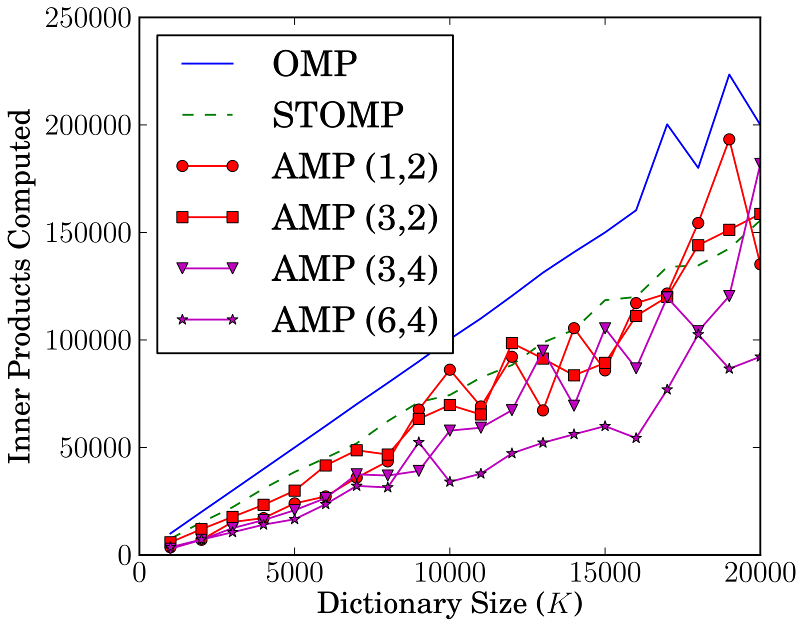


Figure 4.8: Number of inner products averaged over fifty trials as a function of the dictionary size, K. The dimensionality is $M = 100$, and the sparsity is $||\mathbf{s}_0||_0 = 10$.

function of the dictionary size, K. The number of inner products is averaged over fifty trials. We fix the dimensionality to be $M = 100$, the sparsity as $||\mathbf{s}_0||_0 = 10$, and we vary the dictionary size from 1000 to 20000. We find that AMP uses fewer inner products than OMP or STOMP for most values of K. Despite averaging results over fifty trials, there is still a fair amount of variance among the number of inner products required because LSH is a randomized algorithm, especially for large dictionary sizes. In future work, we shall further investigate the empirical impact of dictionary size on computational complexity.

4.6 Music Transcription

Here, we show how AMP can be used to perform music transcription. First, we discuss how to build a dictionary, and then we provide examples on real musical signals.

4.6.1 Dictionary Construction

The success of this method is determined in part by how representative the dictionary is over the space of inputs. For example, when the input is a spectrum of a musical signal, we expect the dictionary to contain plenty of musical spectra with similar characteristics. Here, we describe how to build a large dictionary to describe a wide range of musical sounds.

For this work, our data comes from the University of Iowa database of musical instrument samples [68], a data set often used in MIR. Each file in the data set is labeled by pitch and loudness, e.g., “Piano C4 mf”, and contains a signal of an isolated note sampled at 44100 Hz. We only consider the subset of piano sounds; in future work, we will consider other instruments, as well. Nevertheless, accurate determination of multiple pitches from polyphonic piano music remains a nontrivial task that we address here.

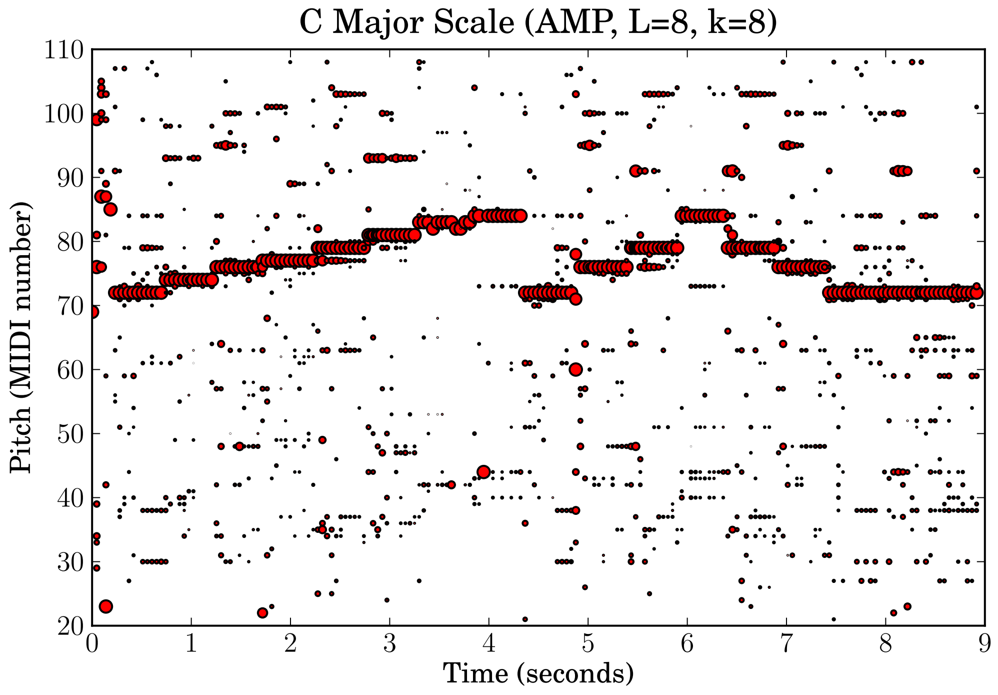
For each signal, we compute a short-time Fourier transform with frame size of 92.9 milliseconds (i.e., 4096/44100) and a hop of 10 milliseconds. To discard silent segments, we detect any spectrum whose power is below a threshold. The remaining spectra are normalized to have unit Euclidean norm and are saved on disk using the HDF5 format along with their pitch and instrument labels. These normalized spectra constitute the dictionary. In total, we have a dictionary of 176,339 spectra of piano sounds covering the entire piano keyboard (i.e., MIDI values 21 through 108).

4.6.2 Examples

In Figure 4.9, we plot the output from AMP for a C-major scale played by a piano. This file was recorded by the authors as a single-channel waveform sampled at 44100 Hz. We obtain the magnitude spectrogram of this signal using a frame size of 92.9 milliseconds and a hop of 46.4 milliseconds. Each column of this spectrogram is decomposed using AMP with the maximum number of iterations set to 8. The parameters used in LSH are $L = 12$ and $k = 10$. Each point on the scatter plot represents one coefficient in the output vector, $\hat{\mathbf{s}}$. The area of each circle is proportional to the magnitude of the corresponding coefficient. (For display purposes, we set constant the area of the largest circle in each frame.) In Figure 4.10, we plot the output from OMP for the same signal.

All of the scatter plots clearly indicate the correct notes being played by the piano. Both of the plots for AMP are very close to that of OMP. In Table 4.1, we show the execution times for each of the algorithms. AMP is clearly faster than OMP for multiple choices of parameters for L and k.

In Figures 4.11 and 4.12, we show the outputs from AMP and OMP for *Clair de Lune* by Claude Debussy, measures 1–4. Here, the sparse decompositions are not



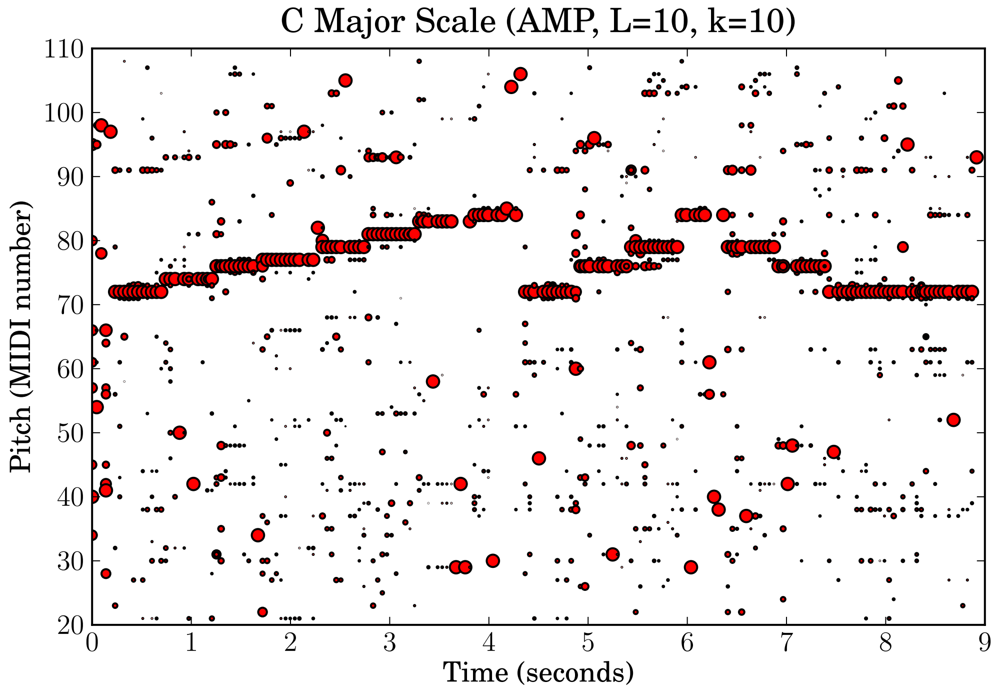
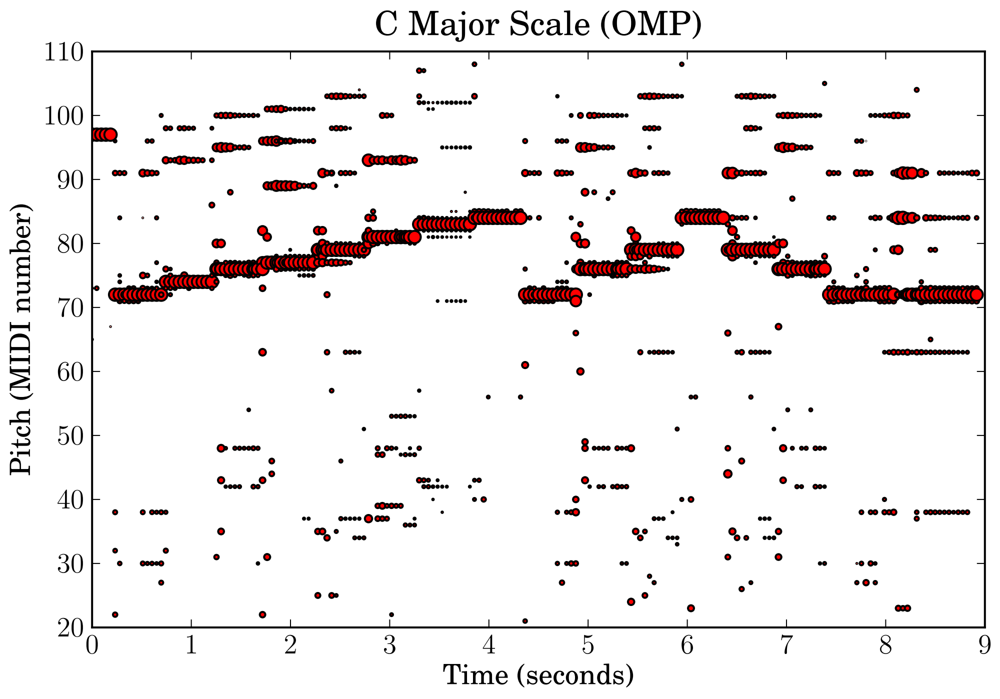


Figure 4.9: Scatter plots of atoms as detected by AMP using LSH with parameters $(L, k) = (8, 8)$ and $(10, 10)$ for a C-Major scale. The maximum number of iterations per frame, and equivalently $||\hat{\mathbf{s}}||_0$, is set to 8.



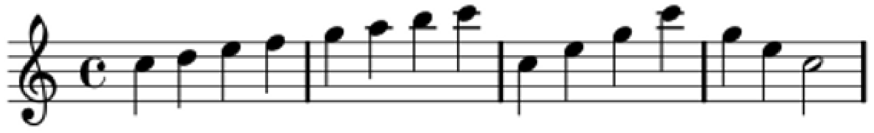


Figure 4.10: Top: scatter plot of atoms as detected by OMP for a C-Major scale. The maximum number of iterations per frame, and equivalently $||\hat{\mathbf{s}}||_0$, is set to 8. Bottom: corresponding sheet music notation.

Song	OMP	$\mathrm{AMP}_{8,8}$	$\mathrm{AMP}_{10,10}$
C-major scale	81.05	43.63	21.03
Debussy mm. 1-4	118.57	88.45	29.01
Debussy mm. 5-8	123.05	121.73	121.84

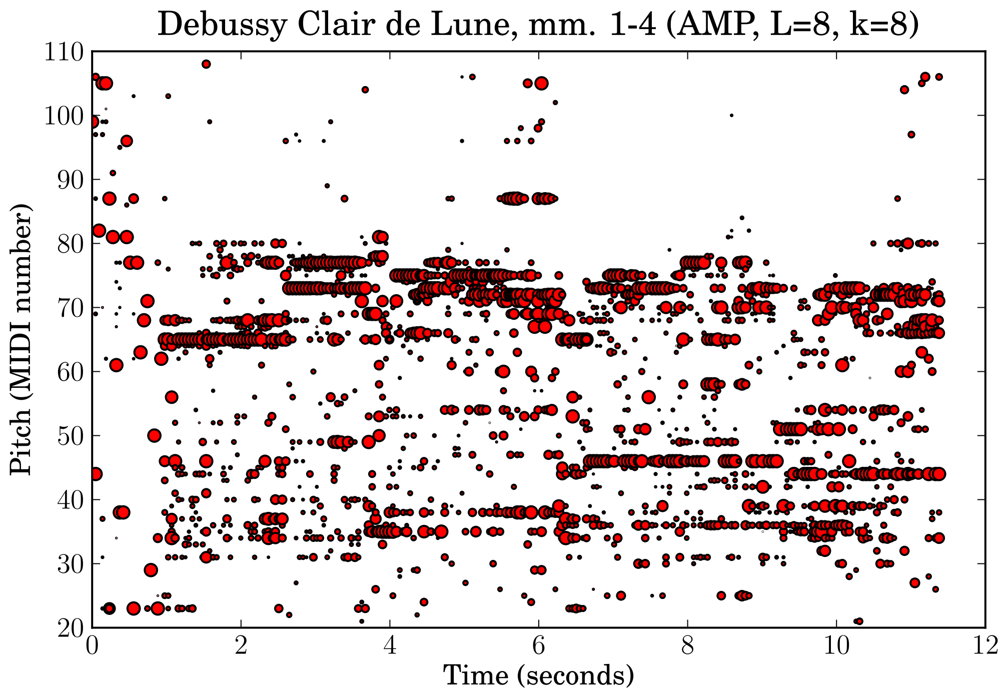
Table 4.1: Execution times in seconds.

as clear, even for OMP. A subsequent postprocessing step may help during transcription. Nevertheless, Table 4.1 shows that AMP is still faster. Finally, Figures 4.13 and 4.14 show outputs for *Clair de Lune*, measure 5–8. This passage contains more overlapping notes and a denser decomposition. Interestingly, in Table 4.1, we find that the benefits of AMP are not as pronounced in this case, perhaps due to the polyphonic nature of this excerpt.

4.7 Summary

We have proposed AMP, a pursuit algorithm that is nearly as accurate and robust as OMP while requiring fewer computations to achieve convergence. This algorithm is fast and scalable; complexity is sublinear in the dictionary size, and expanding the dictionary is straightforward. Given our particular choice of the LSH hash function which maps vectors whose cosine distance is small into the same bin, we have shown that using LSH reduces the number of computations while maintaining an accurate sparse decomposition. Finally, we illustrated the usefulness of AMP for music transcription. By visualizing the pitch labels of the largest sparse coefficients, we find that AMP can accurately retrieve the proper dictionary atoms.

The execution times in Table 4.1 are encouraging, but as Figures 4.11 and 4.13 show, there is room for improvement. As future work, we will investigate the impact of the LSH parameters L and k more closely. The proper choice of these parameters depends upon many factors including the type of music, the dimensionality, and the dictionary size. A careful analysis of pairwise distances among dictionary atoms can



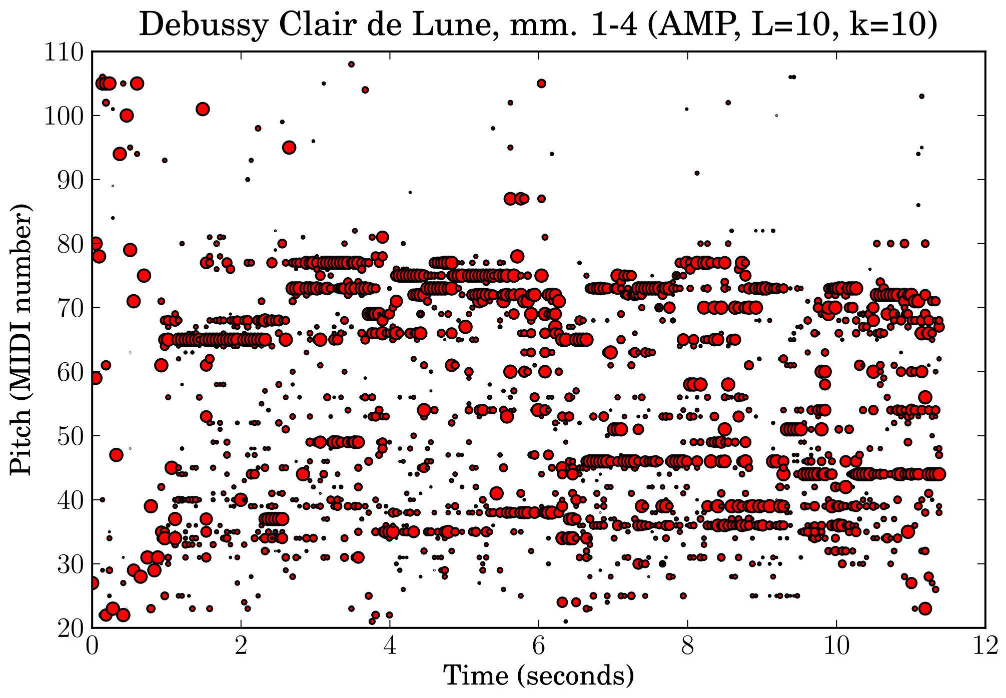
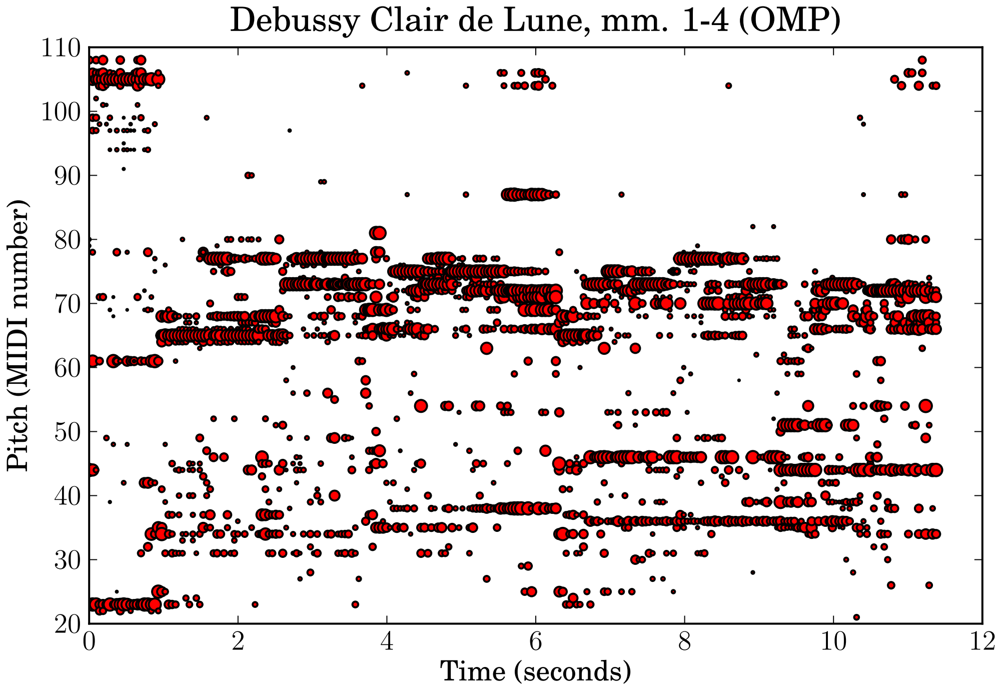


Figure 4.11: Scatter plots of atoms as detected by AMP using LSH with parameters $(L, k) = (8, 8)$ and $(10, 10)$ for *Clair de Lune* by Debussy, mm. 1-4. The maximum number of iterations per frame, and equivalently $||\hat{\mathbf{s}}||_0$, is set to 8.



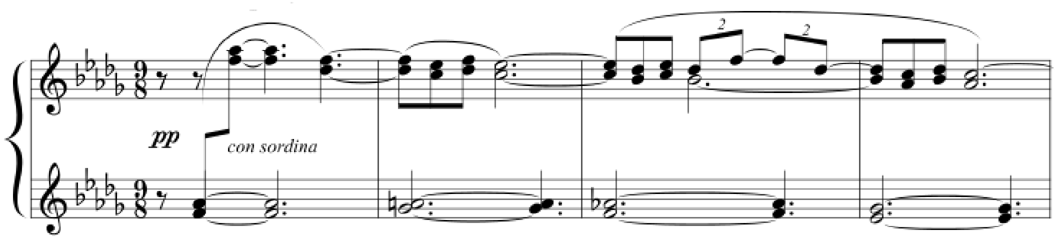
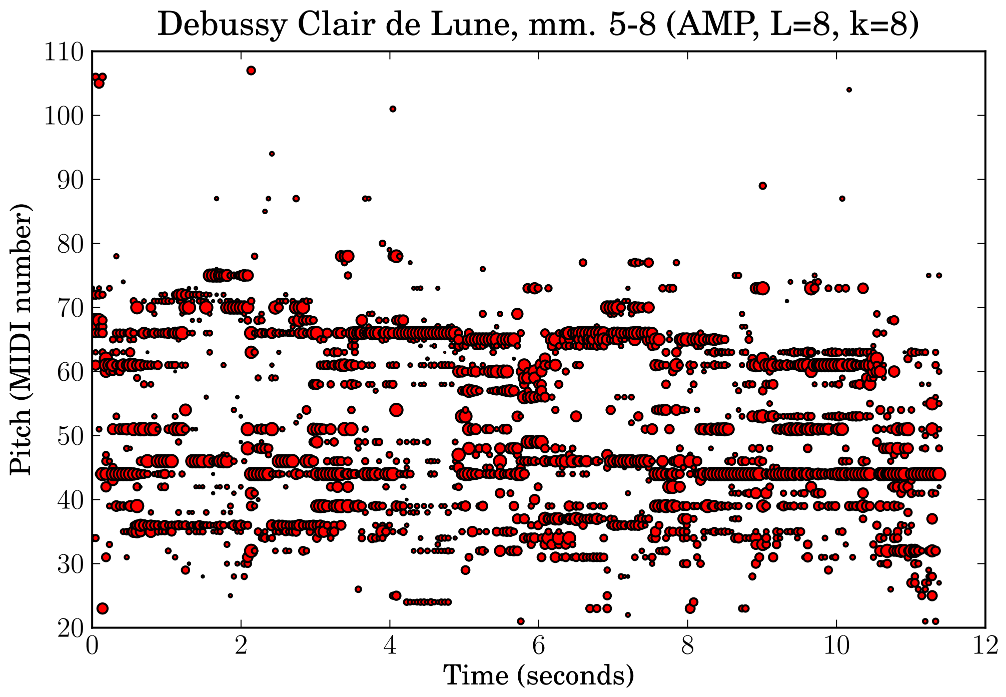


Figure 4.12: Top: scatter plot of atoms as detected by OMP for *Clair de Lune* by Debussy, mm. 1-4. The maximum number of iterations per frame, and equivalently $||\hat{\mathbf{s}}||_0$, is set to 8. Bottom: corresponding sheet music notation.



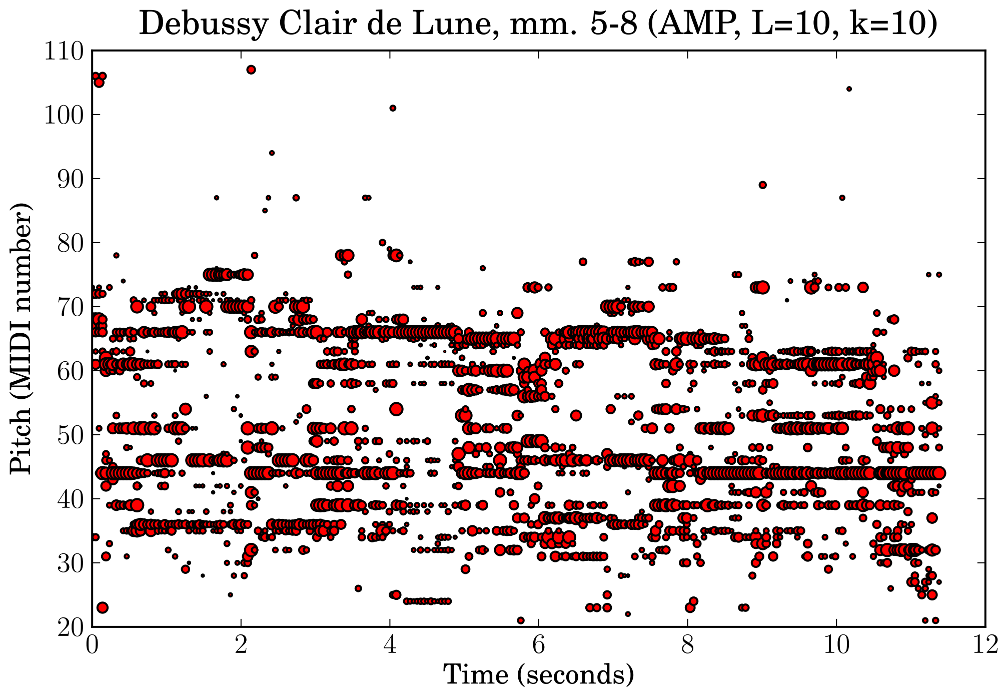
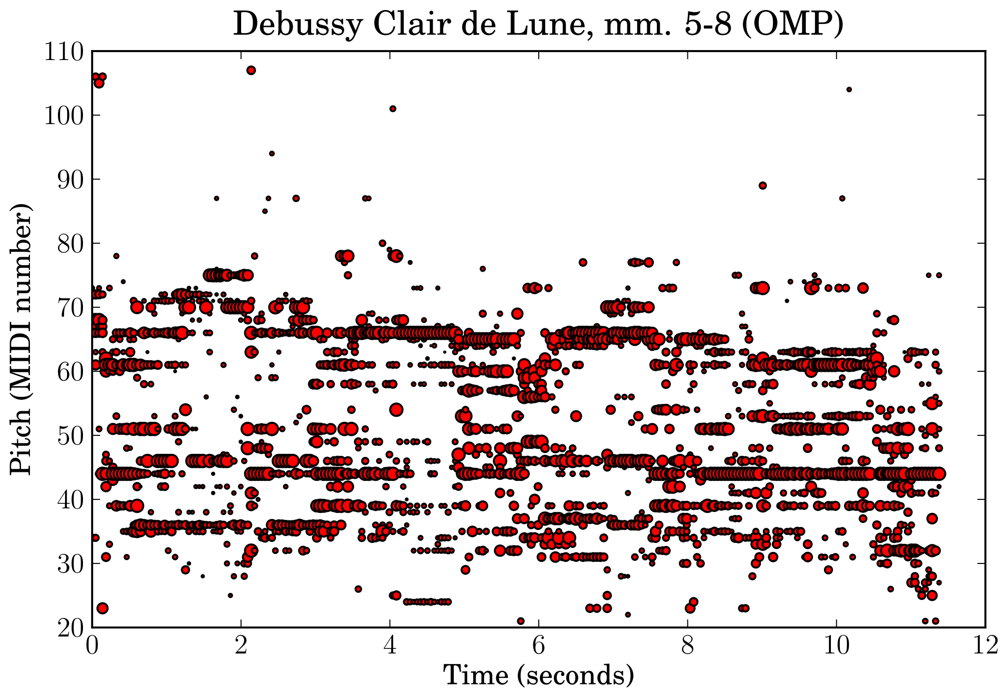


Figure 4.13: Scatter plots of atoms as detected by AMP using LSH with parameters $(L, k) = (8, 8)$ and $(10, 10)$ for *Clair de Lune* by Debussy, mm. 5-8. The maximum number of iterations per frame, and equivalently $||\hat{\mathbf{s}}||_0$, is set to 8.



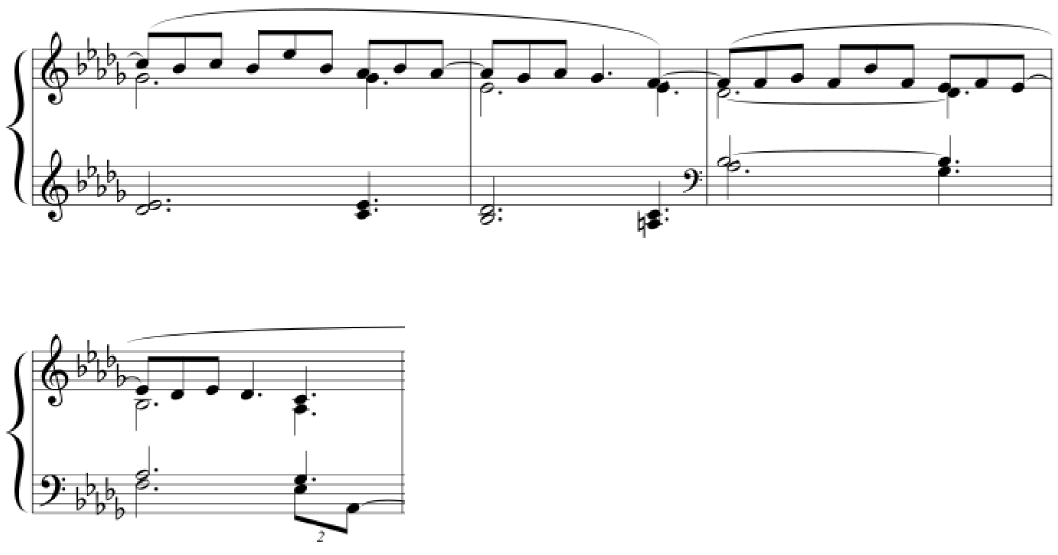


Figure 4.14: Top: scatter plot of atoms as detected by OMP for *Clair de Lune* by Debussy, mm. 5-8. The maximum number of iterations per frame, and equivalently $||\hat{\mathbf{s}}||_0$, is set to 8. Bottom: corresponding sheet music notation.

reveal which set of parameters minimizes the probability of error. The dictionary itself has a significant impact on the decomposition. Therefore, future work will also include proper dictionary design, i.e., how to create dictionary atoms from musical data sets for maximum accuracy and efficiency.

After using AMP in the manner shown in this chapter, we want to use the decomposition to perform full music transcription. Future work includes measuring the transcription accuracy of AMP by clustering common pitch and instrument labels together as a single note by using onset and offset detection and temporal smoothing. We also believe that AMP can be used directly in the time domain like other methods that decompose signals as combinations of Gabor atoms [79, 76, 77, 78]. AMP can also be used entirely for instrument recognition by finding the most common instrument labels among the sparse coefficients of the decomposition.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this dissertation, we have explored three important areas in sparse and nonnegative factorization for music understanding.

1. Constrained dictionary learning. Adding constraints to the learning process allows us to learn dictionary atoms that are semantically meaningful in a musical context.

2. Spectral-temporal musical instrument recognition. NMF decomposes signals into a product of spectral atoms and temporal atoms, both of which can be used to improve upon the state of the art in instrument recognition.

3. Fast, fixed-dictionary factorization using approximate matching pursuit. When using a large, overcomplete, predefined dictionary for factorization, approximate matching pursuit can provide sparse decompositions that are nearly as accurate and robust as orthogonal matching pursuit while reducing computational complexity.

The work presented in this dissertation represents a significant technological advance in the way factorizations are applied to problems in music information retrieval. We have focused on MIR problems at low, fundamental levels such as the signal level. Adequately solving such problems will make an enormous impact throughout the world of MIR. The seminal works in NMF and sparse coding [99, 5, 6, 14] may have already revolutionized signal processing, but there remains plenty of progress to be made in order to refine these methods for specific application domains such as MIR. In their unmodified forms, these factorization methods simply cannot perform tasks such as music transcription, source separation, or classification at the level that end users expect from a commercial-strength music understanding system.

This dissertation brings us closer to that goal by improving upon ordinary NMF and sparse coding algorithms. By intelligently adding constraints and making use of available data, we were able to improve transcription performance, classification performance, and reduce complexity. As a result, subsequent systems that annotate discrete musical events will have an easier time doing so. Furthermore, this work may make an additional impact on neighboring application domains such as signal enhancement, noise removal, superresolution, and more.

First, in Chapter 2, we presented a novel method of dictionary learning based upon nonnegative K-SVD which can separate sources that are otherwise inseparable using common methods. Despite the simplicity of our algorithm, it performs well for a variety of musical scenarios involving pitched sounds with spectral-temporal overlap. We also proposed novel multiplicative update rules that impose co-occurrence constraints on either of the matrices produced through NMF. These update rules can minimize different divergence metrics, and they integrate easily with the basic NMF multiplicative updates. The constraints are useful when representing objects with multiple atoms, and they provide a natural way to cluster co-occurring atoms. Examples involving music transcription show that these constraints are operate successfully either in the frequency or time domains.

Second, in Chapter 3, we presented a method for extracting information from NMF atoms at multiple resolutions for the purpose of musical instrument recognition. Inspired by the early cortical stage of the human auditory system, this method performs multiresolution analysis upon NMF atoms through the use of MFCCs and a *multiresolution gamma filterbank*. The filters that compose this filterbank are capable of describing any combination of attack time and decay rate. Although there are other works that explore the combination of spectral and temporal information for instrument recognition, to our knowledge, this work is the first to parameterize the profiles of temporal NMF atoms through multiresolution analysis. Our original hypothesis was that the combination of spectral and temporal information extracted from NMF atoms would improve instrument recognition over systems that use spectral information alone. The results support this hypothesis, but the conclusion is stronger for isolated sounds than it is for solo melodic phrases where the improvement in performance is less significant.

Finally, in Chapter 4, we presented AMP, a pursuit algorithm that is nearly as accurate and robust as OMP while requiring fewer computations to achieve convergence. This algorithm is fast and scalable; complexity is sublinear in the dictionary size, and expanding the dictionary is straightforward. Given our particular choice of the LSH hash function which maps vectors whose cosine distance is small into the same bin, we have shown that using LSH reduces the number of computations while maintaining an accurate sparse decomposition. Finally, we illustrated the usefulness of AMP for music transcription. By visualizing the pitch labels of the largest sparse coefficients, we find that AMP can accurately retrieve the proper dictionary atoms.

5.2 Future Research

The MIR community has only begun to scratch the surface of possibilities related to sparse and nonnegative factorization. Although many researchers are now aware of its potential, additional work must be performed in order to bring the performance of these algorithms up to a satisfactory level for regular use by the general public.

First, there is a great need for studies related to the statistics of musical data. Many works impose assumptions related to sparsity or a particular probability distribution. However, it is not yet clear how well natural musical signals follow these assumptions, or how the statistics of musical signals change under environmental distortion such as additive noise or reverberation. It is also unclear how transformations of musical data – e.g., into chroma or constant-Q transform – changes the statistics of data. Therefore, we plan to investigate the robustness proposed algorithm under different acoustic conditions, particularly for music that includes additive noise or unpitched sources, along with decomposition of time-frequency representations and other generic transformations of natural music signals.

Then, to perform music transcription, the factorization output must be intelligently postprocessed. After learning a dictionary of perceptually meaningful atoms, the next stage involves clustering of the dictionary atoms according to their musical source. For example, all atoms sharing the same pitch and instrument label over a short interval could be treated as a single note; this note event could become an element of a MIDI file. While some clustering methods already exist, difficulties remain when doing this in an unsupervised manner. If combined with a successful atom clustering method, we believe that our proposed algorithms can offer state-of-the-art accuracy and robustness in music transcription and source separation tasks.

We also believe that these new factorizations will become useful in many applications addressed by NMF beyond music transcription and source separation. For example, AMP can also be used entirely for instrument recognition by finding the most common instrument labels among the sparse coefficients of the decomposition. It would be enlightening to compare the performance of AMP as an instrument recognition algorithm against our spectral-temporal instrument recognizer using NMF. AMP can also be used for search; the statistics of sparse coefficients of decomposed input signals can be observed using histogram analysis, and these statistics can be used to compare songs at varying levels of similarity. Some works already use LSH in a similar way for search and retrieval but over a much coarser time window and without any decomposition provided by matching pursuit.

In instrument recognition, an important direction for future research is to test this method on polyphonic music. Although we expect the task to become more difficult, we believe that the primary obstacle to successful classification is NMF. Ideally, if NMF can learn perfect dictionary atoms, then the polyphonic nature of music is not an obstacle. Unfortunately, NMF is not perfect. If NMF cannot accurately learn dictionary atoms, then any following feature extraction may fail.

Therefore, one direction for future research may include combining novel constrained dictionary learning algorithms such as the ones proposed in this dissertation with spectral-temporal instrument recognition using NMF. By improving the decomposition performance, subsequent processing using a multiresolution gamma filterbank will likely provide better results.

The boundary between locality-sensitive hashing and constrained factorization is also a new research frontier. Our work related to AMP is one of the first to employ LSH inside of a factorization algorithm. In the future, I fully expect enormous growth in the area of classification and factorization that makes use of side information. The gains that have been realized by unsupervised algorithms have reached a plateau. In order to break past this barrier, researchers must loosen assumptions related to the system model, for example, by assuming that side information is present such as manual user input via an interactive human-computer interface. This simple change in assumptions could be the impetus for achieving better performance gains and reducing complexity.

Despite these many directions for future research, no one can enumerate all of the open problems that are left to be solved in music information retrieval. Because of its highly interdisciplinary nature, scholars continue to raise new questions about music from different perspectives and provide truly innovative solutions. Our vision for the future of MIR has not yet been realized, but given the explosive progress witnessed thus far, we may not be far away.

Appendix A

Pursuit Algorithms

For Approximate Matching Pursuit (AMP), see Algorithm 1.

Algorithm 2 Matching Pursuit [1]

```
Input: $\mathbf{x} \in \mathbb{R}^M$; $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_K] \in \mathbb{R}^{M \times K}$ s.t. $||\mathbf{a}_k||_2 = 1$ for all $k$.
Output: $\hat{\mathbf{s}} \in \mathbb{R}^K$
Initialize: $\mathcal{S} \leftarrow \emptyset$; $\mathbf{s} \leftarrow \mathbf{0}$; $\mathbf{r} \leftarrow \mathbf{x}$; $\epsilon > 0$.
while $||\mathbf{r}|| > \epsilon$ do
    $k \leftarrow \text{argmax}_j \, \mathbf{a}_j^T \mathbf{r}$
    $\mathcal{S} \leftarrow \mathcal{S} \cup k$
    $\mathbf{r} \leftarrow \mathbf{x} - \mathbf{A}\mathbf{s}$
$\hat{\mathbf{s}} \leftarrow \mathbf{s}$
```

Algorithm 3 Orthogonal Matching Pursuit [2]

```
Input: $\mathbf{x} \in \mathbb{R}^M$; $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_K] \in \mathbb{R}^{M \times K}$ s.t. $||\mathbf{a}_k||_2 = 1$ for all $k$.
Output: $\hat{\mathbf{s}} \in \mathbb{R}^K$
Initialize: $\mathcal{S} \leftarrow \emptyset$; $\mathbf{s} \leftarrow \mathbf{0}$; $\mathbf{r} \leftarrow \mathbf{x}$; $\epsilon > 0$.
while $||\mathbf{r}|| > \epsilon$ do
    $k \leftarrow \text{argmax}_j \, \mathbf{a}_j^T \mathbf{r}$
    $\mathcal{S} \leftarrow \mathcal{S} \cup k$
    Solve for $\{s_j | j \in \mathcal{S}\}$: $\min_{s_j | j \in \mathcal{S}} ||\mathbf{x} - \sum_{j \in \mathcal{S}} \mathbf{a}_j s_j||$
    $\mathbf{r} \leftarrow \mathbf{x} - \mathbf{A}\mathbf{s}$
$\hat{\mathbf{s}} \leftarrow \mathbf{s}$
```

Algorithm 4 Stagewise Orthogonal Matching Pursuit [3]

```
Input: $\mathbf{x} \in \mathbb{R}^M$; $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_K] \in \mathbb{R}^{M \times K}$ s.t. $||\mathbf{a}_k||_2 = 1$ for all $k$.
Output: $\hat{\mathbf{s}} \in \mathbb{R}^K$
Initialize: $\mathcal{S} \leftarrow \emptyset$; $\mathbf{s} \leftarrow \mathbf{0}$; $\mathbf{r} \leftarrow \mathbf{x}$; $\epsilon > 0$.
while $||\mathbf{r}|| > \epsilon$ do
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{k | \mathbf{a}_k^T \mathbf{r} > \tau_{\mathbf{r}}\}$
    Solve for $\{s_j | j \in \mathcal{S}\}$: $\min_{s_j | j \in \mathcal{S}} ||\mathbf{x} - \sum_{j \in \mathcal{S}} \mathbf{a}_j s_j||$
    $\mathbf{r} \leftarrow \mathbf{x} - \mathbf{A}\mathbf{s}$
$\hat{\mathbf{s}} \leftarrow \mathbf{s}$
```

Bibliography

[1] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[2] Y. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Asilomar Conf. Signals, Systems and Computers*, 1993, pp. 40–44.

[3] D. L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck, "Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit," Stanford University, Tech. Rep., 2006.

[4] P. Paatero and U. Tapper, "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values," *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994.

[5] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.

[6] ——, "Algorithms for non-negative matrix factorization," in *Adv. Neural Information Processing Syst.*, vol. 13, Denver, 2001, pp. 556–562.

[7] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Processing*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[8] M. Aharon, M. Elad, and A. M. Bruckstein, "K-SVD and its non-negative variant for dictionary design," in *Proc. SPIE Conf. Wavelets*, vol. 5914, Jul. 2005, pp. 327–339.

[9] K. Engan, S. O. Aase, and J. H. Husoy, "Method of optimal directions for frame design," in *Proc. of the IEEE Conf. Acoustics, Speech, and Signal Processing*, vol. 5, Mar. 1999, pp. 2443–2446.

[10] C. Joder, S. Essid, and G. Richard, "Temporal integration for audio classification with application to musical instrument classification," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 17, no. 1, pp. 174–186, 2009.

[11] A. A. Wieczorkowska, J. Wróblewski, P. Synak, and D. Ślęzak, "Application of temporal descriptors to musical instrument sound recognition," *J. Intelligent Information Systems*, vol. 21, no. 1, pp. 71–93, 2003.

[12] A. Eronen and A. Klapuri, "Musical instrument recognition using cepstral coefficients and temporal features," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 2, 2000.

[13] S. Essid, G. Richard, and B. David, “Instrument recognition in polyphonic music based on automatic taxonomies,” *IEEE Trans. Audio, Speech, Language Processing*, vol. 14, no. 1, pp. 68–80, Jan. 2006.

[14] P. Smaragdis and J. C. Brown, “Non-negative matrix factorization for polyphonic music transcription,” in *Proc. IEEE Workshop on Appl. Signal Processing to Audio and Acoustics*, New Paltz, NY, Oct. 2003, pp. 177–180.

[15] S. A. Abdallah and M. D. Plumbley, “Unsupervised analysis of polyphonic music by sparse coding,” *IEEE Trans. Neural Networks*, vol. 17, no. 1, pp. 179–196, Jan. 2006.

[16] M. Schroeder, “Period histogram and product spectrum: New methods for fundamental-frequency measurement,” *J. Acoustical Society of America*, vol. 43, p. 829, 1968.

[17] M. Elad, Software: http://www.cs.technion.ac.il/~elad/software/.

[18] P. Smaragdis, M. Shashanka, B. Raj, and G. Mysore, “Probabilistic factorization of non-negative data with entropic co-occurrence constraints,” in *Proc. Int. Conf. Independent Component Analysis and Signal Separation*. Springer, 2009, pp. 330–337.

[19] A. Cichocki, R. Zdunek, and S.-i. Amari, *Csiszárs Divergences for Non-negative Matrix Factorization: Family of New Algorithms*, ser. Lecture Notes in Computer Science. Springer-Verlag, 2006, vol. 3889, ch. 5, pp. 32–39.

[20] F. Wang, T. Li, and C. Zhang, “Semi-supervised clustering via matrix factorization,” in *Proc. SIAM Conf. Data Mining*, 2008.

[21] T. Virtanen, “Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria,” *IEEE Trans. Audio, Speech, and Language Processing*, vol. 15, no. 3, pp. 1066–1074, Mar. 2007.

[22] A. Cichocki, R. Zdunek, and S.-i. Amari, “New algorithms for non-negative matrix factorization in applications to blind source separation,” in *IEEE Int. Conf. Acoustics, Speech and Signal Processing*, vol. 5, 2006.

[23] K. Miyamoto, H. Kameoka, T. Nishimoto, N. Ono, and S. Sagayama, “Harmonic-temporal-timbral clustering (httc) for the analysis of multi-instrument polyphonic music signals,” in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, 2008, pp. 113–116.

[24] O. Gillet and G. Richard, “Transcription and separation of drum signals from polyphonic music,” *IEEE Trans. Audio, Speech, and Language Processing*, vol. 16, no. 3, pp. 529–540, 2008.

[25] A. Klapuri, “Timbre modeling for the purpose of sound source recognition and separation in music,” *J. Acoustical Soc. America*, vol. 122, p. 2988, 2007.

[26] J. J. Aucouturier, F. Pachet, and M. Sandler, "The way it sounds: timbre models for analysis and retrieval of music signals," *IEEE Trans. Multimedia*, vol. 7, no. 6, pp. 1028–1035, 2005.

[27] M. Levy, M. Sandler, and M. Casey, "Extraction of high-level musical structure from audio data and its application to thumbnail generation," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, vol. 5, 2006, pp. 13–16.

[28] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Trans. Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[29] I. Kaminskyj and A. Materka, "Automatic source identification of monophonic musical instrument sounds," in *Proc. IEEE Int. Conf. Neural Networks*, 1995, pp. 189–194.

[30] K. Martin, "Sound-source recognition: A theory and computational model," Ph.D. dissertation, Massachusetts Inst. Tech., Jun. 1999.

[31] A. Eronen, "Automatic musical instrument recognition," Master's thesis, Tampere University of Technology, Oct. 2001.

[32] T. Kitahara, M. Goto, and H. Okuno, "Musical instrument identification based on f0-dependent multivariate normal distribution," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, 2003.

[33] P. Herrera-Boyer, G. Peeters, and S. Dubnov, "Automatic classification of musical instrument sounds," *J. New Music Research*, vol. 32, no. 1, pp. 3–21, 2003.

[34] B. Kostek, "Musical instrument classification and duet analysis employing music information retrieval techniques," *Proc. IEEE*, vol. 92, no. 4, pp. 712–729, Apr 2004.

[35] N. D. Chétry, "Computer models for musical instrument identification," Ph.D. dissertation, Queen Mary, University of London, Apr. 2006.

[36] E. Vincent and X. Rodet, "Instrument identification in solo and ensemble music using independent subspace analysis," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2004, pp. 576–581.

[37] S. Essid, G. Richard, and B. David, "Musical instrument recognition by pairwise classification strategies," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1401–1412, Jul. 2006.

[38] F. Sha and L. K. Saul, "Real-time pitch determination of one or more voices by nonnegative matrix factorization," in *Adv. Neural Information Processing Syst.*, Vancouver, 2004, pp. 1233–1240.

[39] S. Raczynski, N. Ono, and S. Sagayama, "Multipitch analysis with harmonic nonnegative matrix approximation," in *Proc. Int. Conf. Music Information Retrieval*, 2007, pp. 381–386.

[40] E. Vincent, N. Bertin, and R. Badeau, "Harmonic and inharmonic nonnegative matrix factorization for polyphonic pitch transcription," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 2008, pp. 109–112.

[41] N. Bertin, R. Badeau, and E. Vincent, "Enforcing harmonicity and smoothness in bayesian non-negative matrix factorization applied to polyphonic music transcription," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 538–549, Mar. 2010.

[42] S. K. Tjoa, M. C. Stamm, W. S. Lin, and K. J. R. Liu, "Harmonic variable-size dictionary learning for music source separation," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Dallas, TX, Mar. 2010, pp. 413–416.

[43] A. Holzapfel and Y. Stylianou, "Musical genre classification using nonnegative matrix factorization-based features," *IEEE Trans. Audio, Speech, Language Processing*, vol. 16, no. 2, pp. 424–434, Feb. 2008.

[44] C. Fevotte and S. J. Godsill, "A bayesian approach for blind separation of sparse sources," *IEEE Trans. Audio, Speech, Language Processing*, vol. 14, no. 6, pp. 2174–2188, Nov. 2006.

[45] T. Blumensath and M. Davies, "Sparse and shift-invariant representations of music," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 50–57, Jan. 2006.

[46] R. Lyon and S. Shamma, "Auditory representations of timbre and pitch," in *Auditory Computation*, H. L. Hawkins, Ed. Springer, 1996, ch. 6, pp. 221–270.

[47] S. Dubnov and X. Rodet, "Timbre recognition with combined stationary and temporal features," in *Int. Computer Music Conf.*, 1998.

[48] S. K. Tjoa and K. J. R. Liu, "Musical instrument recognition using biologically inspired filtering of temporal dictionary atoms," in *Proc. Int. Soc. Music Information Retrieval Conf.*, Utrecht, Netherlands, Aug. 2010, pp. 435–440.

[49] P. Herrera-Boyer, A. Klapuri, and M. Davy, *Signal Processing Methods for Music Transcription.* New York: Springer, 2006, ch. 6, pp. 163–200.

[50] F. Fuhrmann, M. Haro, and P. Herrera, "Scalability, generability, and temporal aspects in automatic recognition of predominant musical instruments in polyphonic music," in *Proc. Intl. Soc. Music Information Retrieval Conf.*, 2009, pp. 321–326.

[51] H. Hermansky and S. Sharma, "TRAPS - classifiers of temporal patterns," in *Int. Conf. Spoken Language Processing*, 1998.

[52] ——, "Temporal patterns (TRAPS) in ASR of noisy speech," in *Int. Conf. Acoustics, Speech, and Signal Processing*, 1999, pp. 289–292.

[53] M. Athineos and D. Ellis, "Frequency-domain linear prediction for temporal features," in *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2003, pp. 261–266.

[54] M. Athineos, H. Hermansky, and D. Ellis, "LP-TRAP: Linear predictive temporal patterns," in *Int. Conf. Spoken Language Processing*, 2004, pp. 1154–1157.

[55] N. Morgan, Q. Zhu, A. Stolcke, K. Sonmez, S. Sivadas, T. Shinozaki, M. Ostendorf, P. Jain, H. Hermansky, D. Ellis, G. Doddington, B. Chen, O. Cretin, H. Bourlard, and M. Athineos, "Pushing the envelope - aside [speech recognition]," *IEEE Signal Processing Mag.*, vol. 22, no. 5, pp. 81–88, 2005.

[56] M. Athineos and D. Ellis, "Autoregressive modeling of temporal envelopes," *IEEE Trans. Signal Processing*, vol. 55, no. 11, pp. 5237–5245, 2007.

[57] M. Elhilali, T. Chi, and S. A. Shamma, "A spectro-temporal modulation index (stmi) for assessment of speech intelligibility," *Speech Commun.*, vol. 41, pp. 331–348, Oct. 2003.

[58] P. Ru and S. A. Shamma, "Representation of musical timbre in the auditory cortex," *J. New Music Research*, vol. 26, no. 2, pp. 154–169, Jun. 1997.

[59] Y. Panagakis, C. Kotropoulos, and G. Arce, "Non-negative multilinear principal component analysis of auditory temporal modulations for music genre classification," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 576–588, Mar. 2010.

[60] T. Chi, P. Ru, and S. A. Shamma, "Multiresolution spectrotemporal analysis of complex sounds," *J. Acoustical Soc. America*, vol. 118, no. 2, pp. 887–906, Aug. 2005.

[61] N. Mesgarani, M. Slaney, and S. A. Shamma, "Discrimination of speech from nonspeech based on multiscale spectro-temporal modulations," *IEEE Trans. Audio, Speech, Language Processing*, vol. 14, no. 3, pp. 920–930, May 2006.

[62] M. D. Plumbley, "Conditions for nonnegative independent component analysis," *IEEE Signal Processing Lett.*, vol. 9, no. 6, pp. 177–180, Jun. 2002.

[63] M. Helén and T. Virtanen, "Separation of drums from polyphonic music using non-negative matrix factorization and support vector machine," in *Proc. EUSIPCO*, 2005.

[64] B. Gold and N. Morgan, *Speech and Audio Signal Processing.* Wiley, 2000.

[65] A. Klapuri and M. Davy, Eds., *Signal Processing Methods for Music Transcription.* New York: Springer, 2006.

[66] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," 2001–. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm

[67] E. Jones, T. Oliphant, P. Peterson *et al.*, "Scipy: Open source scientific tools for python," 2001–. [Online]. Available: http://www.scipy.org

[68] L. Fritts, "Musical instrument samples," Univ. Iowa Electronic Music Studios, 1997–. [Online]. Available: http://theremin.music.uiowa.edu/MIS.html

[69] F. Opolko and J. Wapnick, "Mcgill university master samples," McGill Univ., 1987.

[70] "Free sound samples – olpc," One Laptop per Child. [Online]. Available: http://wiki.laptop.org/go/Sound_samples

[71] "Freesound project," Music Technology Group, Univ. Pompeu Fabra. [Online]. Available: http://www.freesound.org

[72] R. Weiss and J. Bello, "Identifying repeated patterns in music using sparse convolutive non-negative matrix factorization," in *Proc. Int. Soc. Music Information Retrieval Conf.*, Utrecht, Netherlands, Aug. 2010, pp. 123–128.

[73] V. Tan and C. Févotte, "Automatic relevance determination in nonnegative matrix factorization," in *Proc. Signal Proccesing with Adaptive Sparse Structured Representations*, Saint-Malo, France, Apr. 2009.

[74] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent Data Analysis*, vol. 6, no. 5, pp. 429–449, 2002.

[75] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: Music genre database and musical instrument sound database," in *Proc. Int. Conf. Music Information Retrieval (ISMIR)*, Oct. 2003, pp. 229–230.

[76] P.-A. Manzagol, T. Bertin-Mahieux, and D. Eck, "On the use of sparse time-relative auditory codes for music," in *Proc. Intl. Soc. Music Information Retrieval Conf.*, 2008, pp. 603–608.

[77] E. Smith and M. S. Lewicki, "Efficient coding of time-relative structure using spikes," *Neural Computation*, vol. 17, no. 1, pp. 19–45, 2005.

[78] E. C. Smith and M. S. Lewicki, "Efficient auditory coding," *Nature*, vol. 439, pp. 978–982, 2006.

[79] N. Cho, Y. Shiu, and C. C. J. Kuo, "Efficient music representation with content adaptive dictionaries," in *Proc. IEEE Int. Symp. Circuits and Systems*, 2008, pp. 3254–3257.

[80] F. Cañadas Quesada, P. Vera-Candeas, N. Ruiz-Reyes, R. Mata-Campos, and J. Carabias-Orti, "Note-event detection in polyphonic musical signals based on harmonic matching pursuit and spectral smoothness," *J. New Music Research*, vol. 37, no. 3, pp. 167–183, 2008.

[81] J. Carabias-Orti, P. Vera-Candeas, F. Cañadas Quesada, and N. Ruiz-Reyes, "Music scene-adaptive harmonic dictionary for unsupervised note-event detection," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 473–486, Mar. 2010.

[82] S. K. Tjoa and K. J. R. Liu, "Multiplicative update rules for nonnegative matrix factorization with co-occurrence constraints," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Dallas, TX, Mar. 2010, pp. 449–452.

[83] S. Chen and D. L. Donoho, "Application of basis pursuit in spectrum estimation," in *IEEE Int. Conf. Acoustics, Speech, Signal Processing*, vol. 3, 1998, pp. 1865–1868.

[84] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1999.

[85] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Review*, vol. 43, no. 1, pp. 129–159, 2001.

[86] R. Gribonval and E. Bacry, "Harmonic decomposition of audio signals with matching pursuit," *IEEE Trans. Signal Processing*, vol. 51, no. 1, pp. 101–111, 2003.

[87] D. Donoho and J. Tanner, "Thresholds for the recovery of sparse solutions via l1 minimization," in *40th Annual Conf. Information Sciences and Systems*, Mar. 2006, pp. 202–206.

[88] M. Ryynänen and A. Klapuri, "Query by humming of midi and audio using locality sensitive hashing," in *IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 2008, pp. 2249–2252.

[89] Y. Yu, M. Crucianu, V. Oria, and E. Damiani, "Combining multi-probe histogram and order-statistics based lsh for scalable audio content retrieval," in *ACM Int. Conf. Multimedia*. ACM, 2010, pp. 381–390.

[90] C. Cotton and D. Ellis, "Finding similar acoustic events using matching pursuit and locality-sensitive hashing," in *IEEE Workshop Appl. Signal Proc. Audio and Acoustics*. IEEE, 2009, pp. 125–128.

[91] M. Casey and M. Slaney, "Song intersection by approximate nearest neighbor search," in *Proc. ISMIR*, vol. 6, 2006, pp. 144–149.

[92] ——, "Fast recognition of remixed music audio," in *IEEE Int. Conf. Acoustics, Speech and Signal Processing*, vol. 4. IEEE, Apr. 2007, pp. IV–1425–IV–1428.

[93] M. Casey, C. Rhodes, and M. Slaney, “Analysis of minimum distances in high-dimensional musical spaces,” *IEEE Trans. Audio, Speech, and Language Processing*, vol. 16, no. 5, pp. 1015–1028, 2008.

[94] J. Bentley, “Multidimensional binary search trees used for associative searching,” *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[95] R. Weber, H. Schek, and S. Blott, “A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces,” in *Proc. Int. Conf. Very Large Data Bases*, 1998, pp. 194–205.

[96] A. Gionis, P. Indyk, and R. Motwani, “Similarity search in high dimensions via hashing,” in *Proc. Int. Conf. Very Large Data Bases*, 1999, pp. 518–529.

[97] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, “Locality-sensitive hashing scheme based on p-stable distributions,” in *Proc. 20th Annual Symposium on Computational Geometry*, 2004, pp. 253–262.

[98] M. S. Charikar, “Similarity estimation techniques from rounding algorithms,” in *Proc. ACM Symp. Theory of Computing.* ACM, 2002, pp. 380–388.

[99] B. A. Olshausen and D. J. Field, “Emergence of simple-cell receptive field properties by learning a sparse code for natural images,” *Nature*, vol. 381, pp. 607–609, Jun. 1996.