Developing a Noise-Robust Beat Learning Algorithm for Music-Information Retrieval

A Thesis

Submitted to the Faculty

of

Drexel University

by

David Kurt Grunberg

in partial fulfillment of the

requirements for the degree

of

PhD in Department of Electrical and Computer Engineering

October 2014

Dedications

To my loving parents, whose support was invaluable on this journey.

Acknowledgements

I would like to thank my advisor, Dr. Youngmoo Kim, for all of the advice and support he has given me in the course of my studies. From the beginnings of my research career as an undergraduate SuperNOVA student through my final months as a PhD Candidate, Youngmoo has been supportive of me every step of the way. He provided all the tools and guidance that I needed in order to progress as far as I have, and I would not have made it as far without him. I benefitted immensely by working in his research group, and I am grateful that I had the opportunity to do so.

Next, I would like to thank my thesis committee. In addition to taking the time out of their busy schedules to evaluate and judge my thesis project, the members of this committee have greatly helped me to refine my final project. As such, I thank Drs. John Walsh and Yon Visell from Drexel University, Dr. Dan Lee from the University of Pennsylvania, and Dr. Paul Oh from the University of Nevada, Las Vegas. Dr. Oh in particular allowed me to start working with the Hubo robot as early as my freshman year of college, which in turn helped to set up my thesis work using that platform. This was an incredible opportunity, and I think him for it.

Several professors profoundly impacted me during my time at Drexel University. Dr. Timothy Kurzweg, one of my first electrical engineering professors, taught me how fascinating the field could be. Dr. Ben Taskar of the University of Pennsylvania showed me how machine learning algorithms could be of use to my research. Drexel's Dr. Thomas Houessou-Adin taught me how to craft technical documents. Finally, Dr. Jun-Ho Oh, of the Korean Advanced Institute for Science and Technology, gave me the chance to work in his robotics lab for months. I thank all of these professors.

My interest in math and science was kindled at my high school, the Illinois Mathematics and Science Academy. There I learned to love the process of learning thanks to the diligent work of the teachers at that institution. I would like to thank my

math teachers Drs. Donald Porzio and Steve Condie in particular, the former for helping me master calculus and go even further into the realm of mathematics, and the latter for his assistance with turning our math team into a force to be reckoned with. I would also like to thank Dr. Claiborne Skinner, who improved my writing ability dramatically; I still find myself checking my papers for overuse of 'to be' verbs and the passive voice. Lastly, I thank Mr. Michael Casey, for helping me to push my boundaries and learn in a way that few other teachers managed. Thank you all once again.

The support of my colleagues in the Music & Entertainment Technology Laboratory has been invaluable. As such, I thank Dr. Andrew McPherson, Dr. Erik Schmidt, Dr. Raymond Migneco, Dr. Travis Doll, Jeff Scott, Jeff Gregorio, Matthew Prockup, Brandon Morton, Alyssa Batula, Mike Caro, William Hilton, Alex Hrybyk, Brian Dolhansky, Matthew Zimmerman, and Manu Colacot. Dr. Schmidt, Matt, and Jeff Scott were particularly helpful as I progressed to this point. I would also like to thank Dr. Robert Ellenberg and Dr. Kiwon Sohn of the Drexel Autonomous Systems Laboratory for their assistance as I learned to use Hubo.

My life as a student involved studying abroad for long periods of time, but though it was sometimes hard to adapt to different cultures, my fellow travelers from Drexel were always there for me. I would thus like to thank RJ Gross, Clayton McNeil, Pareshkumar Bhrambhatt, and Thomas Wambold for their support.

I thank my oldest friends–Emily Morson, Nicole Nadeau, Nathan Sattler, and Chelsea Link–for always helping me to smile when it mattered most.

Finally, I conclude by thanking my family. Mom, Dad, I could never have done it without you. Thanks a million.

Table of Contents

List of Tables

List of Figures

Abstract

Developing a Noise-Robust Beat Learning Algorithm for Music-Information Retrieval

David Kurt Grunberg
Advisor: Youngmoo Edmund Kim, PhD

The field of Music-Information Retrieval (Music-IR) involves the development of algorithms that can analyze musical audio and extract various high-level musical features. Many such algorithms have been developed, and systems now exist that can reliably identify features such as beat locations, tempo, and rhythm from musical sources. These features in turn are used to assist in a variety of music-related tasks ranging from automatically creating playlists that match specified criteria to synchronizing various elements, such as computer graphics, with a performance. These Music-IR systems thus help humans to enjoy and interact with music.

While current systems for identifying beats in music are have found widespread utility, most of them have been developed on music that is relatively free of acoustic noise. Much of the music that humans listen to, though, is performed in noisy environments. People often enjoy music in crowded clubs and noisy rooms, but this music is much more challenging for Music-IR systems to analyze, and current beat trackers generally perform poorly on musical audio heard in such conditions. If our algorithms could accurately process this music, though, it would enable this music too to be used in applications such as automatic song selection, which are currently limited to music taken directly from professionally-produced digital files that have little acoustic noise. Noise-robust beat learning algorithms would also allow for additional types of performance augmentation which create noise and thus cannot be used with current algorithms. Such a system, for instance, could aid robots in performing synchronously with music, whereas current systems are generally unable to accurately

process audio heard in conjunction with noisy robot motors.

This work aims to present a new approach for learning beats and identifying both their temporal locations and their spectral characteristics for music recorded in the presence of noise. First, datasets of musical audio recorded in environments with multiple types of noise were collected and annotated. Noise sources used for these datasets included HVAC sounds from a room, chatter from a crowded bar, and fans and motor noises from a moving robot. Second, an algorithm for learning and locating musical beats was developed which incorporates signal processing and machine learning techniques such as Harmonic-Percussive Source Separation and Probabilistic Latent Component Analysis. A representation of the musical signal called the stacked spectrogram was also utilized in order to better represent the time-varying nature of the beats. Unlike many current systems, which assume that the beat locations will be correlated with some hand-crafted features, this system learns the beats directly from the acoustic signal. Finally, the algorithm was tested against several state-of-the-art beat trackers on the audio datasets. The resultant system was found to significantly outperform the state-of-the-art when evaluated on audio played in realistically noisy conditions.

1. Introduction

The field of Music-Information Retrieval (Music-IR) involves the development of algorithms that can analyze musical audio and extract various high-level musical features [36]. Many such algorithms have been developed, and systems now exist that can reliably identify features such as beat locations, tempo, and rhythm from musical sources. These features in turn are used to assist humans in performing a variety of music-related tasks. For example, humans often find it useful to select a group of songs (i.e., a *playlist*) from a much larger database according to specified criteria. Knowledge of beats and tempi can enable a system to automatically find music matching certain criteria, such as 'music faster than 140 beats per minute', without requiring that a human manually listen to and analyze all of the music. Another use for systems which detect these features is in the field of musicology; researchers seeking to analyze various aspects of music can make use of algorithms to automatically identify these features, again without having to listen to and annotate every piece of music themselves.

While current beat tracking algorithms are useful in a wide variety of situations, most of them were developed on relatively noiseless music. The music used to train and test beat trackers is generally taken directly from digital files that were produced in professional studios, and thus contain very little noise. For example, the Music Information Retrieval Evaluation eXchange (MIREX) has run a beat tracking competition for the last ten years, and is the largest competition of its kind in the world.[1] The most recent competition uses three datasets of audio files, but for every set, even one created specifically to be challenging to track, the music was taken directly from digital files recorded in professional studios [66, 90]. As a result, even state-of-the-art

[1] `http://www.music-ir.org/mirex/wiki/2013:Audio_Beat_Tracking`

beat trackers that are able to accurately process a wide variety of acoustically-clean audio were not developed for, and are less able to handle, audio heard in noisier environments.

A similar situation once existed in the field of automatic speech recognition (ASR). There, as in Music-IR today, systems were first developed for clean audio recorded in laboratory settings [29, 102]. These systems were able to produce good results on such audio, but were ill-suited for speech that was heard in conjunction with the various sorts of acoustic noise found in the outside world [82]. It took subsequent specialized research in denoising audio and developing noise-robust algorithms and features to improve the accuracy of ASR systems when operating on audio observed in noisier environments [128]. This work has been essential in making ASR systems useful in the wide variety of real-world situations that they are used in today, from speaking on a smart phone while on a noisy street to calling an automated service over a hissing phone line.

Similarly, Music-IR algorithms that could accurately locate musical beats even in noisy conditions would be very useful. Humans frequently listen to music in noisy environments, ranging from songs played over speakers in crowded bars and clubs to amateur bands recording in garages and other rooms that are not soundproofed. Such algorithms would enable us to perform various tasks, which can currently only be reliably performed on music that has little noise, on much more of the music that we humans enjoy. Playlist generation algorithms, for instance, could be used on this music to find more songs that match a user's specific criteria. Musicological researchers could more effectively study this music by using these algorithms to accurately determine their tempi and rhythmic patterns. Such systems would also make it possible to synchronize graphics or other computer-controlled elements with music played in noisy environments. In fact, noise-robust beat trackers could even allow

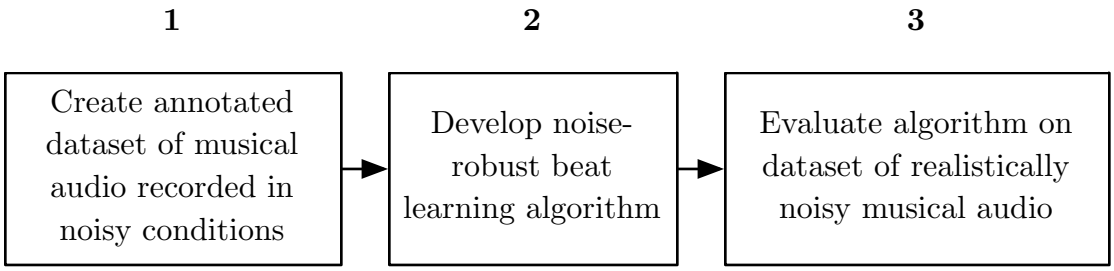

Figure 1.1: Contributions of this thesis.

robots to perform with and augment music, a task which is being explored by several groups [119, 131]. Current beat trackers can have difficulty tracking in the presence of robot noise, but a noise-robust tracker could allow for robots to move in time with recorded music or live human musicians, despite the presence of its own noise, and thus perform along with the music [64, 71].

This thesis contains three main contributions (Figure 1.1). The first is the collection and annotation of musical audio recorded in the presence of various noise sources. In order to evaluate Music-IR algorithms, systems are generally tested on audio for which ground-truth values are known [122]. A beat tracker, for example, might be evaluated on musical tracks for which experts have identified the positions of all of their beats. However, there are no publicly-available datasets with annotated noisy audio. The first series of tasks for this project thus involved collecting musical audio, recording it in environments which also contained various noise sources at specified acoustic levels, and obtaining reliable annotations. This data is not only of use for the other steps in this project, but can also be used by other researchers who are interested in designing algorithms that can operate in real-world acoustic conditions.

The second step is the development of a novel Music-IR beat learning algorithm that is robust to noise and is thus more useful overall for the determination of high-

level features in audio observed in real-world conditions. This algorithm utilizes powerful signal processing and machine learning techniques, such as Harmonic-Percussive Source Separation and Probabilistic Latent Component Analysis, which can automatically deconstruct an acoustic signal into components that can be useful for various Music-IR tasks [117]. While most algorithms for beat tracking use hand-crafted features in order to estimate the chances of a beat being located in any particular place, this system learns the spectral characteristics of the beats directly from the noisy audio. This more closely approximates how a human might listen to a completely new piece of music and pick out the beats, and allows for a more flexible system that does not make as many prior assumptions about what a beat 'should' sound like.

Lastly, the algorithm is evaluated on the annotated music in order to verify its robustness to noise. Because different types of noise can effect the system in different ways, results are presented from experiments that evaluated the beat learning algorithm's performance on music recorded in the presence of a variety of noise sources. Such sources included common sounds from indoor environments, such as HVAC systems and lighting hums, sounds common in crowded environments like bars and clubs which frequently contain music, such as people talking and plates clattering on tables, and noises that will be important for performance augmentation, such as the motors and fans of a moving robot. This system is also compared to multiple other state-of-the-art beat trackers in this step, in order to verify whether or not it is an improvement over existing systems. Ultimately, the experiments do show that the proposed work does outperform other state-of-the-art beat trackers on a wide variety of audio recorded in environments containing various types of noise.

2. Background

This work draws on research from a variety of fields which encompass Music Information Retrieval, machine learning, signal processing, robotics, and more. Three areas of research, however, are especially important to the proposed system. First, because this work involves designing a system that can learn beats and locate their positions in a musical signal (i.e., it performs the task of *beat tracking*), knowledge of existing beat tracking algorithms is incorporated in order to design a superior system. Second, work on designing noise-reduction and noise-robust algorithms is also taken into account in order to ensure that the proposed algorithm can accurately and efficiently operate even under noisy acoustic conditions. Finally, knowledge of algorithms designed for existing signal processing tasks that are traditionally performed in noisy environments, specifically automatic speech recognition and audio fingerprinting/Query-By-Analysis, is used in the final system. By drawing on prior work in beat tracking, general de-noising and noise-robust signal processing, and noise-robust Music-IR algorithms, the proposed system is more able to function accurately in noise than prior beat learning and beat tracking algorithms.

2.1 Beat Tracking

Knowledge of beat locations is a fundamental aspect of music analysis. A *beat* is defined as the basic unit of time of a piece of music, and thus provides the rhythmic structure and pulse for a musical piece. Beats can provide useful knowledge for segmenting a work temporally [28]. This knowledge can then be used in a wide variety of algorithms, including estimating the chord pattern of a work [10], segmenting the audio into meaningful chunks in order to develop 'thumbnails', or low-resolution

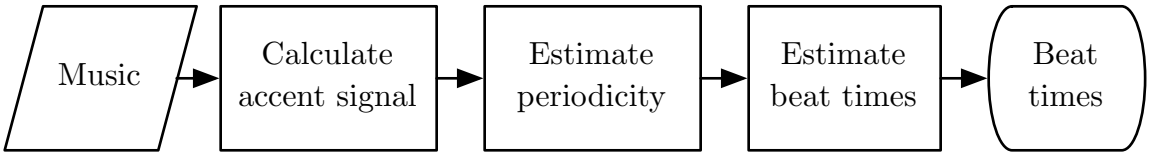

Figure 2.1: General algorithm for beat tracking.

representations, of the audio [8], and classification via rhythmic patterns [35]. Beat tracking also has particular relevance when enabling robots to perform music alongside humans, as knowledge of beats can help various systems maintain synchrony with the music, which in turn could be useful for automatic accompaniment of live musical performances [96, 134].

The majority of beat tracking algorithms have three major components (Figure 2.1) [75]. In the first step, the acoustic waveform is converted into a lower-dimensional representation, sometimes called an *accent signal*. This signal ideally has local extrema at beat locations. In the second step, the system analyzes the accent signal to estimate the periodicity, or *tempo*, of the music. Last, the system uses its knowledge of the accent signal and the tempo estimate to predict a sequence of locations that are likely to contain beats. Though most beat tracking systems share these three fundamental steps, the individual techniques used at each part of the system vary widely between algorithms.

2.1.1 Accent Signal Calculation

The first accent signals were discrete lists of potential beat locations, calculated by analyzing the temporal locations of notes in a musical piece (called *onset times*) and determining which onset times most likely corresponded to beats. Very early systems, such as the Longuet-Higgins algorithm, had to read the onset times off of piano rolls as there was no other reliable way to obtain them [86]. Some used heuristics, such

as 'the highest pitched notes in chord sequences are more likely than other notes to indicate beat locations', to estimate the usefulness of each onset [17, 109]. The most advanced of these looked for extrema of spectral features, such as the spectral power, to estimate which frames contained onsets that were most likely to correspond to beat locations [53, 55]. These systems were very restricted, as not all music was performed on piano roles or corresponded to the specified heuristics, and the discrete nature of the accent signals resulted in discarding information from frames not containing onsets.

Continuous accent signals were developed at the turn of the century, in conjunction with another important development: the splitting of the input acoustic signal into various subbands. This would allow subbands with beat information (such as drumbeats) to influence their systems more strongly. Important systems of this type include the work of Alghoniemy [2], which retained only low-frequency bands likely to contain drum beats, Dixon [32], which retained only high-frequency bands as these might vary more substantially on beats, and Scheirer [111], which analyzed several subbands independently to determine which ones had the most useful information. A related attempt to filter a signal into 'harmonic' and 'percussive' components and calculate an accent signal from the remainder was implemented by Alonso [4, 5]. These systems proved more robust than the prior implementations, not only because they were continuous and so could retain information from frames initially judged not to be onsets, but because they provided methods to isolate the most rhythmically important frequencies. Nonetheless, many of these systems still made assumptions about the audio they analyzed (such as that there would always be heavy drum content which was a good indicator of beat locations) that could damage their accuracy when operating on audio that did not conform to those assumptions [66].

At this point, the state-of-the-art systems all used continuous accent signals, and

many operated only on specific subbands, but they still used a wide variety of different features to actually generate the accent signal. A few large-scale tests were thus conducted to determine which features worked the best [33, 56, 74]. Several notable features included the energy in the high frequencies of a signal, the spectral centroid (a kind of 'center of mass' of an acoustic spectrogram), and the spectral flux (the compressed and differentiated frequency content of an acoustic signal). The most useful of all, though, was the complex spectral difference [56], which calculates a distance between the spectral content of a frame and the estimated content of that frame based on prior data. When the actual value in a spectral bin differs from the estimate by a large amount, a probable cause is a sudden beat at that location. As such, the complex spectral difference was found to be a good accent signal, at least for drum-heavy music in which its beat frames are markedly different from surrounding ones. Many of these features are still used in state-of-the-art systems today, e.g., the tracker developed by Oliveira et al [100]. However, the presence of noise can reduce the utility of these features, as the true values of the features based solely on the audio signal may not correlate with the calculated value of the features from the music-noise mixture.

Lastly, some groups have used preprocessing techniques to try to improve the audio or spectrogram before extracting the accent signal. Peeters et al, for example, calculated the *reassigned spectrogram*, in which the group delay and instantaneous frequency of each spectral bin was calculated and used to implement a temporal and frequency shift [106]. This incorporates the phase information of the signal into the magnitude spectrogram, thereby more accurately representing the original signal.

2.1.2 Periodicity Detection

As the first accent signals were discrete, the first periodicity (i.e., tempo) estimation systems simply calculated the intervals between subsequent note onsets [49]. Frequent intervals were strong candidates for being the tempo of the work. Heuristics were also often employed, such as searching for common rhythmic intervals [21]. When scores were available, systems could try to align the onsets to a score and then use the time it took to progress a certain distance to estimate the tempo [127]. These systems, however, were liable to break down if the onset estimation algorithm was imperfect, if beats did not fall on onsets (e.g., rests), or if the audio did not comport with the heuristics.

Techniques for estimating periodicity from a continuous signal were subsequently implemented. One of the most popular was using an autocorrelation function to sum a signal with shifted and reversed copies of itself; peaks in the autocorrelation were likely to occur when the shift was proportional to the signal's period [14]. This technique could also be used on small segments of a piece, instead of just the entire piece at once, which could allow for more accurate results in individual sections [38]. Another technique was to use comb filters, which also summed a signal with a shifted version of itself, and produced a strong response when the delay of the echo was proportional to the signal's period [111]. Comb filters were found to require more computation than autocorrelation operations, but were also more sensitive to multiples and fractions of the periodicity than autocorrelation [89]. Finally, as a period is the inverse of a frequency, Alonso et al developed a system to estimate the fundamental frequency of a musical piece, then inverted that frequency [4]. These systems, in general, outperformed the discrete systems of the past.

One common problem with the preceding periodicity estimation systems was that they could only make one choice for a tempo estimate; if that choice was wrong, they

could get 'stuck' and produce erroneous beat locations based on a misunderstanding of the true tempo. Allen et al were the first to design a multiple state system, which could store and use multiple tempo estimates, to reduce this risk [3]. Goto et al also found a multiple state system to be useful for this problem [55]. Another major breakthrough in this area was performed by Klapuri et al, who used a Hidden Markov Model (HMM) with different states corresponding to different tempo values [79]. HMMs allowed for the system to 'lock in' to a tempo estimate, but also to be able to change it if subsequent frames of the accent signal, which influenced the observation probabilities, did not comport with that estimate [28].

A relatively recent development for this task was the development of tempo 'thumbprints', low-dimensionality representations of tempos, for use in these algorithms. For example, pieces of audio with similar tempos might have similar autocorrelation functions [113], so the autocorrelations of audio with a certain tempo could be used as the thumbprints of that tempo. Eronen et al used such a system to achieve state-of-the-art accuracy for tempo estimation [44].

Lastly, the combination of multiple tempo estimators can be more useful than any individual algorithm. Kao et al developed that a system that uses a neural network to combine multiple periodicity estimations [75]. This system surpasses each individual estimator in terms of accuracy.

2.1.3 Beat Location Estimation

The earliest systems for estimating beat locations simply looked for onsets in the accent signal that were spaced according to the tempo estimate [86]. Heuristics, such as 'low frequency onsets are more likely to coincide with beats than high frequency onsets', were also used to attempt to better identify which onsets were significant [37], These systems, just like the similar ones used for tempo estimation, were vulner-

able to incorrect onset times and audio which did not obey the heuristic rules. The implementation of multi-state systems did help somewhat to alleviate these issues [3, 22, 54], although it could not remove them entirely.

A significant innovation in this area was formulated by Schierer, who determined that, when using comb filters to estimate the tempo, the delay states of the most resonant comb filters would be large during beat locations for many pieces of audio [111]. As such, the delay states of comb filters became a popular tool for beat location estimation [28, 79, 112]. These proved to be more robust than the simple systems that only looked for appropriately-spaced onset locations.

Another algorithm used at this time for beat location estimation was Dynamic Time Warping (DTW) [40]. Using DTW, the beat location estimation problem is formulated as a cost function in which high accent signal values reduce cost and deviations from the estimated tempo increase cost. DTW algorithms can then formulate a least-cost path through the audio. This algorithm is useful, although the weighting of how much the different components in the cost equation matter can be difficult to set in a general case.

Finally, a set of algorithms developed for this purpose calculate the distance of audio frames to other frames or signals, then use that as a metric for the likelihood of a beat being present. Foote et al, for example, determined that frames with beats would likely be more similar to each other than to frames without beats (and than frames without beats would be similar to each other), and so used distance metrics to calculate the distance between sets of frames, then marked minimally distant ones as beats [48]. Unfortunately, the similarity algorithm was found to not always be reliable. Peeters et al used Linear Discriminant Analysis (LDA) to make templates for beat and non-beat frames in a training set, then compared the onset signal frames of an input song to both templates [106]. Frames with a smaller distance to one class

were found to generally belong to that class, and this particular system achieved state-of-the-art levels of beat tracking in 2011.

2.2 Audio Processing in Noise

Many techniques have been developed for processing signals that contain noise. Some of these are useful in general cases for a wide variety of acoustic signals that need to be processed despite the presence of different types of noise sources. Others were developed for more specific types of signals, such as techniques to reconstruct missing spectral content that has been erased by noise. Finally, there are some techniques that are useful for specific types of noise, including the noise that robots can produce when they move. All three types of these techniques can provide insight as to how to make Music-IR algorithms more accurate in various types of noisy environments.

2.2.1 General Auditory Noise-Reduction Techniques

The earliest techniques for reducing acoustic noise were relatively simple compared to more recent systems. One prominent algorithm, the Wiener filter method, attempted to minimize the error between the true signal and its estimate when provided with estimates of the spectral characteristics of both the noise and the underlying signal [88].This system did require a reliable estimate of the signal's spectral characteristics, and also assumed that the noise was stationary and non-correlated with the signal, which was not always a safe assumption. Another historical technique, spectral subtraction, involved first estimating the noisy spectrum, then subtracting that estimate from the noisy auditory signal [12]. In this way, ideally, only the signal of interest would be left. The noise spectrum could also be amplified to better verify that the subtraction step was really getting rid of all the noise [11]. Spectral subtraction systems made the same assumptions about the noise (e.g., that it was stationary

and additive) as the Wiener filter systems.

Unfortunately, both Wiener filtering and spectral subtraction lead to a phenomenon known as *musical noise* [16]. This is because, after noise is reduced using the above methods, the frequency bands that once contained most of the noise now contained some spectral peaks (where the frequency bin happened to be very strong and was thus judged by the system to be part of the signal, not the noise) amidst attenuated or removed 'noisy' bins. The remaining noise thus contains frequencies that come and go rapidly depending on the original signal and the noise content. This results in noise that contains intermittent pure tones (where the signal energy or power at a given bin was strong enough to escape attenuation), hence the term 'musical noise.' Such noise is disconcerting and distracting, thus reducing the usefulness of spectral subtraction and Wiener filtering. Overestimating the noise strength and removing entire noisy bands can eliminate musical noise, but at the cost of throwing out any useful information in those bands – another undesirable outcome [16].

One powerful method to counter musical noise was proposed by Ephraim et al, and involves applying a gain factor to the spectral bins incorporating the estimated Signal to Noise Ratio (SNR) of the current time-frame as well as preceding ones [41, 42, 43]. The historical techniques generally suppressed a bin based only on the estimated noise content of each individual frame. Using the systems developed by Ephraim et al, strong attenuations would occur when the SNR of the prior frames was low, and weak attenuations when it was relatively high. The SNR of the current frame acted as a sort of correction factor, only significantly influencing the results when the SNR of the prior frames was already low. This helped prevent loud peaks from occurring in heavily-redacted areas, reducing musical noise. While this system was biased against sudden onsets, as large attenuations were set if the signal had been relatively quiet in preceding frames, a speech model that could estimate when speech

or other signals of importance were occurring and override the attenuation helped to reduce this problem [23].

These correction systems, while effective, require parameters (specifically relating to the characteristics of the noise) that must be carefully fine-tuned and input into the system [135]. This is inconvenient, and can result in a loss of utility if the noise parameters change during the system's use. Yu and Mallat proposed another system based on the block thresholding algorithm pioneered by Cai [15, 135]. This system divides the audio into blocks and then attenuates the blocks while trying to reduce a Stein risk estimate between the estimated signal and the signal the system is producing [121]. By continually recalculating the risk based on the observed signal, the parameters of the filter system can be updated on the fly [135]. This obviates the need to determine and enter them all by hand, and also allows for the system to adapt to changing signals. The resultant system was more robust than prior algorithms.

2.2.2 Missing Musical Audio Recovery

One problem that acoustic noise causes is that it obscures or erases important spectral information. To ameliorate this problem, some algorithms have been developed to reestimate and recover the content that has been lost. If this estimation is accurate, then the reconstructed audio is closer to the true signal than before, and Music-IR algorithms will likely function more accurately. Techniques developed for Missing Musical Audio Recovery algorithms thus have much to offer noise-robust Music-IR systems, because they have been verified to be able to extract useful information from musical signals despite the presence of noise.

Algorithms have been developed for this purpose that operate either on an entire musical signal or on small excerpts from one, but both sorts of system have significant problems. An example of a missing musical audio recovery algorithm that operates

on an entire signal is provided by Smaragdis et al, in which the authors use Singular Value Decomposition to estimate the values of the missing frames, then estimate them again based on those values until convergence [118]. LeRoux et al use a similar system that models the corrupted audio, forms a function based on that model, estimates the missing spectral bins, and continues until the system converges [80]. These sorts of systems are ill-equipped to deal with rare or sudden events and will be biased towards the 'average' value of the signal. Conversely, other algorithms examine the neighborhood of corrupted spectral bins, then try to find similar neighborhoods elsewhere in the signal without corruption, and use those values to estimate values in the bins containing the most noise [118]. These algorithms are better able to deal with unusual events (as long as they are not truly unique), but cannot exploit any underlying structure in the acoustic signal.

One option that combines benefits from both the local and global cases is Probabilistic Latent Component Analysis [117]. In this system, the auditory spectrogram is assumed to be drawn from a large number of partials. The partials are estimated using Expectation-Maximization (EM) based on the parts of the signal that do exist. Simultaneously, the probability of any particular missing value being formed from a weighted sum of particular partials is also estimated. As such, the entire signal is analyzed to find underlying structure, but each section of the audio can be comprised of different latent components, so more common components don't overwhelm rarer ones. Subjects reported that noisy audio processed using this algorithm sounded better and more complete than audio not reconstructed with this system.

2.2.3 Audio Processing with Robot Noise

Robots are becoming increasingly common in a variety of musical performances [119, 131]. To ensure that they can perform autonomously, though, the robots must be

able to incorporate acoustic feedback into their actions. For example, they might need to listen to other musicians to determine the current measure position within a piece. However, robots can produce noise that contaminates the acoustic environment, and their motor noises are unlike many other common sources of auditory noise [71]. Unlike other potential noise sources, this noise is not stationary, as it is based on motor parameters (such as position and velocity) which change over time. A robot may also produce roughly-periodic noise if it is moving in a specific repeating pattern, potentially resulting in a situation where the system begins to track the robot's noise and not the music. Dealing with this problem requires specialized algorithms.

One popular method for reducing robot noise is to use templates to model that noise and subtract it from the audio [98]. These templates can be found and entered into a library by recording the robot moving at different positions, using either the entire robot [68] or individual motors [72]. The templates converted into a model based on parameters of the robot, such as the velocity and position of its motors at a certain point in time [73] to avoid the problem of requiring enormous libraries [67]. These techniques have been found to improve Automatic Speech Recognition (ASR) accuracy onboard robots [98]. However, as with other types of spectral subtraction techniques, musical noise can be introduced to the system and cause further distortions [24, 71].

Another family of techniques exploit the known geometry of the robot and its sensors to extract the underlying acoustic signal from a noisy environment. Beamforming, which adjusts the phase and magnitude scalings of multiple microphones to generate constructive interference aimed at a signal source and destructive interference aimed at noise sources, is one such example [65, 126]. Noise occurring outside of destructive interference zones, though, can still contaminate the signal, and it can be difficult to remove all noise sources with this technique. Another option is Ge-

ometric Source Separation (GSS), in which the positions of microphones and other sensors are incorporated directly into a source separation algorithm [105]. GSS has been shown to be effective in reducing noise contamination, but it requires that all sensor positions be known exactly, and if one is moved even slightly accuracy can fall [98, 125].

Combinations of techniques have been shown to be more effective than individual algorithms on their own at reducing noise. Combining GSS with template subtraction, for instance, achieved better results than using either algorithm individually [69]. Combining template estimation with some algorithms designed to remove stationary noise also did better than using either type of technique on its own [70].

2.3 Traditional Tasks with Noisy Audio

Two research topics that traditionally use noisy audio are Automatic Speech Recognition (ASR) and Audio Fingerprinting/Query-By-Example. The task of ASR requires a computer to identify the most likely sequence of words that could form an acoustic utterance provided to it by a user [51]. While such systems were originally confined to clean environments, further research has developed ASR algorithms to the point where they can be used in very noisy areas [20]. Audio Fingerprinting/Query-By-Example is a task in which a computer, provided with an excerpt of an unlabeled song, must identify that song from a database. It was specifically designed so that the identification could be performed accurately even in noisy environments such as dance clubs and bars.

2.3.1 Automatic Speech Recognition

Automatic Speech Recognition (ASR) is a task that has been pursued by researchers for several decades (Figure 2.2) [110]. The first systems generally used one

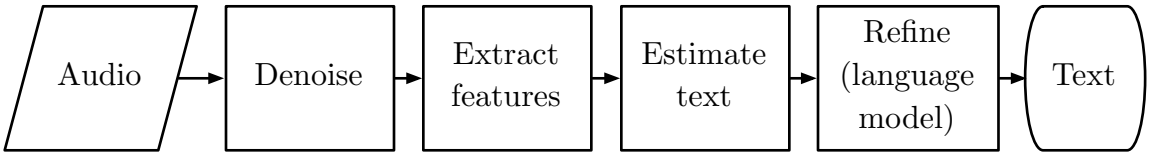

Figure 2.2: General algorithm for ASR. Not all elements, such as noise reduction and refinement via language model, will appear in all systems.

of two sets of features, both of which performed well on clean audio but not as well in noise. The first of these feature types was Linear Predictive Coefficients (LPCs), a compact representation of speech [102]. Systems using these features attempt to predict future values of a speech signal by weighting prior values with the LPCs. Speech, a signal often treated as a source produced by vocal chords and being excited by the 'filter' of the vocal track, can be well-modeled with LPCs [92]. The other type of feature was Mel Frequency Cepstral Coefficients (MFCCs), a representation of audio developed specifically for speech recognition [91]. To calculate MFCCs, the frequency content of an acoustic signal is perceptually warped to mimic the human ear and compressed before being passed through a Discrete Cosine Transform (DCT). Both LPCs and MFCCs obtained good initial results for speech recognition systems, which largely used clean, idealized audio [29, 102]. However, these features were sub-optimal for noisy environments. MFCCs were noted as being particularly weak in noise because of the compression step, which squashed the speech components down closer to the level of any background noise [124].

The classification algorithms for early ASR systems were also relatively simple. Dynamic Time Warping (DTW), in which one signal is compared to stretched and transposed versions of others, was a frequent choice [110]. This algorithm worked well because an utterance could be compared to many others in a dictionary, and the DTW would account for the test utterance being spoken more quickly or slowly than the

dictionary examples. HMMs were also frequently used [51], often in conjunction with Gaussian Mixture Models (GMMs) [6]. However, though these systems worked with clean audio tested in the same conditions as the training set, accuracy fell sharply when noisy audio was used or when the testing utterances were different in tone (e.g., whispered instead of spoken) from the training set [82]. In order to improve robustness, the models were sometimes trained on multiple *styles* of speech, such as words shouted, whispered, and spoken normally, but noise remained a significant problem [83]

Some groups used standard noise-reduction algorithms to pre-process noisy audio for better ASR. One common choice was spectral subtraction, as this removes the noisy frequency bins [12]. Another was Minimum Mean Squared Error (MMSE), which attempts to model the probability distribution of the noisy audio signal, then to estimate the values of the underlying clean speech signal based on a weighted sum of linear prediction from the noisy signal's distribution [51]. A few different methods for estimating the noisy signal's distribution were proposed, some requiring stereo recording [31] and others operating on monophonic audio [95]. These techniques were successful in raising the accuracy of ASR systems in noisy environments [51].

Language models were also used to refine the results from ASR systems [97]. These are models in which rules of a language are used to try to massage the text determined by an ASR system into something more plausible than the initial output. For instance, if in a language two words or syllables almost never follow each other, a model can take an ASR output in which those two elements are next to each other, determine that this is unlikely, and look for a better alternative. Such systems can improve accuracy, though they introduce the risk that the model could itself be inapplicable to certain rule-breaking utterances and so could damage performance.

Other researchers chose to select features which were naturally robust to noise

in the first place. Even humans have difficulty with speech recognition using noisy MFCCs, which indicated to some that MFCCs might be inherently inadequate for this problem [107]. One proposed option was to use spectral peaks, the most energetic spectral bins of a signal, as features because these bins were likely to still be the most influential ones even in the presence of noise [103]. Mel-Filter Bank energies, the energies in the frequencies that make up each individual Mel-scale band, were also found to be robust to noise [120]. In this way, features that could remain useful for ASR even in the presence of noise were identified.

Advanced machine learning techniques have also been used in an attempt to design systems powerful enough to overcome noise. Neural nets, for example, were found to provide some robustness to noisy environments. [30, 123]. Parallel Model Combination (PMC), in which multiple HMMs are used to represent both clean and noisy speech, also achieved good results in noise [50]. Finally, Deep Belief Networks (DBNs), a sophisticated machine learning algorithm, were used to learn features directly from an acoustic signal that could overcome noise contamination [128]. In several recent studies, a system using DBN-learned features surpassed GMMs on a variety of datasets [26, 94]. These datasets included both clean audio such as the traditional TIMIT corpus, a historic database containing 630 speakers each stating ten sentences and used in numerous research studies [45, 52, 114], and noisy audio such as a business dataset taken from users speaking into their phones [1]. Another study showed that a system using DBN learned features had a Word Error Rate of about 24% less than one using MFCCs on a set of clean and noisy utterances [128].

2.3.2 Audio Fingerprinting and Query-By-Example

A typical query-by-example algorithm involves two or three steps (Figure 2.3) [19]. First, the algorithm converts a provided musical excerpt into a 'fingerprint', a

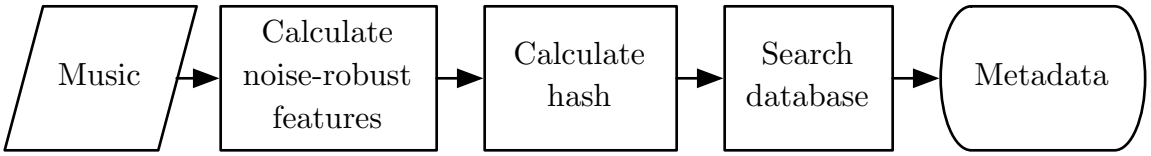

Figure 2.3: General algorithm for query-by-example.

low-dimensional representation of the music that is, ideally, unique to that particular piece, and is usually some kind of hash. Second, the fingerprint is compared to a database of audio fingerprints, and the closest matches are identified. Additional tests will sometimes be run on the n-closest candidates to find the truly closest match. Because these databases are very large, the fingerprints are designed to be very fast to search.

A hash function is simply a function that maps one sequence of data into another sequence, with the latter preferably more compact and of a fixed length. In order to be useful for query-by-analysis problems, these hashes must also be quickly searchable, as this will allow the algorithm to move through an entire database in a matter of moments [129]. The user can then quickly be updated with the metadata about the music they are listening to. If the hashes are binary, as they generally are for this problem, then the system can also easily compute a distance between the hash descriptor of the input audio and that of the music in the database. The Hamming metric is a popular choice for calculating the distance [129]:

$$H(a,b) = \Sigma_{i=1}^{L}[a(i) = b(i)] \tag{2.1}$$

Where H is the Hamming distance, a and b are two strings of equal length L, and [and] are Iverson brackets. Using this metric, not only can hashes that exactly match the input hash be found, but, if the user is willing to trade off some speed for

increased accuracy, hashes which are missed by a certain number of characters can also be returned. The result is a set of candidates that is vastly smaller than the entire database, which can then be analyzed further to help ensure that the correct song, and moment within that song, is chosen. By using hashes, only a very small subset of the database must be thoroughly analyzed, which saves on computation time and helps keep such systems usable by people who want quick query returns.

Several different features have been proposed for fingerprinting algorithms. One popular algorithm converts audio to a wavelet representation, then directly calculates a hash using the Min-Hash technique [7, 13]. Another uses Haar wavelets [129] as inputs to set about 25,000 feature candidates, including differentiated spectral power and the difference between successive strongest frequencies [76]. Adaboost is then used to identify which of the features is the most useful, and the signs of the features (without the magnitudes) is hashed. Shazaam, an app for mobile devices that can identify audio in a variety of real-world situations, uses spectral peaks as the basis for its feature, on the basis that these peaks are likely to remain such even if the audio contains noise [130]. The peaks are paired with their nearest neighbors, and coordinates describing the pairs are saved for each song. All three sets of feature representations allow for efficient searching of large datasets.

3. Preliminary Work

Over the past several years, I have developed a variety of algorithms for Music-IR on clean audio. While these systems were not designed to function accurately on noisy audio, the techniques they used to analyze a variety of musical signals and estimate useful high-level features helped lay the foundation of this thesis. Similarly, I also conducted work involving enabling humanoid robots to contribute to musical performances, such as by dancing or by playing a musical instrument. The discoveries made during these projects further helped to provide useful building blocks for the development of systems which could accurately process audio, even in the presence of robot noise, and could further help enable those robots to respond to music.

3.1 Music-IR on Clean Audio

I have worked on multiple Music-IR tasks in a variety of applications. For example, I developed an algorithm that enabled a computer to follow an orchestral performance and note its position in the score [108]. This relates to the task of *audio-to-score alignment*, another task studied in the Music-IR community [132]. Such systems can be used for a variety of purposes, and the one that I was working on was enabling a mobile application to deliver real-time updates to a user's mobile device to help them understand and better enjoy a performance. By allowing the system to know the position of the orchestra down to the measure, updates could be sent to help alert the listener to what they had just heard or were about to hear. As such, I designed a system that not only aligned the audio to the score accurately, but did it causally and quickly enough that it could operate in real-time on a live musical signal. The resultant system is being used in a mobile application by the Philadelphia Orchestra

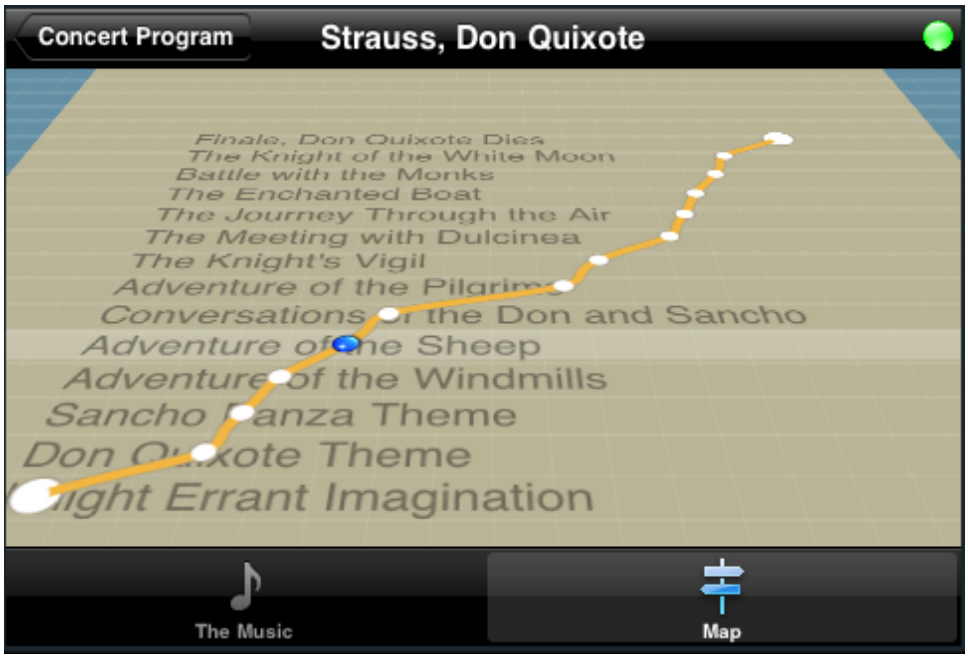

Figure 3.1: Mobile application using the score tracker algorithm to determine the position of an orchestra in a musical performance.

to help guide listeners through their concerts (Figure 3.1).

While this particular work did not need to be robust to noise, it did demonstrate the importance of a system that can operate on music obtained over a potentially noisy acoustic channel. The nature of this system requires it to operate on live audio just as it is being produced; there is no time for it to be recorded and cleaned up in post-production. As such, for this system to function, it must be capable of dealing with any incidental noises that occur during the performance. In this particular case, the environment was a concert hall that was relatively quiet except for the music, and so the system did not need to take noise into account. However, similar systems for score tracking, or other Music-IR tasks, that are put to use in noisier locations such as bars, clubs, and outdoor venues, also must operate on music that cannot first be cleaned of noise in a studio environment. As such, to ensure accurate performance,

Music-IR systems used in these situations must be robust to noise.

I also worked on several beat tracking systems. As the eventual goal of these was for use with robots responding to live music, I focused on making them causal and fast enough to run in real-time, but they still operated under traditional Music-IR conditions (e.g., using clean audio taken directly from digital files) [39, 62]. The beat trackers I designed, which utilized subband spectral energy as the accent signal, autocorrelation and comb filtering for tempo estimation, and cost functions and comb filterbank delay states for determining beat locations, worked well enough to enable robots to dance to prerecorded music [61].

While these beat trackers were successfully able to locate the beats in certain types of audio, they made assumptions about the characteristics of the beats that limited their utility. For example, my system that utilized subband spectral energy as the onset signal made the assumption that a new beat will always have a large amount of energy in at least one subband, and that the relevant subbands had preset central frequencies and bandwidths. A better option would be to design a system that could learn the spectral profiles of beats after being provided with a few seconds of audio. Not only would this allow for more information about the beats, such as their spectral characteristics, to be observed, but it could also potentially improve accuracy by allowing the system to adaptively determine what it should be looking for based on a particular musical piece.

3.2 Robot Motion

The research team that I am part of, the Music & Entertainment Technology Laboratory (MET-Lab), has been studying how to enable robots to participate in musical performances. This could allow robots to perform in musical ensembles alongside humans, which in turn could inspire new musical performances and could also allow

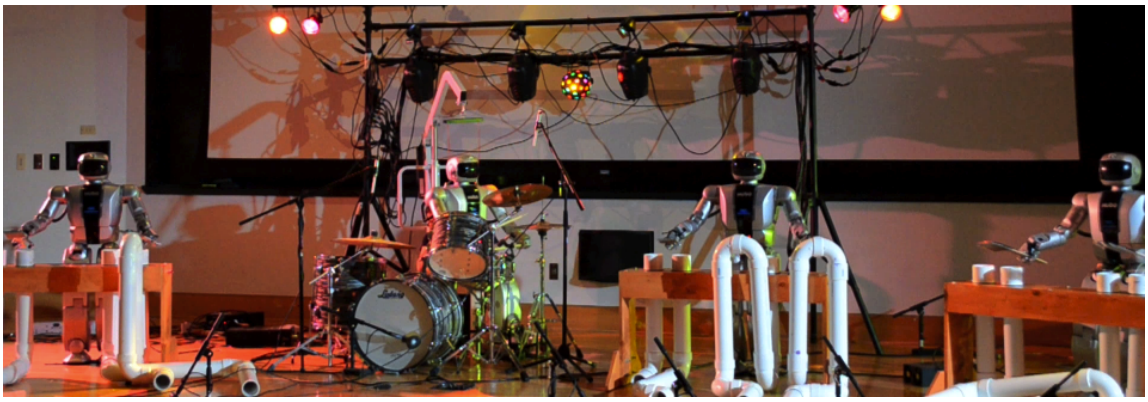

Figure 3.2: Arrangement of 4 Hubo robots using a drum set and pitched pipes to cover 'Come Together' by The Beatles.

for additional study of various aspects of performance environments. For example, a robot could be made to play in different ways, and the reactions of the crowd or the other musicians could be monitored. As such, I have participated in several projects involving teaching robots to dance or play musical instruments [77, 78].

One such project involved using a set of four Hubo robots to perform a cover version of 'Come Together' by the Beatles (Figure 3.2) [58].[1] The robots were enabled to perform several gestures in order to strike pitched pipes and nonpitched drums, then controlled to play specific gestures at certain times in order to perform the required piece. In this way, we enabled the robots to perform in a musical manner, opening the door to using them in other performances for which the ability to play musical notes could be useful.

While this project was successful, it also highlighted the importance of utilizing Music-IR in conjunction with robotic musical performances. Because the robots could not hear themselves or react even to their own noises, human operators had to continually adjust them by hand in order to obtain the desired sounds. Even when

[1] https://www.youtube.com/watch?v=UMQLX-aw_dc

a robot missed a note entirely, it could not hear the problem and correct it, so the performance had to be suspended while the robot was manually shifted in order so that it could strike its pipes correctly. Furthermore, the robots were all producing large quantities of noise during this activity, indicating that any Music-IR system used for this task would have to be noise-robust.

I also worked on enabling robots to perform dance gestures [60]. I designed gestures that could be parameterized according to the emotional content of the music in order that the robots could respond sensibly to a variety of music. Parameterized values indicating emotional qualities, such as 'Arousal' to specify how intense an emotion was, could be passed into the robot and used to set the motion. This would allow, for example, a motion to be sharper and more violent in an 'angry' song, or peaceful and more expansive in a 'calm' one. This work was validated by human subjects and was found to produce motions that 'matched' the intended emotion.

As with the Beatles performance, the work on this project emphasized how using Music-IR in scenarios like this could be helpful. Because our various Music-IR algorithms were not yet noise robust, the robots could not determine from live music which gestures to produce or when to produce them. Gestures were instead chosen by predetermining or manually selecting the types of gestures that would be used and when they would begin and end. While such a system could be made to appear synchronized with music by means of dead-reckoning, it would be unable to adjust to any changes in the music, and so was of limited utility. These gestures required noise-robust Music-IR algorithms in order to be utilized to their full potential.

3.3 Music-IR with Robots

I have studied methods for detecting emotion specifically for use in robot performance systems [59]. This work emphasized a fast, causal system that could reliably

estimate emotion for use in parameterizing the dance motions of several robots. The requirement of using multiple robots also forced me to keep the system generalizable by using a Music-IR algorithm that was not specifically constrained to one particular robot (by, for example, forming a noise template based on the particular sounds produced by that robot's parameters and using that particular template to redact noise sources) but could be used in conjunction with all of them. Human users found that, when the robots analyzed the music for emotional content and then selected the gesture with the appropriate mood, the performances appeared to be congruent with the music [59].

While this work was another step towards using Music-IR to enable robots to interact with music, it still had limitations. Because the algorithms were not yet noise-robust, the music had to be piped directly into the robot digitally, which required additional operator time to set up and maintain. Additionally, as the human listeners were hearing the audio through the acoustic channel, they were hearing a subtly different sound than the robots, which heard the music in its clean form. As such, there may have been a slight disconnect between what the humans and the robots heard, which may have negatively affected the human perception of the robot performance. Noise-robust Music-IR systems would alleviate these problems.

Finally, I have worked on a project that used both auditory and haptic feedback in order to help a performance robot learn whether or not it is playing an instrument correctly (Figure 3.3) [9]. It is often difficult and time-consuming to manually calibrate robots, and they can sometimes begin missing notes even in the middle of performances. It would be useful for such systems to be able to calibrate themselves, and as a first step, such robots must be able to determine whether or not they are playing their instruments successfully. This work therefore involved having the robot strike at pipes many times, sometimes hitting and sometimes missing, and using the

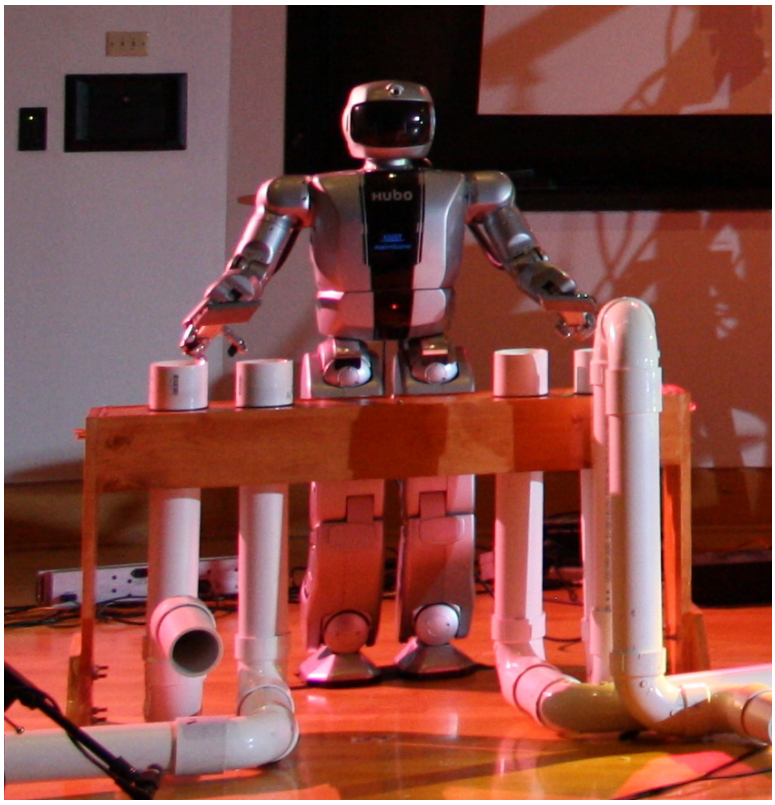

Figure 3.3: Hubo set up to play pitched pipes. Calibration can be performed with the help of an algorithm that incorporates auditory and haptic feedback, obtained from microphones and force sensors in the robot's wrists, respectively.

resultant auditory and haptic feedback in a classification system so the robot could better determine whether or not any given strike successfully hit the note. Accuracy was over 98%, indicating the utility of the system.

This system helped us progress towards making our robots capable of performing with music, but further noise robustness could improve it further. The setup that we used involved mounting a microphone very near the pitched pipes, picking up the sound produced by the performance with a minimum of other noise sources. It might not always be possible to mount microphones to the instruments that will be used in any given performance, however, and if the system could be run only using

the robot's onboard microphones, however, it would be more flexible;. Enabling the system to be more noise-robust would allow for the microphones to be placed directly on the robot, where, though they would be further from the sound and closer to noise sources such as the robot's motors, would be more convenient for the performers and the operators.

4. Compilation of Annotated Audio Datasets

The first component of this thesis project is a large database of annotated musical audio. The requirements for this database are that it must contain a large amount of music, both with and without various types of noise, and must be annotated with ground-truth values for the Music-IR high-level features that I am interested in, specifically beat locations. Additionally, it should be spread over a variety of genres and artists to avoid biasing my algorithms in any particular direction. I therefore collected one hundred and twenty songs from CDs and various digital archives, re-recorded this audio in the presence of various noise sources, and created ground-truth annotations that I could use to compare with the results from my algorithms.

4.1 Audio Collections

I first obtained clean audio from a variety of digital archives and CDs. This music was divided into two collections. The first was made up of music with strong, heavy beats at regular intervals. This collection was designed such that its music tracks were easy for conventional beat trackers to analyze when there was no noise added to their signals, and would thus be useful in analyzing the effects of noise on the trackers, since any performance degradation could be attributed to the noise. The second collection varied more widely, with music from ten different genres and by over ninety different artists, in order to ensure that the database had enough diversity that the results would be meaningful.

4.1.1 Dance Collection

I first created a collection of music tracks that could be analyzed accurately by existing beat trackers so long as it remained relatively free of noise. For example, music with prominent, consistent beats are often easier for beat tracking algorithms to analyze than music with changing tempos, unusual rhythms, or subtler, less obvious beats. Such music would allow me to obtain useful baselines for my work. I could use this dataset to determine, for example, the raw effects of noise on this music; since state-of-the-art beat trackers would process the music very accurately without noise, any subsequent performance degradation would be solely attributable to the noise. This dataset could therefore help quantify exactly how bad noise degradation was. More difficult music, by contrast, might track poorly either due to the noise or the complexities in the music, and the resultant ambiguity would render such music less useful for this purpose. Music with consistent, heavy beats is also of use in attempts to enable robots to respond to music, since the simpler nature of the music can in turn be served by relatively simple and intuitive responses.

In order to obtain this music, I examined dance music as well as songs from other popular genres. Dance music often has strong, heavy beats which are easy for beat tracking systems to pick up. Using CDs and online resources, I searched for music that maintained a constant tempo, was set in a common time signature, and lacked sudden rhythmic shifts. In order to ensure that my system was not biased towards any particular tempo, I looked for music spanning the range of 80 beats per minute (bpm) to 160 beats per minute, within which 95% of popular music falls [93]. Lastly, I made sure to find music from a variety of artists and albums to avoid biasing my dataset towards any one particular musician or recording. This would all help ensure that the resulting audio files, while easy to beat track when no noise was added to them, were varied enough that the results obtained by running my Music-IR algorithms on

these songs would be meaningful.

Ultimately, I came up with a collection of 20 songs that fit all of my criteria. These songs are listed in Table B.1. The slowest of these songs has a tempo of 89 bpm and the tempo of the fastest is 152 bpm, thereby representing a wide rhythmic range of music. Also, these songs are from 18 artists in total, so no one artist overwhelms and biases the dataset. In total, this collection comprises 1 hour, 19 minutes, and 33 seconds of music that can be used for this thesis project.

4.1.2 General Collection

While the Dance Collection would help verify the problem of noisy audio, using only such audio could result in a system that is biased towards that particular type of music. Dance and pop music with steady, prominent beats is indeed likely to occur frequently in bars, clubs, and other venues in which noisy music is heard, but other types of music are played in such environments as well, and dance songs with varying tempos, changing beats, unusual rhythms, or other aspects that make them hard to track are also a possibility. In order to validate my system, another collection of audio was needed, one that was not necessarily easy to track even without adding additional noise, but that was varied enough that results obtained using that audio could reasonably be said to be generally applicable. Such a collection also must be varied in terms of artists and albums as well, so as to prevent an overabundance of musicby one particular artist, for example, of biasing the resultant beat learning algorithm.

As a result, I identified ten genres that encompass a wide variety of music. These genres are: classical, country/western, dance, folk, hip-hop, jazz, metal, pop, rock, and soul. These genres were based on those commonly found within the beat tracking literature [100]. Within each genre, I located a list of the most influential, popular, or

important songs developed by reputable groups such as VH-1 and MTV (Table B.2). I then searched digital archives to find music tracks from these lists. By utilizing lists curated by experts and hundreds of fans, I ensured that my music selections would be truly representative of these genres, and reduced the possibility that the music collection would instead be biased by my own tastes and the music that I personally was familiar with.

In the end I obtained 100 tracks, 10 from each of the 10 genres that I wanted to consider. These songs, along with the lists that indicated the importance of each one, are detailed in Table B.3. I then obtained one-minute excerpts from each song in order to make a collection with 100 minutes of audio in total. Unlike the Dance Collection, much of this music does not have steady, consistent beats. Some of the songs features tempo changes, some are not in common time, and some use different instruments and notes to mark the beat, which can confuse conventional beat tracking systems. The result was a collection that spanned a wide range of music and thus allowed for a more general determination of how the proposed algorithm can perform when provided with noisy musical audio.

In total, the Dance and General Datasets contain 2 hours, 59 minutes, and 33 seconds worth of music for use in this thesis project.

4.2 Adding Noise to the Dataset

The work performed in this thesis is focused on using Music-IR on musical audio that contains noise produced by various sources, and is thus lower in acoustic quality than the clean audio that is usually found on professionally-produced CDs and digital files. In order for my audio collections to be useful for this project, the audio files within them had to include specific types of acoustic noise. I therefore obtained audio recorded in the presence of noise sources including HVAC systems and lighting hums,

the chatter found in bars and clubs, and even computer sounds and motor noises produced by robots.

Creating a dataset of noisy audio cannot simply be accomplished by taking a clean musical signal and digitally mixing it with a recording of noise, because such a mix would neglect any distortion caused by the speaker equipment that played the audio, or the microphones that allowed the system to hear it. It would also disregard the effects that the room itself, with echoes and other nonlinear frequency response, might have on the music. Instead, to obtain datasets with realistic noise, I played the audio over a speaker system in a noisy environment, then recorded the resultant acoustic signal on microphones for processing. I added three types of noise to the dataset in this manner: room noise, bar noise, and robot noise.

4.2.1 Room Noise

One type of noise that will occur in almost any live environment is the sounds of the room itself. Heating, Venting, and Air Conditioning (HVAC) systems, hums from lights, opening and closing sounds from doors, and other sounds that naturally occur in enclosed rooms can contaminate the acoustic signal and reduce the accuracy of conventional Music-IR algorithms. Music produced in studio environments does not suffer as much from these effects, because the environmental sounds can be carefully controlled and sometimes removed by sophisticated post-processing techniques. However, music heard in other environments, such as bars, is not guaranteed to be free of that type of noise, and so Music-IR algorithms must be able to cope with it.

In order to incorporate this variety of noise into my dataset, I recorded my audio datasets in the Expressive and Creative Interaction Technology (ExCITe) Center's workshop area, a large space in which researchers and students from various departments collaborate on multi-disciplinary tasks (Figure 4.1). Inherent noise sources in

Figure 4.1: Photograph of the ExCITe Research Center workshop room. Noise sources include lights in the ceiling, ventilation ducts, and electronic locks on the doors which make a clicking sound when unlocking.

this environment include sounds from the HVAC ducts positioned throughout the Center, the clicking of the locks in the electronic doors, and the hums from the lights mounted in the ceiling. By playing and recording my audio collections in this room, I could ensure that the audio would contain noises produced by those sources. This would then allow me to test my system against audio containing basic environmental noises.

I performed one round of recordings using only the room noises. In this step, I played the music through a set of Creative Inspire T10 computer speakers, standard computer speakers which were judged to be suitable in terms of quality for this particular experiment. I also used an iRig MIC Cast in order to record the audio

Figure 4.2: iRig MIC Cast.[1]

(Figure 4.2). This is a small microphone specifically designed for mobile devices, and is a common choice for recording audio in commercial environments such as bars, making it a suitable choice for this project. It also has a relatively flat frequency response, with only 3 dB of variation between 100 Hz and 15 kHz, ensuring that the microphone itself will not add unwanted distortion to the system.[2] To ensure that the recording took full advantage of the dynamic range of the speakers without clipping, the music was first normalized such that the maximum amplitude within each excerpt was set to 1. In order to determine the overall strength of both the noise and the music signals, I used a Realistic Category Number 33-2050 Sound Level Meter in order to record the sound pressure levels (SPL). When no music was playing, the base SPL of the room was about 74 dB. When music played through the speakers, the SPL ranged between 78 and 80 dB. In this way, I recorded audio such that the music was louder than the noise, but not by so much that the noise was drowned out.

[1] `http://www.ikmultimedia.com/products/irigmiccast/`
[2] Ibid

Because this particular noise was inseparable from the room, when I recorded datasets in the presence of other noises, room noises were picked up too. However, because the other noises were much louder, the room noises had the largest contribution to this particular dataset.

4.2.2 Bar Noise

One goal of this thesis project is to enable Music-IR algorithms to operate accurately on music heard in noisy venues such as bars, clubs, and dance halls. As such, it is crucial that the noises from such institutions should be added to the dataset so that the algorithms can be tested on musical audio which also contains those particular interfering sound sources. For obvious reasons, actually playing and recording three hours of audio within a real bar would be impractical; in addition to annoying the other guests and staff, the bar might be playing music of its own which could potentially overpower the music that I want to record. Furthermore, a live recording of that nature would make it much harder for me to take additional datasets if I so chose, because I would have to return to the bar and hope that the noise was sufficiently similar in character to that of the previous recording. Instead, I made a recording of the sounds in a local bar, and then played those sounds back when I performed another recording of my music datasets.

The venue I chose was the Landmark Americana, a restaurant and bar located in the city of Philadelphia (Figure 4.3). I entered the bar in the early evening on a Friday night; this is a popular time for people to visit Landmark Americana, and the venue was accordingly loud and raucous. I then sat at the bar and recorded approximately forty minutes worth of audio using the iRig Microphone Cast, as it is a common choice for recording sounds in public environments such as bars. The noises recorded included people talking, plates and silverware clattering against each

Figure 4.3: Landmark Americana.

other, and the scraping of bar stools against the floor. In this way I was able to obtain a good selection of the sounds that might be heard in any conventional bar.

I subsequently performed another noisy recording in the laboratory. I once again used the Creative Inspire T10 computer speakers to play the music and recorded on an iRig. To play the noise, I used an Apple iPod Hi-Fi A1121 speaker, with the frequency response depicted in Figure 4.4. The human vocal range is between about 300 and 3400 Hz, and this speaker has a strong response in those ranges, ensuring that the voices in the bar noise (which I observed to the most prominent components of the noise upon listening to my recording) would be performed at suitable strength [18]. This recording setup ensured that both the music and the noise could be heard

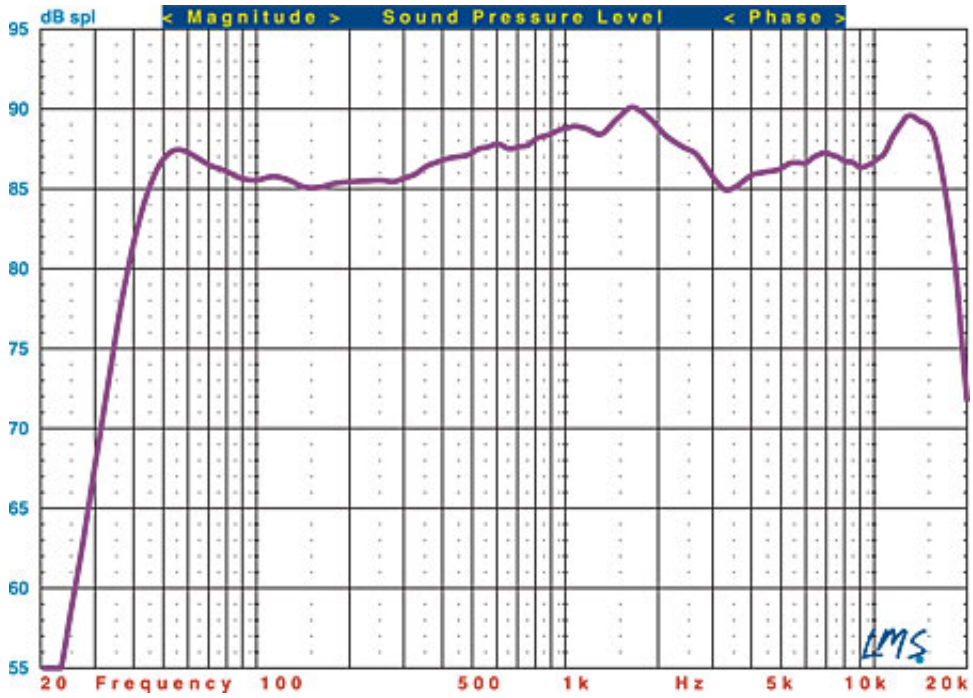

Figure 4.4: Frequency response of Apple iPod Hi-Fi speaker.[3]

and recorded together to create a noisy signal.

The base SPL for the bar noise signal played in the laboratory was 86 dB, and the base audio level was about 90-92 dB. The SPL of the two played together extended between 90 and 92 dB, depending on the particular piece being played. The music and the noise alike were both audible in the final recordings, mimicking the environment of a bar in which music may be heard at the same time as people talk, eat, and otherwise produce sound.

4.2.3 Robot Noise

As one of the objectives of this project is to design Music-IR algorithms that could be used in conjunction with musical robots in order to enable those robots to

[3] http://www.soundandvision.com/content/apple-ipod-hi-fi-ht-labs-measures

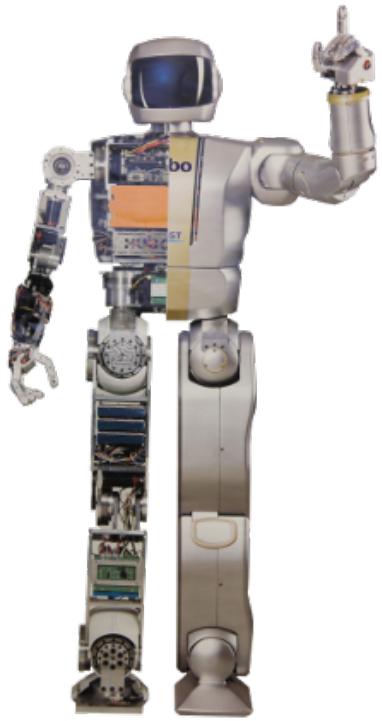

Figure 4.5: Hubo+, with its shell off (left half) and on (right half) [85].

understand and intelligently respond to music, it was important to include the noises and sounds produced by a robot in the dataset. In order to most closely simulate actual usage conditions, I selected a versatile humanoid robot called the Hubo+ (Figure 4.5) [81, 104]. I specifically selected a humanoid because many instruments and dance styles are designed for the human body, and a humanoid robot would be more likely to be able to perform with a given instrument or in a given style than another robot. While a specialized robot may be better able to perform in one particular way, such as a dedicated drumming robot with many sensors and motors devoted specifically to producing the motions and feedback required to play the drums, its results might not be generalizable to other types of performances [131]. Hubo is also big enough to play several types of instruments and possesses enough degrees of freedom to actuate them, thereby allowing it to perform without

requiring the development of custom instruments or modification of the robot, and has been used in prior music performance tasks [9]. I thus judged Hubo to be a suitable platform for this project. Additional specifications for Hubo are detailed in Appendix A.

As with the other two noise sources, I played music through a set of speakers and recorded them over microphones in order to add robot noise. However, due to the nature of the robot, the particulars of the microphones changed from the preceding steps. My initial experiments with enabling Hubo to process live audio involved using external microphones mounted around Hubo's head and connected to an external computer [64]. A problem with this setup, though, was that it required several pieces of additional hardware, such as microphones, their accompanying stands, computers, and offboard mixers, which was inconvenient for performances. I found that the wires not only got in the way of the external equipment, but that the robot could accidentally pull on these wires while moving and potentially cause damage either to itself or the other equipment. The robot's motion was also restricted by the externally-mounted microphones, since it could not move without leaving the sensors behind. Another significant concern was that the system was vulnerable to deviations in its physical setup, such as the height of a microphone stand being adjusted slightly from day to day, that could result in inconsistencies from recording to recording and potentially affect the sound in a non-reproducable manner. Lastly, I found that using offboard equipment both allowed for lag to creep into the system and made it possible that the different computers would have slightly desynchronized clocks, which could result in misalignment between the computers controlling the robot and the ones connected to the microphones. As such, I determined that an onboard approach would be more suitable when recording audio that was to have noisy audio added to it. It was also a better approximation of how the system might be set up in a real

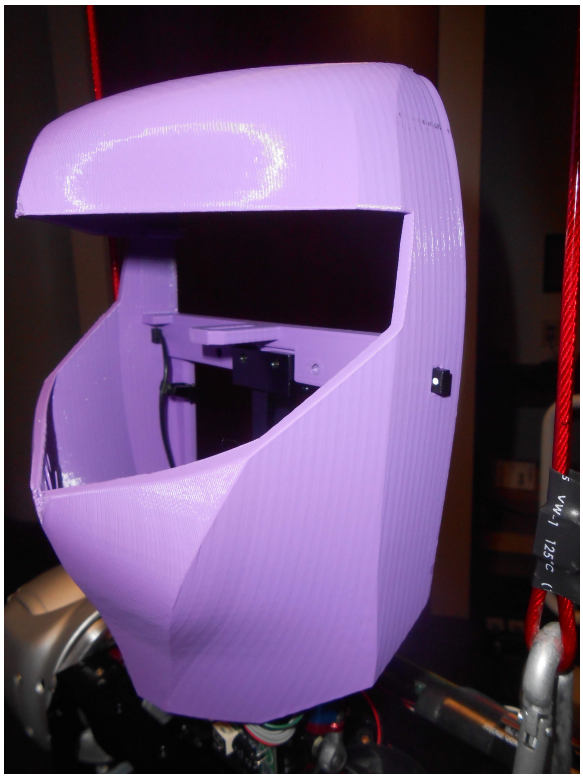

Figure 4.6: Image of Hubo's head with Shure SM93 microphones.

performance environment.

As such, the final setup for Hubo's recording environment was designed such that everything could take place through Hubo's computer, with no external equipment necessary. I mounted two lapel microphones on Hubo's 'head' shell, in approximately the positions of human ears on a human head (Figure 4.6). The position of the microphones was chosen because human ears provide sufficient sensory information for the extraction of high-level features (e.g., beat locations) from noisy musical signals, and so a similar setup on a robot should in theory provide sufficient information for these same tasks. This head was machined in a 3D printer and was created with slots specifically designed for the lapel microphones, as well as for other sensor equipment such as cameras. This allowed me to fix the microphones in specific positions, so that

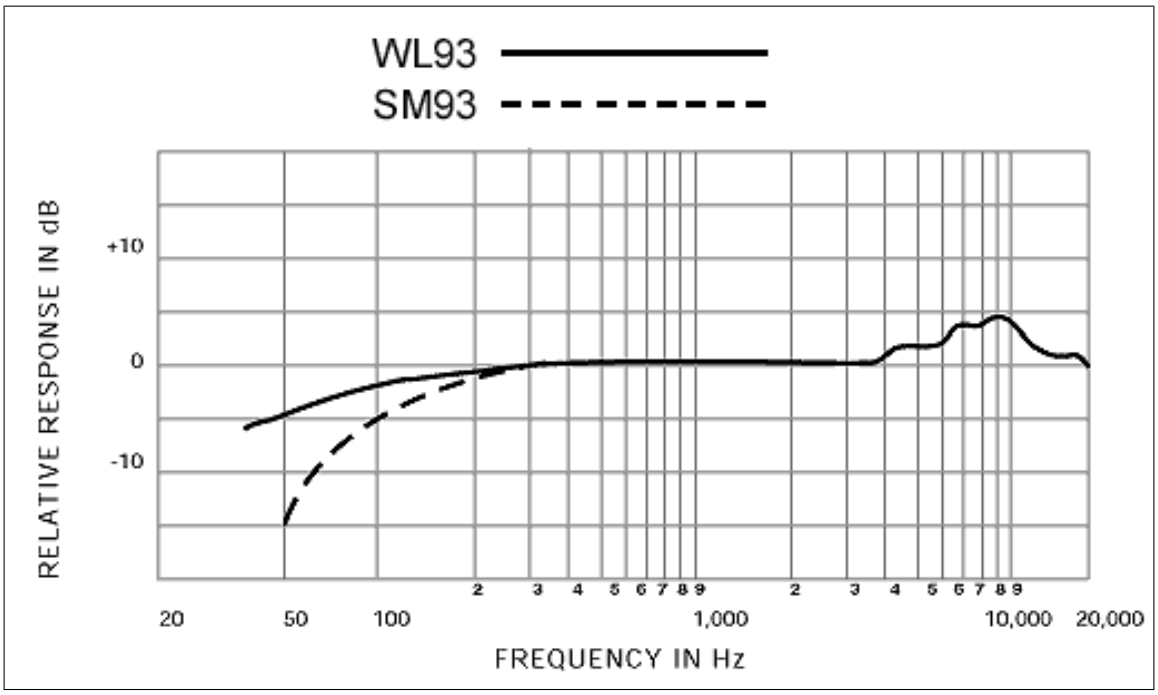

Figure 4.7: Frequency response of Shure SM93 microphone.[4]

even if the microphones are removed, they can be returned later to exactly the same position. The microphones can also send audio directly to Hubo's computer, so while it is still possible to use external devices for this purpose, it is no longer required.

The acoustic profile of the Hubo's lapel microphones is shown in Figure 4.7. As the frequency response plot shows, the Shure SM93 microphone can pick up frequencies over almost the entire range of human hearing, from 20Hz to 20kHz. Additionally, there is a flat profile in the 3kHz - 9kHz range [87], which encompasses a wide range of frequencies for which the response is mostly constant. This frequency response demonstrates that these microphones are suitable for my purposes. I also opted to use the Apple Hi-Fi A1121 in order to play the music. The scenario being simulated was a performance environment in which music is played for a robot to listen to

[4] http://www.shureasia.com/products/microphones/sm93

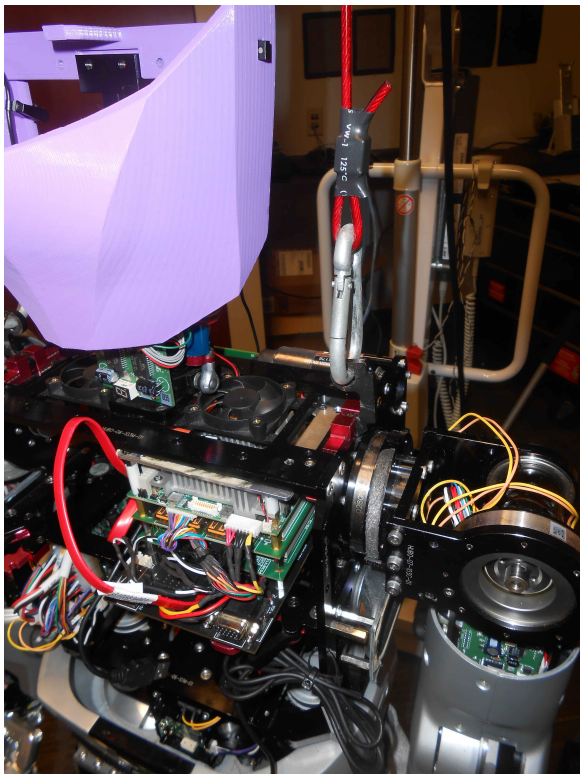

Figure 4.8: Noise sources near Hubo's left microphone, including fan, computer, and exposed motor.

and respond to, and in that situation a higher-quality speaker than the Creative Inspire T10 computer speakers would be likely to be used. The Hi-Fi, with its strong frequency responses across a wide range of frequencies, particularly bass frequencies where beats are often found, was found to be suitable for this purpose.

The robot also required some configuration in order to produce all of the noises that I desired to record. Some of the Hubo's noise sources are active simply by activating the robot, such as the fans that prevent the robot's computers from overheating and the hum that the motors produce when they are turned on whether or not they are moving (Figure 4.8). These I could trust to be present for the microphones to record no matter what the robot did. However, I did not want to neglect

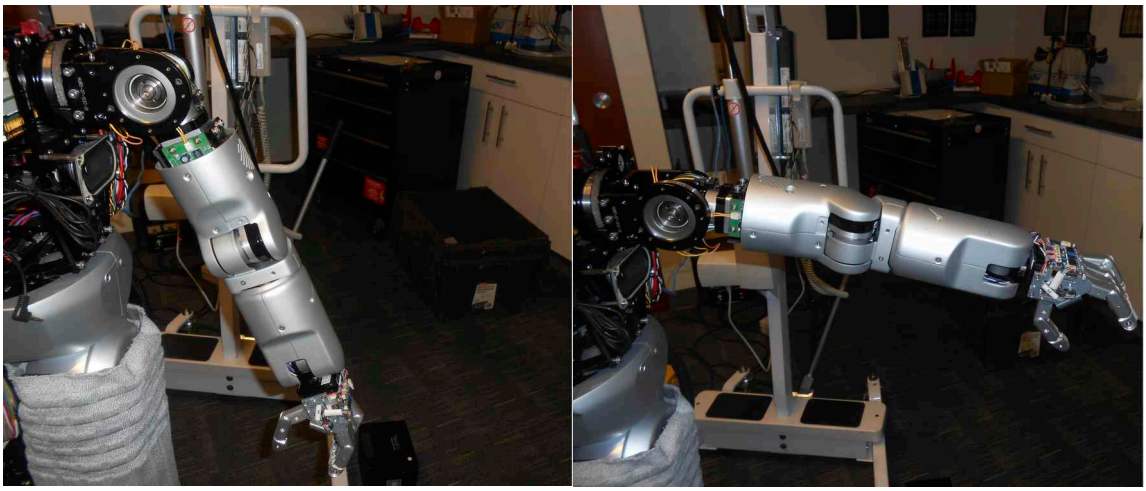

Figure 4.9: Hubo left arm at the nadir and apex of its motion.

another robot noise source that I was interested in, specifically, the noises produced by the robot's motors. Motor noise is different from most other noise sources at issue because it is intermittent, nonstationary, and in certain situations can be roughly periodic [64, 71]. Because the robot is not constantly moving in the exact same way, the sounds that the motors make will vary from frame to frame. These sounds are thus not stationary, and cannot be dealt with in the same manner that, for instance, a constant hum at 60Hz could be accounted for. These sounds can also become roughly periodic, such as if the robot repeats a gesture at a consistent speed. In the worst case, if the robot begins moving periodically but at a different period than the music (such as if the music changes suddenly and the robot does not adjust itself), the Music-IR algorithms may begin tracking the robot's motion instead of the music, resulting in drastically reduced accuracy. This is a problem that must be overcome to enable sensible response to audio by robotic systems. Accordingly, it is vital that such systems be tested on audio that contains this particular type of noise.

I therefore created a periodic motion that the robot would perform while recording

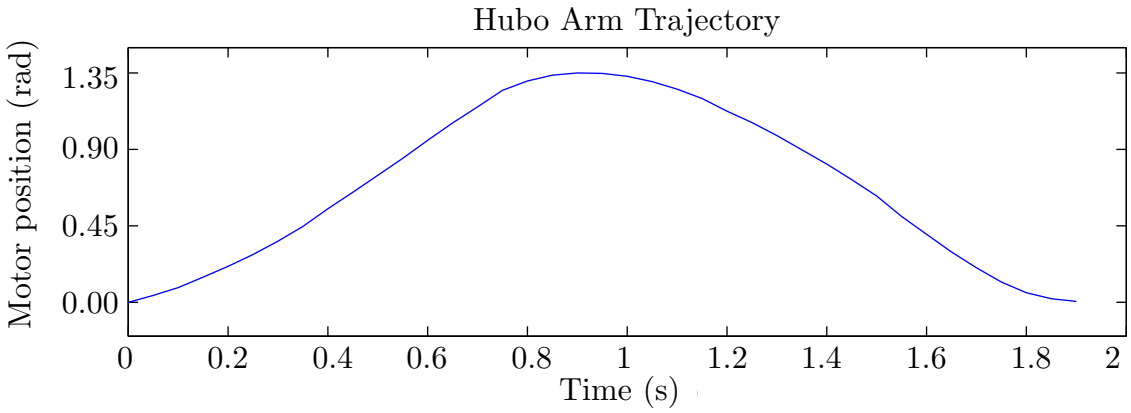

Figure 4.10: Position of Hubo's shoulder roll motors over one instance of the gesture.

the audio. During this motion, the arms of the robot move up and down .7rad, moving in a smooth and fluid manner. Images of the robot's arms at different times in this gesture are shown in Figure 4.9, and the trajectory of the robot's shoulder roll motors is shown in Figure 4.10. This motion, produced by shoulder motors very near Hubo's microphones, produced an audible noise source. This allowed for the audio to be recorded in the presence of nonstationary, roughly periodic noise that was not linked to the audio. A spectrogram of audio recorded by the Hubo's microphones while the robot performed its motion is shown in the bottom half of Figure 4.11, with the arm rising from .3s to 1.1s and falling from 1.4s to 2.2s. The spectral effects of the noise are clearly visible in the spectrogram.

In order to more fully evaluate the effects of the robot's noise, two recordings were taken. For the first, the robot did not move, and the music played through the speaker and was recorded on its microphones along with sounds from the fans, computers, and other components. For the second, the robot was directed to perform the rising and falling motion while the music was played and recorded (Figure 4.12). In order to ensure that the robot's motor and computer noises would be audible to the

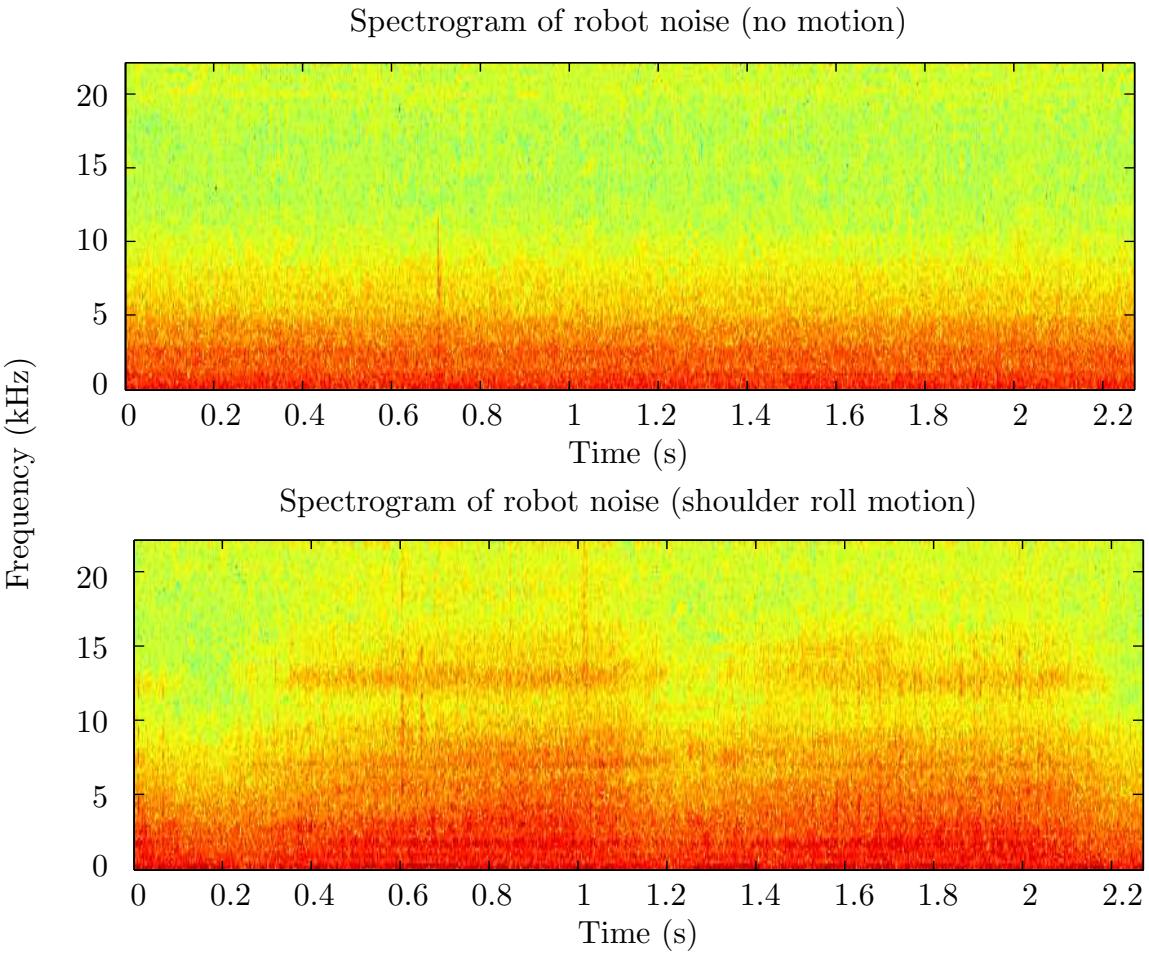

Figure 4.11: Spectrogram recorded in the laboratory on Hubo's microphones. The robot is making no motion in the top spectrogram, and is making the motion indicated in Figure 4.10 in the bottom spectrogram.

microphones, sections of the robot's shell were removed, as in Figure 4.12. The SPL recorded by the microphones when the robot was on but immobile was 66 dB, and once the arms were moving the SPL increased to between 68 and 72 dB, depending on the exact position of the arms at a given moment. The SPL when music was added on top of that was between 72 and 73 dB.

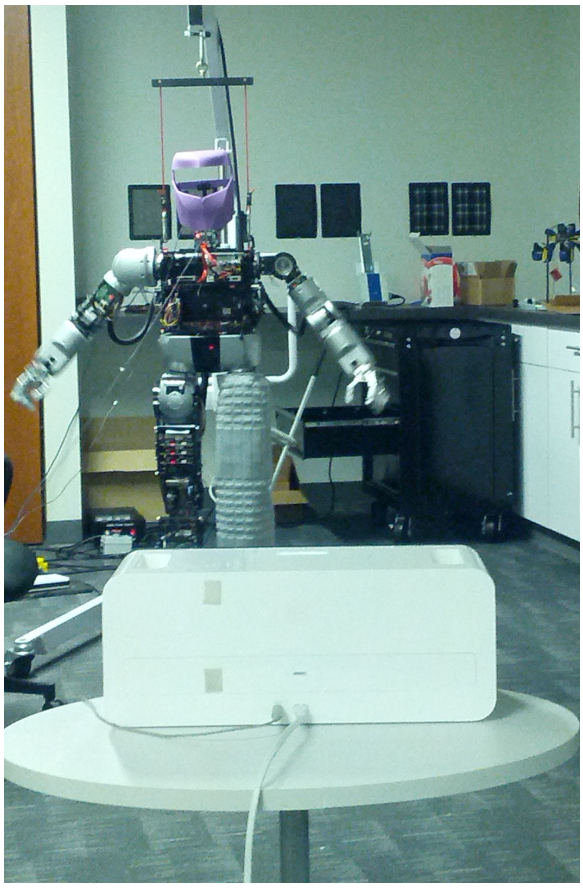

Figure 4.12: Hubo and speaker positioned for recording.

4.3 Annotating Audio

I obtained ground-truth data for all of the music in my audio collections relating to the specific Music-IR task of beat tracking. I did this in order that I could evaluate the accuracy of state-of-the-art systems designed to perform this task, both with and without my noise-mitigation techniques, on that audio. I chose the beat tracking task both because it is significant, as knowledge of beat, rhythm, and tempo is important when analyzing or working with musical audio, and also because it is comparatively simple to determine an accurate ground truth for as compared with several other common Music-IR tasks. For instance, it is difficult to annotate the emotional content

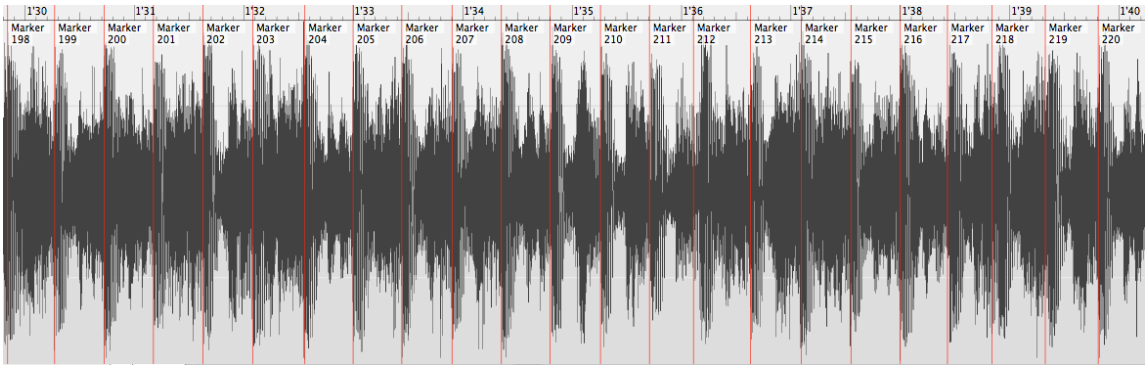

Figure 4.13: Beats marked on an excerpt of the song 'Moskau' by Deschingus Khan, part of the Dance Dataset.

of a musical work, because different people can have very different opinions as to how a particular work makes them feel. Multiple annotations from multiple people must be collected and then combined in some way, and uncertainty can creep into the underlying ground-truth data (e.g., if one annotator has a very different response than everyone else, is that legitimate, or were they not paying attention?) Genre is hard to annotate for the same reason; one particular song could be classified, for instance, as rock, pop, or country depending on the individual annotator. It is much easier to agree on beat locations, by contrast, as while there might be a few possible sets of beats in a piece (such as if one annotator determines the tempo to be at twice or half the value of another annotator), there are still vastly fewer possibilities, which makes annotation both simpler and more reliable. More reliable annotations in turn allow for more confidence in the results, since the system will be compared against a ground truth that I am certain of, as opposed to a ground truth that may or may not be truly representative of the music.

Beats were indicated using the SoundStudio program. SoundStudio conveniently allows for users to mark flags in a musical file at the touch of a button (Figure 4.13), and these flags can be read in to MATLAB, Python, and other programs. By pressing

the flag button during each beat as the music plays, the marked locations can be used as beat times, allowing for the collection of ground truth information in an intuitive manner. Once done, I went back over the files and shifted beat locations by hand where needed in order to ensure that the markers were positioned as close to the actual beat location as possible. This procedure was performed for the entirety of every song in my Dance Collection and my General Collection, using the noise-free audio files I obtained from CDs and online digital archives.

In order to translate these annotations onto the noisy audio files, I went through those songs and marked a single location in each one. The recordings taken with the robot audio were performed with some breaks in the recording process to allow the robot to reset its motors and begin its sequence of gestures again, so a location (generally an early beat in the piece) was marked for each song. The other datasets were taken in one continuous stream, so one location was marked for the entire set. The corresponding locations in the clean audio were found, and the shift calculated. The beat locations from the clean audio were then shifted appropriately and copied to the noisy audio, resulting in ground truths for all of the datasets. The annotated audio files for the Dance and General Collections were denoted as the Dance and General Datasets, respectively.

5. Baseline Performance of State-of-the-Art Systems on Noisy Audio

The inability of state-of-the-art Music-IR systems to adequately cope with noisy musical audio is a serious problem in the field. Much of the audio that humans listen to is noisy, ranging from music heard over speakers in public venues such as bars and clubs to music heard in the presence of noisy machines. Furthermore, the effect of this noise on the performance of conventional Music-IR algorithms is indeed substantial. Even in situations where human listeners are easily able to listen to the music and still identify the desired high-level features, such as beat locations or tempi, automated systems can often fail. Lastly, prepending these systems with algorithms designed to remove the noise from the audio is not necessarily an adequate solution; in addition to such algorithms having difficulty dealing with certain types of noise, such as robot ego noise, they can also add distortion to the audio which can make it harder for the Music-IR algorithms to perform accurately.

In order to precisely determine the effects of noise on the specific task of beat tracking, I tested several state-of-the-art beat tracking algorithms on the music tracks within the clean and noisy datasets of the Dance Dataset. Because this music was chosen such that conventional beat trackers could accurately process it in noise-free situations, analyzing their accuracies when processing the same audio with added noise allowed me to measure the deleterious effects of sounds such as robot motors and fans. I also built my own algorithm that used a noise-reduction preprocessing system in order to investigate if the procedure of first removing the noise, then proceeding with a conventional Music-IR algorithm, would be sufficient to accurately beat track the noisy audio.

5.1 State-of-the-Art Beat Tracking Algorithms

A large number of beat tracking algorithms have been developed by various institutions and research groups. In order to select suitable candidates as state-of-the-art trackers, I referred to a recent comparison of sixteen prominent algorithms in the field [66]. This comparison study used novel and challenging audio datasets in order to separate out the weaker beat trackers from the stronger, and culminated in selecting five high-performing trackers for use in a multiple-tracker system. Three of the five selected trackers are publicly available, and I chose to use these three systems as my state-of-the-art systems for the purposes of determining the baseline accuracy of current trackers on noisy audio.

Most beat trackers include three major components: accent signal calculation, periodicity estimation, and determination of beat locations [75]. The accent signal calculation step involves computing one or more features in each frame of the audio; these features ideally have local extrema at beat locations and do not have local extrema anywhere else. These features, which taken over a portion of the music are called an *accent signal*, are then passed through a periodicity detection system which attempts to find the period, or tempo, of the music based on those features. Both the period estimate and the accent signal are then used to estimate beat locations, which ideally are positioned near extrema of the accent signal and are spaced at the period calculated in the second step. All three of the state-of-the-art systems, as well as the one that I devised, follow this basic formula.

The first of the state-of-the-art systems used for establishing a baseline was developed by Ellis et al and will be denoted *Ellis* here [40]. This system begins by warping the magnitude spectrogram of the musical audio to the Mel scale and then performing additional perceptual filtering. The Mel scale is a scale in which the frequencies are 'warped' according to how the human ear perceives sound; frequencies

that sound closer together to the human ear are closer together on the Mel scale. Because humans are able to perceive auditory beats, the process of warping the magnitude spectrogram of the audio signal into a space that mimics human perception can help to better better find elements, such as beats, that humans hear. Subsequently, the first-order difference in the warped magnitude spectrogram is calculated for multiple subbands; this difference is likely to be larger when new events, such as beats, occur. The differences are then summed to form an accent signal.

The periodicity detection portion of the Ellis algorithm is an autocorrelation operation. Autocorrelation sums a signal with a delayed and reversed echo of itself; it will likely produce constructive interference and a large output if the signal is shifted by a multiple of its period, so the output of the autocorrelation can be used to identify candidates for the music's tempo. Finally, dynamic programming is used to estimate beat locations. This algorithm uses a cost equation that rewards placing beat locations on local maxima of the accent signal and penalizes deviations from the tempo estimate. A least-cost path of beat locations through the music is then plotted and used as the final beat estimates for the algorithm.

The second system used to establish a baseline was developed by Dixon et al (*Dixon*) [34]. This algorithm utilizes a feature called the spectral flux in order to calculate the accent signal. The spectral flux is the half-wave rectified magnitude difference between successive frames in a spectrogram; like the accent signal of the Ellis system, it also tends to have larger values when new events, such as beats, take place. Periodicity estimation is performed by thresholding the accent signal to find onset candidates and then clustering the differences between successive candidates; if many pairs of onsets are spaced according to the same period, that period is likely to be a multiple of the true tempo (or that tempo itself). Finally, the beat locations are found by locating sequences of maximal values in the accent signal that correspond

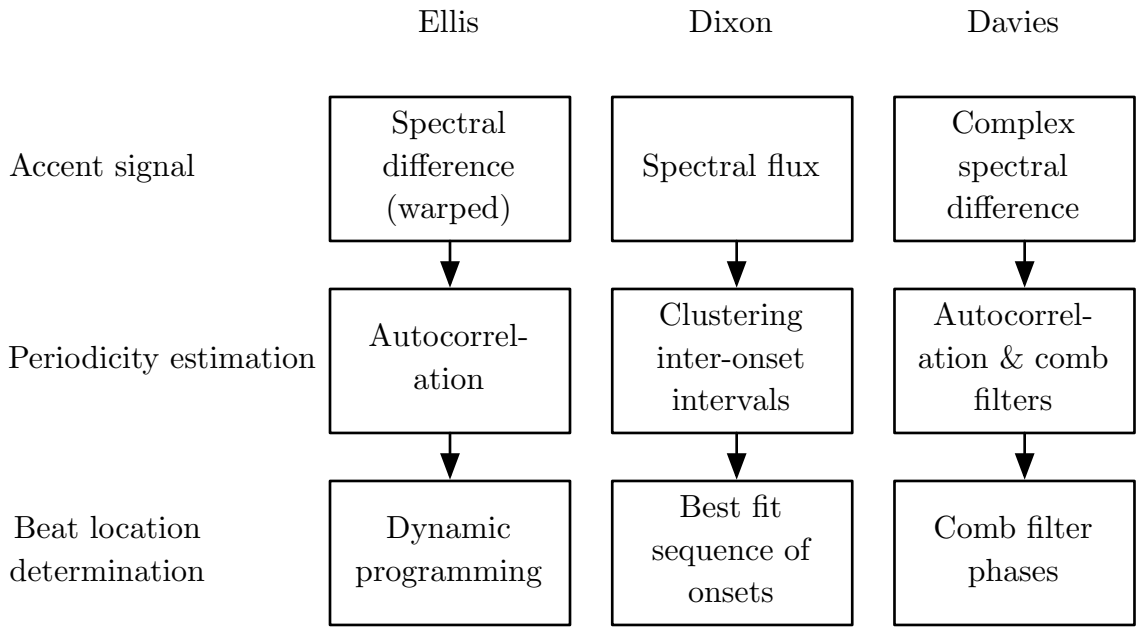

Figure 5.1: Comparison of the three state-of-the-art systems used to establish a baseline accuracy for Music-IR beat trackers in noisy audio.

to one of the tempo predictions. Multiple *agents*, or beat and tempo predictions, are considered at the same time, and the one that best fits the accent signal is considered to be the best prediction of the true beat locations.

The final baseline system was developed by Davies et al (*Davies*) [28]. This algorithm utilizes the complex spectral difference to calculate the accent signal. The complex spectral difference C is calculated for a frame n from a spectrogram S as follows:

$$C[n] = \Sigma_{k=1}^{K} |S_k[n] - \hat{S}_k[n]|^2 \tag{5.1}$$

k is the frequency index, and $\hat{S}$ is a prediction of the spectrum assuming no new events such as beats occur. The predictions are set such that the magnitude of the spectrogram is the same as in the previous frame, and the phase velocity also remains

the same. Conventionally, both the magnitude and phase would be assumed to remain constant in the absence of an event, but this feature permits the phase to keep varying at the same rate as in the preceding frame. This feature is thus less likely to produce incorrect local maxima in situations where the phase changes at a constant rate even when nothing is happening.

The accent signal is subsequently autocorrelated and then multiplied by a series of comb filters. The period of the comb filter that produces the strongest response is the periodicity estimate, and the system 'locks on' to this estimate for the next step. Finally, the accent signal is passed through multiple comb filters parameterized with the locked-on periodicity and different phases. The peak locations of the filter that produces the strongest resonance are estimated to be the beat locations. If the beats begin to deviate too much from the maxima of the accent signal, then the system changes from its 'locked on' state back to the original state, discards the periodicity estimate, and attempts to reestimate the tempo.

A comparison of the three state-of-the-art systems is shown in Figure 5.1.

5.2 Metrics

Two common beat tracking metrics are used to evaluate the baseline accuracy of these systems on clean and noisy audio. The first is based on the F-Measure [27]. This metric asserts that a beat within a certain threshold of a ground-truth beat is 'correct', and calculates a value based on the number of correct beat predictions C, the number of missed beats M, and the number of positions incorrectly marked as beats I:

$$P = \frac{C}{C + I} \tag{5.2}$$

$$R = \frac{C}{C + M} \tag{5.3}$$

$$F = \frac{2PR}{P + R} * 100 \tag{5.4}$$

These equations define the *precision*, P, as the ratio of correct beat predictions to all beat predictions, and the *recall*, R, as the ratio of correctly located beats to all beats in the music. The F-Measure, F, is then set as the harmonic mean of the precision and recall multiplied by 100. The threshold is set to 70ms, as is common in the literature. This metric has a maximal score of 100, when both I and M are equal to 0 (in other words, when there are neither any missed beats nor any positions incorrectly marked as beats), and a minimum score approaching 0 as C approaches 0. F is technically undefined when C is 0, and so is set to a value of 0 by convention. The metric thus has a range of [0,100].

The other metric is the Information Gain [27]. With this system, the neighborhood around each ground-truth beat is divided into several histogram bins, and the bin that each predicted beat falls in is recorded. The bins are then combined over every ground-truth beat, producing a distribution of estimated beat positions relative to actual beats. When a perfectly accurate tracker is used, all estimates will fall into the bin containing the ground-truth beat. A perfectly useless tracker, for which the ground-truth beat information have no bearing whatsoever on the beat estimates, will result in a uniform distribution across all bins. The Information Gain is calculated by taking the Kullbach-Liebler divergence between the actual beat histogram and a uniform histogram. Mathematically, it is defined as:

$$I = \sum_{k=1}^{K} p_b(z_k) \log_2 \frac{p_b(z_k)}{p_u(z_k)} \tag{5.5}$$

Where I is the Information Gain, p_u is a uniform distribution with K bins, p_b is the beat error histogram, also with K bins, and z_k is the kth bin. For the purposes

of these experiments, K was set to 41.

The uniform distribution has a probability of $\frac{1}{K}$ for all bins z_k, so the equation becomes:

$$I = \sum_{k=1}^{K} p_b(z_k) \log_2 \frac{p_b(z_k)}{\frac{1}{K}} \tag{5.6}$$

Rearranging terms:

$$I = \log_2(K) + \sum_{k=1}^{K} p_b(z_k) \log_2 p_b(z_k) \tag{5.7}$$

In terms of the Shannon entropy H, this equation can be written as:

$$H = -\sum_{k=1}^{K} p_b(z_k) \log_2 \frac{1}{p_b(z_k)} \tag{5.8}$$

$$I = \log_2(K) + H \tag{5.9}$$

The entropy H is strictly nonpositive, and so the Information Gain is maximal when $H = 0$. This occurs when all of the beats are concentrated into a single histogram bin (ideally, but not necessarily, the bin containing the actual ground-truth beat). In this case, one bin k^o has a probability of 1, and the other k bins have a probability of 0. This results in:

$$H = -p_b(z_{k^o}) \log_2 \frac{1}{p_b(z_{k^o})} - \sum_{k \neq k^0}^{K} p_b(z_k) \log_2 \frac{1}{p_b(z_k)} \tag{5.10}$$

The summand goes to 0 as $p_b(z_k)$ goes to 0. As for the other term:

$$H = -p_b(z_{k^o}) \log_2 \frac{1}{p_b(z_{k^o})} - 0 = -1 \log_2 \frac{1}{1} = -1 \log_2(1) = 0 \tag{5.11}$$

$$I = \log_2(K) + H = \log_2(K) + 0 = \log_2(K) \tag{5.12}$$

The Information Gain is minimal when the beat error distribution is identical to the uniform distribution that it is being compared to [25]. This indicates that the beat tracker's results are unrelated to the actual ground-truth beat locations. Plugging back into Equation 5.6:

$$I = \sum_{k=1}^{K} p_b(z_k) \log_2 \frac{p_b(z_k)}{\frac{1}{K}} = \sum_{k=1}^{K} \frac{1}{K} \log_2 \frac{\frac{1}{K}}{\frac{1}{K}} = \sum_{k=1}^{K} \frac{1}{K} \log_2(1) = 0 \tag{5.13}$$

Because the maximal value is $\log_2(K)$ and the minimum value is 0, the Information Gain can simply be normalized by $\frac{100}{\log_2(K)}$ to ensure that the range of the output is [0,100], like the F-Measure metric.

$$I = \Big(\sum_{k=1}^{K} p_b(z_k) \log_2 \frac{p_b(z_k)}{\frac{1}{K}} \Big) \frac{100}{\log_2(K)} \tag{5.14}$$

The F-Measure metric is more meaningful at high accuracies; these indicate that a sequence of beats very nearly matches the ground-truth. It is less easy to interpret at low accuracies, since a very low score could be caused, for example, by the beat detection algorithm tracking the offbeat, which is still indicative of competent, albeit imperfect, tracking. Conversely, Information Gain is most meaningful at low accuracies, where the beats are uniformly distributed around the ground-truth beats. A high Information Gain score, by contrast, could not only result from a series of perfect beats, but a series of beats that are always shifted by some constant from the true values. Both metrics are used to present a fuller perspective of how the algorithms perform.

5.3 Experiments with Standard Beat Trackers

The state-of-the-art algorithms were run on the Dance Dataset, both for the set in which no noise was added (the *Clean* dataset) and a set in which noise was added

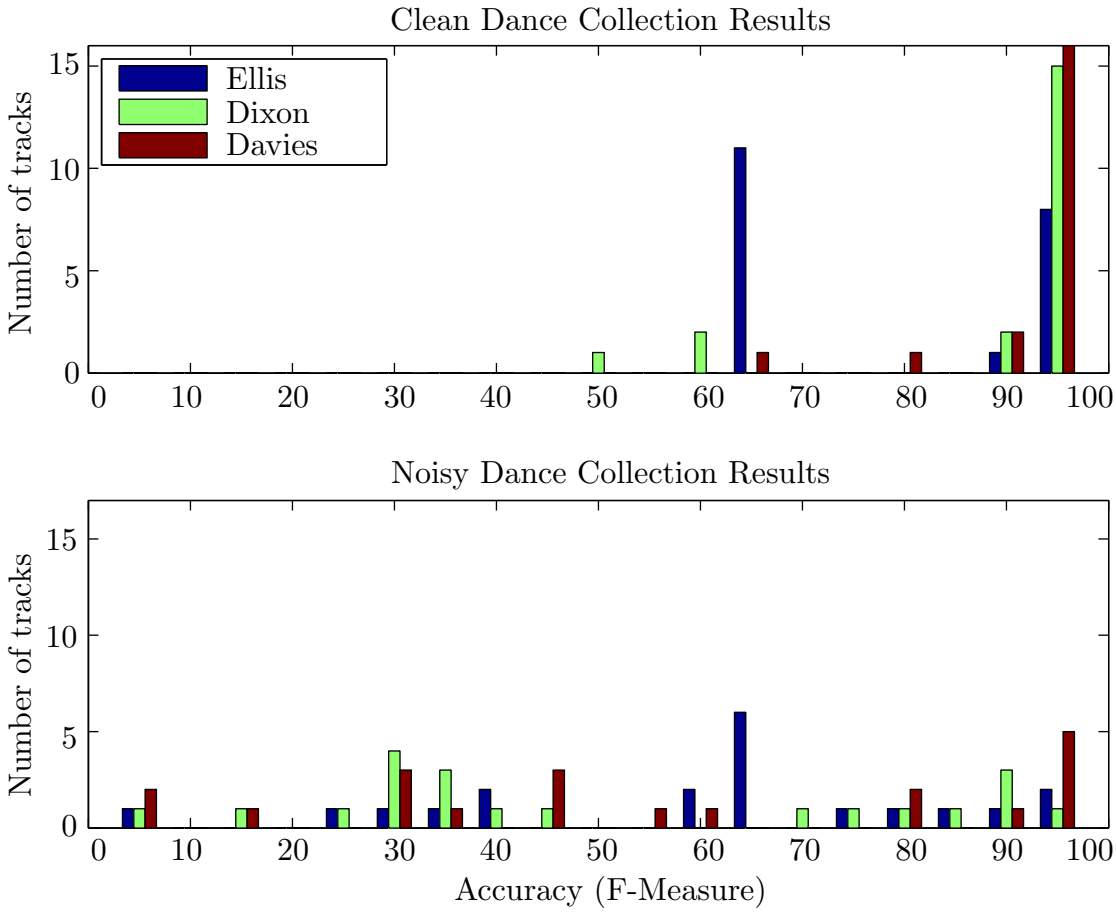

Figure 5.2: Results of the state-of-the-art beat trackers on the clean (top) and noisy (bottom) dance music, in F-Measure.

(the *Noisy* set). The Dance Dataset was chosen for the baseline investigation of the problem because the music is generally easier to track without noise; this would ensure that the effects of the noise would be clear and not masked by inherently difficulty in the music. The noisy dataset chosen was the dataset containing both room noise and robotic noise, in order to present the system with a variety of noises.

Figure 5.2 shows the F-Measure results on these datasets, with the clean music tracking results on top and the noisy music results on the bottom. Figure 5.3 shows

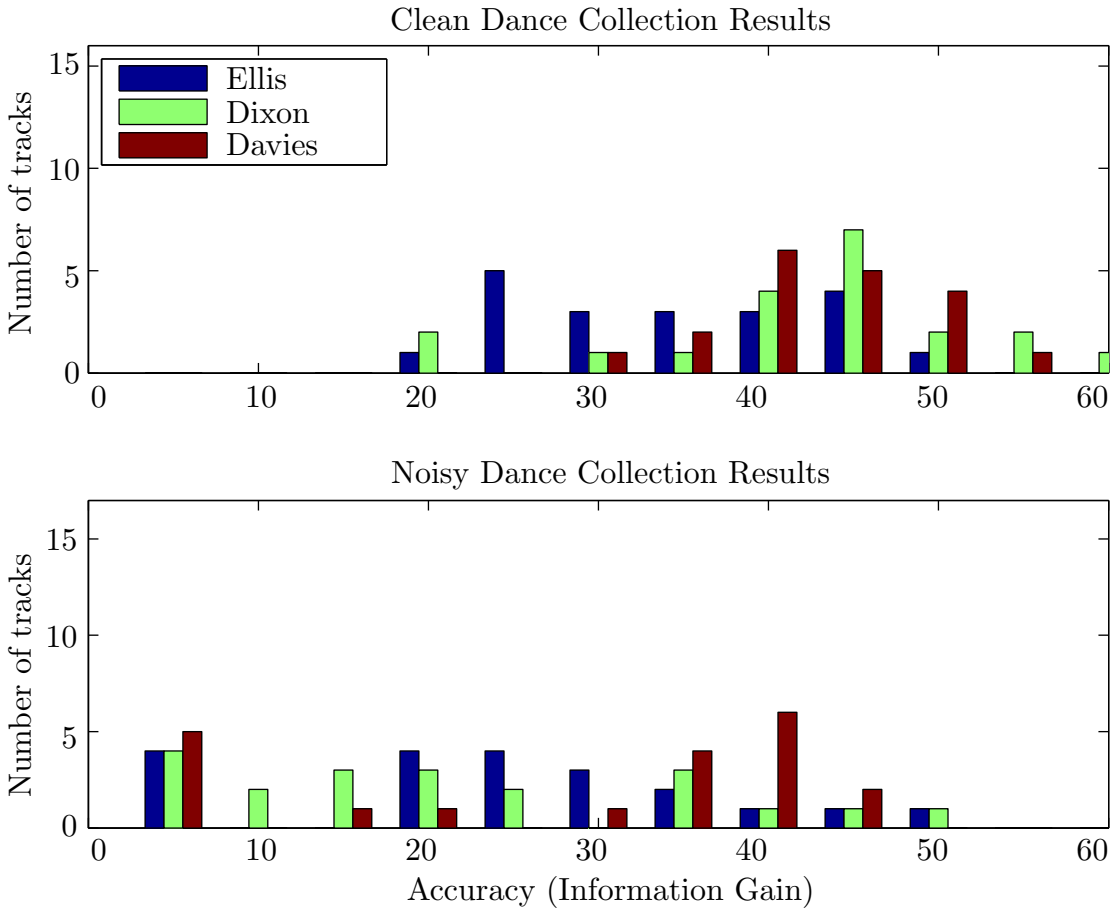

Figure 5.3: Results of the state-of-the-art beat trackers on the clean (top) and noisy (bottom) dance music, in Information Gain.

the Information Gain results on the same music. Additionally, the average change in accuracy on each song, in terms of both F-Measure and Information Gain, was found for each beat tracker (Table 5.1).

The deleterious impact of the acoustic noise is clear. In terms of F-Measure, the clean dance music tracks extremely well, with many tracks from each system near a perfect score of 100. Three of the four trackers have at least 17 songs, or 85% of the set, at above 80 in terms of accuracy. However, results fall drastically after noise is

Table 5.1: Average change in accuracy of beat estimates between clean and noisy versions of each song in the Dance Dataset for all trackers.

Metric	Ellis	Dixon	Davies
F-Measure	-19.0	-39.0	-38.2
Information Gain	-10.0	-20.8	-16.3

introduced. All trackers begin recording scores of 50 and below, and there are even results at 20, 10, and approaching 0.

Similar results are shown with the Information Gain results 5.3. Initial scores are lower, centered at about 40, so the accuracies do not have as far to fall as in the F-Measure case. However, they still do collapse, with each algorithm's scores noticeably lower than they were on the clean audio. With clean music, no tracker ever received a score lower than 20, but on noisy audio, each tracker has at least three scores (or 15% of the dataset) approaching 0, which indicates essentially random beat positions relative to the true positions.

Finally, Table 5.1 quantifies the extent of the reduced accuracy. Even the tracker with the smallest amount of change in terms of F-Measure, the Ellis tracker, still has each song on average fall 19 points between the clean and noisy beat estimates. Information Gain decreases are slower, but that is only because the clean audio scores are also lower on average, so they do not have as far to fall. Even the Ellis system, which again records the least decrease, performs 10 points worse on average when comparing its results on clean and noisy versions of a song.

5.4 Experiments with Noise-Reduction Algorithms

Before developing the proposed noise-robust system, I evaluated the possibility of utilizing noise-reduction systems to strip out much of the noise from the audio

signals in order to enable conventional Music-IR algorithms to achieve greater accuracy. These two approaches, noise-reduction and noise-robustness, have been tried for other tasks involving processing noisy acoustic signals such as Automatic Speech Recognition, and while noise-robustness was generally found to be a very effective option in the ASR case, there was no proof that this would be the case for this particular problem [103, 120]. As such, I performed an experiment to see how some noise-reduction algorithms could cope with the datasets containing music that was recorded in the presence of noise.

I first implemented a conventional algorithm in order to perform the beat tracking task. In order to calculate an accent signal, this system divided the acoustic spectrum into several subbands and calculated the energy in each subband over time [64]. Breaking the audio into different subbands is a common feature in beat tracking algorithms, as it allows for the system to be sensitive to events happening in different ranges of the spectrum [111]. The energy envelopes were then autocorrelated in order to begin determining the period of the music. To find the total period, the autocorrelations were normalized to remove the natural bias of the signals towards their centers, then summed together. The periodicity was subsequently found using a peak-picking approach that also included some heuristics to reduce the possibility of picking an unusually fast or slow tempo.

The period and the subband energy feature were then used to estimate beat locations. The energy of a given frame would be summed over all subbands, and then would be added to the total energies of frames spaced at multiples of the audio period. This determined a feature I termed the *multi-frame energy*. If that frame was a beat location, then not only would it likely have a great deal of energy, but frames spaced at exact multiples of the period behind it would also likely be beats and have relatively high energy values. As such, the multi-frame energy for one frame was

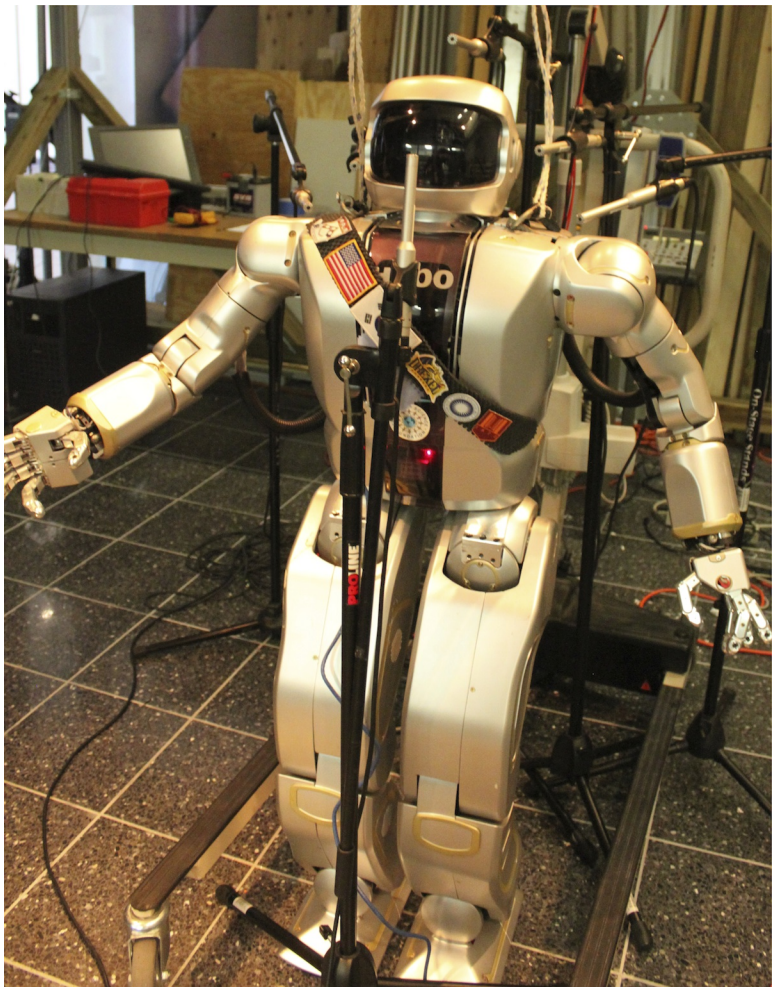

Figure 5.4: Recording setup for the noise-reduction approach experiment.

compared to that of all other frames in a one-period window, and the frame with the highest multi-frame energy was judged to be a likely beat. Because the multi-frame energy system helped ameliorate the effects of one section of music being very loud (in which case everything would likely be energetic) or the occasional rest beat (in which case its frame would likely have low energy), this system was designed to be robust to natural deviations in the music.

I also implemented two different noise-reduction pre-processing algorithms. The first was a static spectral subtraction algorithm which processed a sample of pure noise

taken while the robot was moving, identified the noisiest subbands, and erased those subbands from the spectrogram. In this way, much of the noise was redacted, though some spectral content of the music within those bands was also lost. The second approach attempted to retain more of the useful information by first analyzing the spectrogram of pure noise and calculating a threshold value for each spectral bin based on statistics, such as mean and standard deviation, of that bin. It then adaptively only erased bins that were in the noisiest subbands and whose energy was below the threshold value (and whose energy was thus likely contributed by noise and not the musical signal).

For this preliminary study, another noisy dataset was recorded, this one less noisy than the Dance and General Datasets described in Section 4.1 that would be collected later. The music from the Dance Dataset was recorded in the presence of Hubo, but instead of being recorded on the two lapel microphones positioned in its head, the music was recorded with an array of six stick microphones mounted all around the robot (Figure 5.4). Because some of these microphones were not as close to the noisy components of Hubo, such as its shoulder motors and computers, the effects of noise on the recording were lessened. Also, Hubo's entirely body was covered in its shell, further obscuring the noise. Two recordings were taken, one in which the robot's arms were moving randomly, and one in which they were moving in synchrony with the beats.

The results of this preliminary experiment are shown in Table 5.2 [64]. The algorithm did achieve an F-Measure score of 98 on the clean version of the dataset (with a range of [0,1]), as would be expected given that dataset's tendency towards music with strong, steady beats. Also as expected, accuracy fell once the robot noise was introduced, dropping to 84-86 depending on the particulars of the robot's motion. Neither the static spectral subtraction filter nor the adaptive one, however, improved

Table 5.2: Results of beat tracker on the less-noisy version of the Dance Dataset, recorded with the robot moving both randomly and periodically. The system was tested with no noise reduction, with a static spectral subtraction system, and with an adaptive spectral subtraction system. All results are in F-Measure with a range of [0,100].

	No. of	Clean	Random motions			Periodic motions		
Tempo	songs	audio	None	Static	Adaptive	None	Static	Adaptive
90-115	5	98	82	92	92	83	85	94
116-125	5	98	90	96	96	89	88	92
126-132	5	98	88	95	96	95	93	95
133-152	5	96	77	87	95	76	83	86
Total	20	98	84	93	94	86	87	92

the accuracy to a level that was anywhere near to the original case. The noise reduction approach worked particularly poorly for the case in which the robot was moving its arms periodically; the static filter did almost nothing, and the adaptive system only covered 50% of the distance between the accuracy of the system on noisy audio with no filter and the accuracy of the system on clean audio. The noise reduction approach worked somewhat better on the random motion case, but the system still only improved about 70% of the way back up to the results from when the system ran on clean audio.

The noise reduction approach thus was unable to compensate for even relatively mild amounts of noise. This provided evidence that, in order to develop a system that could learn beats even in noisy acoustic environments, it would be better to follow the model of ASR and similar systems and adopt a noise-robustness approach.

6. Noise-Robust System for Learning Musical Beats

As current state-of-the-art systems are unable to accurately locate, much less learn, musical beats in noisy acoustic signals, I designed a novel algorithm to perform this task. Such an algorithm could have wide utility by making it possible to perform accurate beat analysis of much more of the audio that humans routinely listen to, such as music played at bars and clubs and musical performed in conjunction with machines such as robots. This audio could then be made available for the variety of tasks that beat tracking make possible, such as automatic sorting of music by tempi to create playlists containing tracks at specific speeds, automatic beat-synchronous blending of consecutive musical tracks, and enabling robots to perform in time with the music. Furthermore, the additional knowledge obtained by learning about the beats could also be used for tasks such as beat classification.

The proposed system was designed subject to several constraints. For one, it was imperative that the system be flexible and not assume *a priori* that the beats will correspond to some hand-selected feature. Any hard choice of a feature risks producing inaccurate results if the music in question should have beats which do not correlate well with local extrema of that feature, which is likely when noise interferes with the signal. As such, the features should be learned from the noisy acoustic signal directly, rather than being specified *a priori*. Second, the system must be able to not just find beats, but also learn their spectral characteristics, as such information could be of use in various other systems. Third, because humans are often able to determine information about the beats and other aspects of music quickly, it was desired that the system do the same. In particular, a constraint was imposed that the system should be able to analyze the beats thus far after only five seconds of noisy music.

The proposed system uses several powerful signal processing and machine learning

techniques in order to learn musical beats. In order to identify the the spectral characteristics of the beats, it utilizes a version of Probabilistic Latent Component Analysis (PLCA), an algorithm that employs Expectation-Maximization in order to accurately decompose a signal, such as an acoustic source, into its component parts. However, because the basic PLCA algorithm does not allow for the ability to exploit the time-varying characteristics of a signal, this system also uses a representation of a musical signal called a *stacked spectrogram* which, in conjunction with PLCA, allows for the time-dependent relationships between different spectral bins in a component to be modeled. Harmonic-Percussive Source Separation (HPSS) was also used in order to further decompose the audio signal and separate the percussive portion, which often contains useful beat information, from the harmonic portion, which is less likely to do so. Because basic autocorrelation did not perform sufficiently well for the periodicity-estimation step of my algorithm, other techniques which could obtain more accurate results were explored and employed. By using these algorithms among others, the system was designed so as to perform the beat-learning task accurately even in acoustically noisy environments.

6.1 Probabilistic Latent Component Analysis

In order to learn the spectral characteristics of the beats within a noisy musical signal, the proposed system uses a version of Probabilistic Latent Component Analysis (PLCA) to break the signal down into its component elements. For the system to be able to accurately model the beat components, it must incorporate the way in which those components change over time. PLCA is ill-equipped to do this out of the box, so the first creates a representation of the spectrogram which contains those changes in a manner that is easier for PLCA to process.

6.1.1 Spectrogram Representation

Audio is passed into the system through the microphones at a sampling rate of 44.1 kHz. If multiple microphones are used, as is the case with the datasets incorporating robot noise, then the signals are averaged to form a monaural signal. An initial magnitude spectrogram is then calculated using a 256-point Fourier Transform with 50% overlap. This resolution was found empirically to be high enough for useful identification of beat times and spectral characteristics, while also low enough to help keep the problem computationally tractable. Using higher resolutions provides diminishing returns of accuracy in exchange for the increased computation, and because this algorithm may eventually be used in conjunction with other Music-IR systems that would also require processing power, this particular resolution was found to be optimal in terms of balancing resolution and computational resources.

The spectrogram, with each column containing 128 points representing 22.05 kHz, is then split into frames of 8 columns, or 46.3ms of audio, with 50% overlap. The 8 columns in each frame are then averaged over time to produce a single column representing the entire 46.3ms frame, and the columns are aggregated into a *reduced spectrogram*. This process reduces the size of the data still further, again allowing for more efficient computation, and also helps to reduce the effects of spurious bins in the spectrogram. Because the initial signal is assumed to be noisy, particular bins may exhibit unusual values that could reduce the accuracy of the system. By averaging them over time, the effects of such bins are reduced. While this averaging process cannot remove the effects of all noise, and while it is ineffective against subbands that are noisy throughout the majority of the 46.3ms window, it is useful nonetheless.

Finally, the reduced spectrogram is also split into frames. Each contains three columns of the reduced spectrogram and represents 139ms of data, and the frames are again spaced 23.2ms apart. Because this data has not been averaged together,

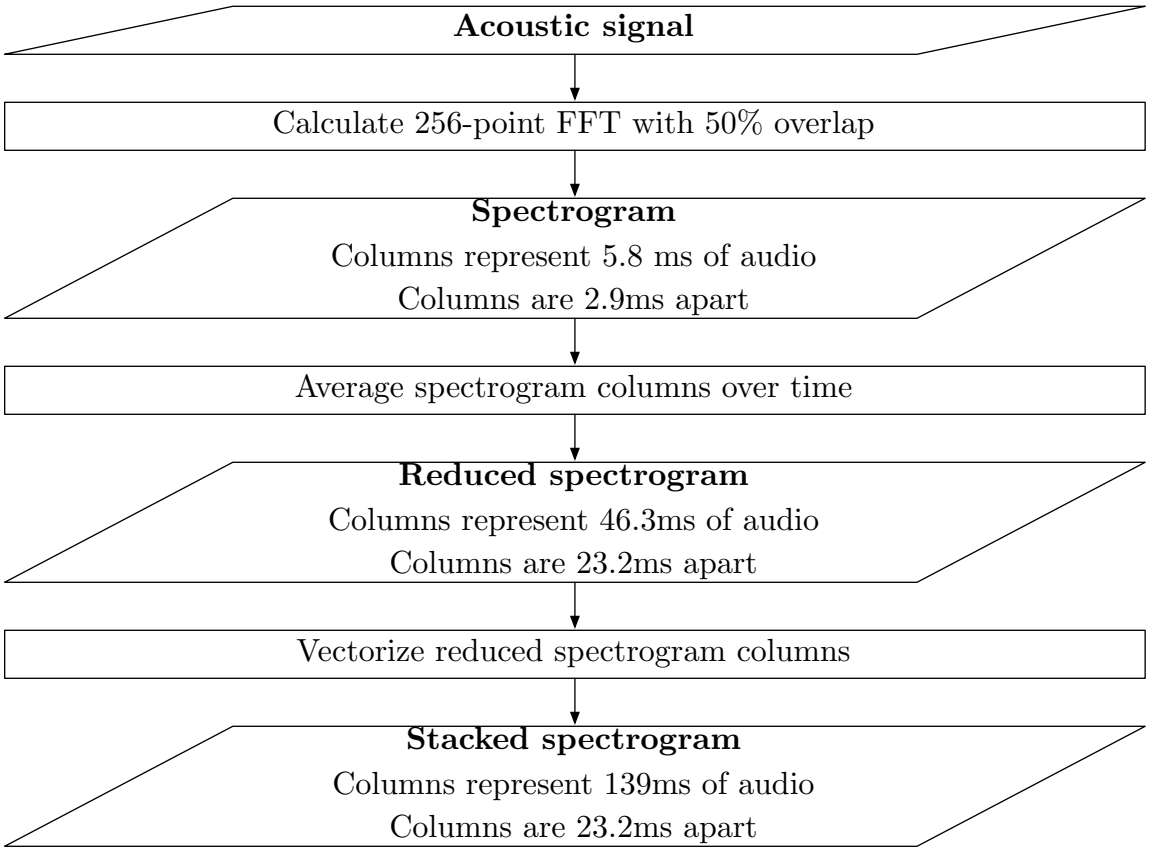

Figure 6.1: Flowchart of the stacked spectrogram calculation.

however, the frame can be used to show how particular spectral bins change between the first third, second third, and final third of the frame. The three columns in each frame are then vectorized into a single column of 384 bins, and those final columns are aggregated into a 'stacked' spectrogram (Figure 6.1). This is the final representation which is passed to the PLCA portion of the system.

The structure of the stacked spectrogram allows for frequency characteristics that last for more than 46.3ms to be represented in a single column. The PLCA algorithm can then utilize those relations, observing for example that a particular beat component might begin with a large value at a given frequency which subsequently decays over the course of the beat. The system can thus learn time-varying spectral

characteristics.

6.1.2 Decomposition of the Acoustic Signal via PLCA

The next step is to break down the stacked spectrogram into its component elements. There are many methods that could be used to decompose the spectrogram, including Principal Components Analysis (PCA) and Independent Component Analysis (ICA), but both of these can produce components with negative values, which are not meaningful for a spectrogram [116]. Other options include Probabilistic Latent Components Analysis (PLCA) and Non-Negative Matrix Factorization (NMF), as they can decompose a matrix such as a spectrogram into non-negative values [115]. PLCA additionally provides a probabilistic framework that allows the system to model the stacked spectrogram as a histogram drawn from a set of latent components. This model allows the use of an efficient Expectation-Maximization algorithm to determine those elements which comprise the signal [117]. This system therefore uses a version of PLCA to decompose the stacked spectrograms into its individual components.

Given a stacked spectrogram S with T time indices and whose columns each contain F bins, the system models each column in that spectrogram as the result of a large number of 'draws' from a mixture of Z components [117]. The distribution for each frame t of audio is described as:

$$P_t(f) = \sum_{z=1}^{Z} P(f|z)P_t(z) \tag{6.1}$$

Where $P_t(f)$ is the distribution of the audio in frame t, $P(f|z)$ is the distribution of each component z, and $P_t(z)$, denoted as the *activation probabilities*, is the probability of each component z being active during frame t. By treating columns of S, denoted as S_t, as observations of $P_t(f)$, the system can estimate the true values of

$P_t(z)$ and $P(f|z)$. These latter two distributions provide the needed information to determine which component is most likely to represent a beat, as well as the spectral characteristics of that beat.

In the Expectation step, the system calculates:

$$P_t(z|f) = \frac{P_t(z)P(f|z)}{\sum_{z'=1}^{Z} P_t(z')P(f|z')} \tag{6.2}$$

This is the *a-posteriori* probabilities of the components z at time t given observed frequencies f. After Equation 6.2 is calculated, the Maximization step is performed to update both $P_t(z)$ and $P(f|z)$:

$$P_t(z) = \frac{\sum_{f=1}^{F} P_t(z|f)S_t(f)}{\sum_{z'=1}^{Z}\sum_{f=1}^{F} P_t(z'|f)S_t(f)} \tag{6.3}$$

$$P(f|z) = \frac{\sum_{t=1}^{T} P_t(z|f)S_t(f)}{\sum_{f'=1}^{F}\sum_{t=1}^{T} P_t(z|f')S_t(f)} \tag{6.4}$$

The Expectation and Maximization steps alternate until convergence or for a certain number of iterations. In practice, 40 iterations was found to be sufficient in order for the system to converge for both $P(f|z)$ and $P_t(z)$. The system then saves both of those probabilities. The activation probabilities will be used subsequently to determine which component is most likely to be a beat, at which point the spectral profile of that component can be used to obtain an understanding of what that beat source looks like in the frequency domain. If needed, the components can also be unstacked to show the spectral characteristics of those components over time.

Two examples from the Dance Dataset are shown in Figure 6.2. Clean and noisy audio spectrograms of two audio excerpts are displayed, as are the activation probabilities and the spectral characteristics of a component that contains beat information for each excerpt. Peaks in the activation probabilities are aligned with the beat structure of the audio, even though that structure is obscured in the noisy audio.

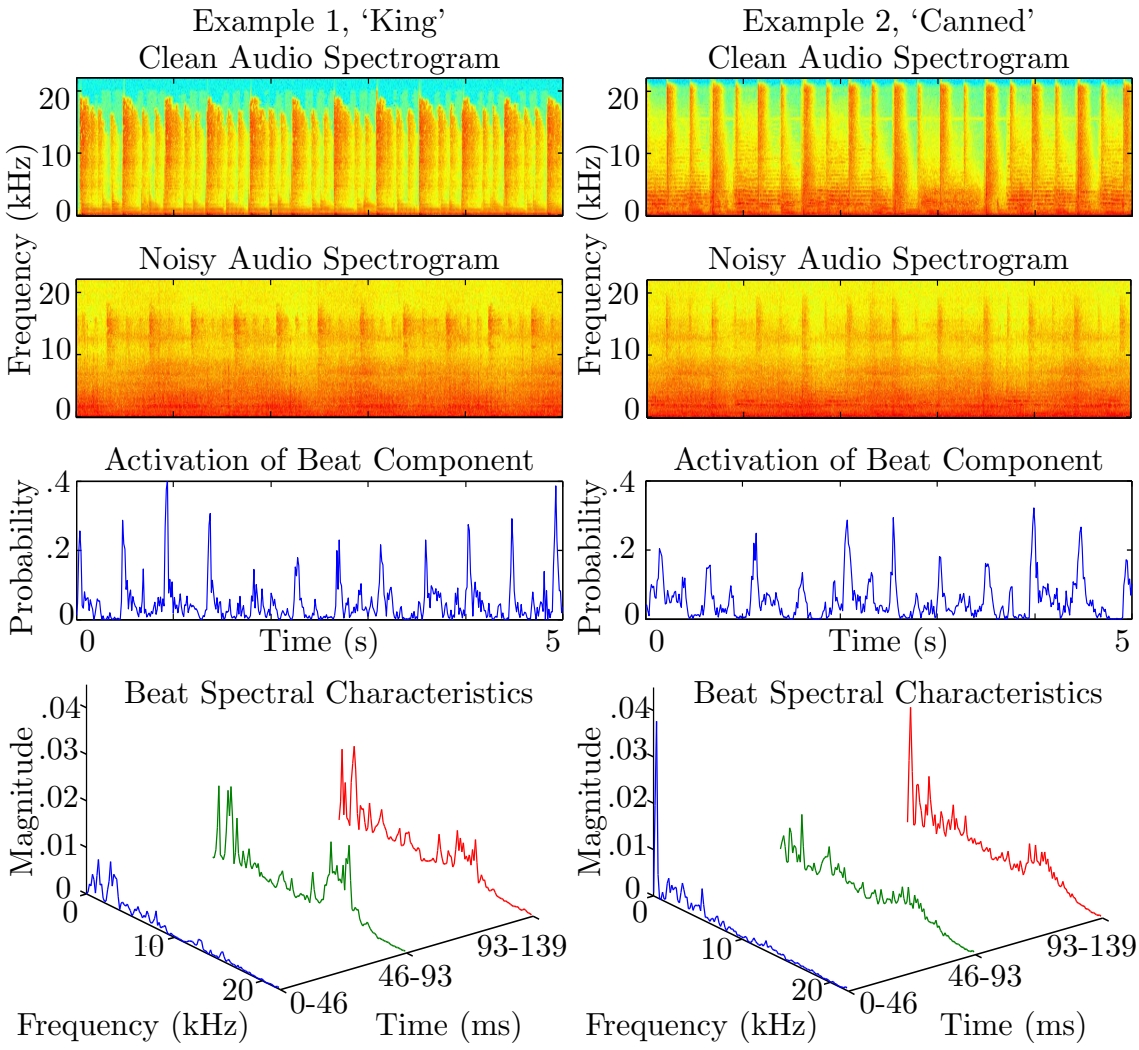

Figure 6.2: Spectrograms, activations, and beat spectral characteristics for excerpts of Blood and Whiskey's 'King of the Fairies' and Jamiroquai's 'Canned Heat'.

Additionally, the differences between the spectral characteristics of the beats in each excerpt are visible. For example, the spectrograms indicate that the beat spectra for the first example stop short of 20 kHz, while those of the second example extend further up the spectrum. This is reflected in the spectral characteristics plots; 'King' has a very sharp cutoff in its frequency spectrum at about 17 kHz, while 'Canned' has a much smoother rolloff after that point, indicating that it contains energy in higher

frequencies.

6.2 Harmonic-Percussive Source Separation

While the PLCA step is capable of decomposing the audio into its component elements and thereby enabling the system to separate out the important components, such as the instruments that produce beats, from unimportant components, such as noise sources, it is also useful to eliminate some of the unimportant information before reaching that step. This reduces the chances of the algorithm accidentally identifying an unimportant component as being related to the beats, which are the important feature at hand. Because beats are largely percussive in nature, one method to get rid of some of the less relevant information is to separate out the harmonic portion of the signal from the percussive portion. The proposed beat learning system therefore performs harmonic-percussive source separation (HPSS) before performing the PLCA operation.

Several methods have been developed for performing HPSS. One popular choice is to record multiple channels of audio, take spectrograms of each channel and stack them into a tensor, and then to use Non-Negative Tensor Factorization algorithms to decompose the tensor into pitched (i.e., harmonic) sources and nonpitched (i.e., percussive) sources [47]. This algorithm, however, is very computationally intensive, and so is suboptimal for a system which might need to run many other tasks at the same time, which is the case for the proposed beat learning system [46]. Another option is to generate models of the expected percussive signals and then find them in the recorded audio using template matching systems [133]. It is difficult to use this approach on unknown music, though, because the nature of the percussive source might not be known. One requirement for this system is that it not assume prior knowledge of what a beat may look like in order that it be flexible enough to deal

with unusual types of beats that may differ from the norm. Finally, median filtering algorithms can be used in order to obtain harmonic and percussive weights for each bin in the spectrogram [46]. An implementation of this final algorithm is utilized in the proposed beat learning algorithm.

The idea behind median filtering is that, when visually inspecting a spectrogram, harmonic sources appear more like horizontal lines, while percussive sources are visible as vertical lines [101]. Harmonic notes and sounds are characterized by energy at narrow frequency ranges; for example, if a pianist plays the key A440, then the resultant spectrogram will show energy clustered tightly at 440 Hz, as well as 880 Hz, 1320 Hz, and other octaves. That energy will also remain for some time in the spectrogram, not decaying for several frames, and so the result is lines that are thin in height, since the energy is narrowband, but extend for a long time, and so look horizontal. Conversely, percussive sources are often wideband, such as the pounding of non-pitched drums or the clapping of hands. Because these sources often provide bursts of energy, their representations in the spectrogram fade very quickly over time. Such sources then are tall in frequency but very short in time, appearing vertical.

This delineation between harmonic and percussive sources allows for a computationally efficient method of separating them. Given a spectrogram S, the frequency content at frame t can be passed through a median filter in which each element of that column is replaced by the median value of its neighbors. Harmonic components, which are essentially outliers crossing the frame, will be wiped out by the median operation, but percussive components will remain [46]. By taking each frame t of the spectrogram and passing it through a median filter, a *harmonic-suppressed spectrogram* is created. Similarly, by taking a row of the spectrogram which represents one frequency range over time, median filtering can be used to suppress percussive components while maintaining the harmonic ones. Using the median filter on all

rows results in a *percussive-suppressed spectrogram*. Median operations are superior to means for this purpose, because then the chosen values will not depend at all on outliers (such as the harmonic or percussive components that are being smoothed out) which could skew the results.

The harmonic- and percussive-suppressed spectrograms are then used to assign the energy in each element of the original spectrogram as belonging to harmonic or percussive components in some proportion. A 'hard' mask, in which each bin is assigned fully to the harmonic or percussive case based on whether the percussive-suppressed or harmonic-suppressed spectrogram has a larger value, is undesirable because it performs poorly for bins which contain a mixture of harmonic and percussive components. It is possible, however, to create a soft mask which avoids this problem. The masks are calculated with:

$$M_{H_{f,t}} = \frac{H_{f,t}^2}{H_{f,t}^2 + P_{f,t}^2} \tag{6.5}$$

$$M_{P_{f,t}} = \frac{P_{f,t}^2}{H_{f,t}^2 + P_{f,t}^2} \tag{6.6}$$

Where M are the masks, H and P represent harmonic and percussive components respectively, and f and t represent frequency and time indices respectively. The values for each mask are in the range [0,1]. The value of the harmonic mask for a given element is large when the equivalent element in the percussive-suppressed matrix is much larger than the equivalent in the harmonic-suppressed matrix, and the same element in the percussive mask is large when the opposite is true. The masks also sum to one for any given element.

The harmonic and percussive spectrograms can then be found by multiply the original magnitude spectrogram by the masks:

$$S_H = S \otimes M_H \tag{6.7}$$

$$S_P = S \otimes M_P \tag{6.8}$$

Where S is the original spectrogram and S_H and S_P are the harmonic and percussive spectrograms, respectively. $\otimes$ represents the element-wise multiplication operation. The proposed beat learning algorithm then retains only the percussive spectrogram and passed it on to the PLCA portion of the algorithm. This part of the system therefore discards the unneeded harmonic information while maintaining the percussive component that is much more likely to contain beat information.

An example of HPSS being performed on a musical excerpt from the Dance Dataset is displayed in Figure 6.3. The topmost spectrogram is from the original signal, and contains a mixture of harmonic and percussive elements. The second spectrogram is the result after the harmonic mask has been applied.The percussive components, especially the weaker ones, are greatly reduced in this spectrogram, while the harmonic components (largely in the bottom of the spectrum) remain. The third spectrogram is the result after the percussive mask was applied to the original signal. The harmonic components are now the ones that have been redacted, with only the vertical ones remaining. This effect is particularly clear in the lowest frequencies. In the original spectrogram, this region was largely obscured by harmonic components, and the percussive components in this area were barely visible. Now, though, the vertical lines that represent the percussive components are apparent even in the lowest frequencies and are no longer obscured by harmonic components. Finally, the last spectrogram shows the sum of the harmonic and percussive spectrograms. It is identical to the first, demonstrating that no information from the signal is lost when performing HPSS in this manner.

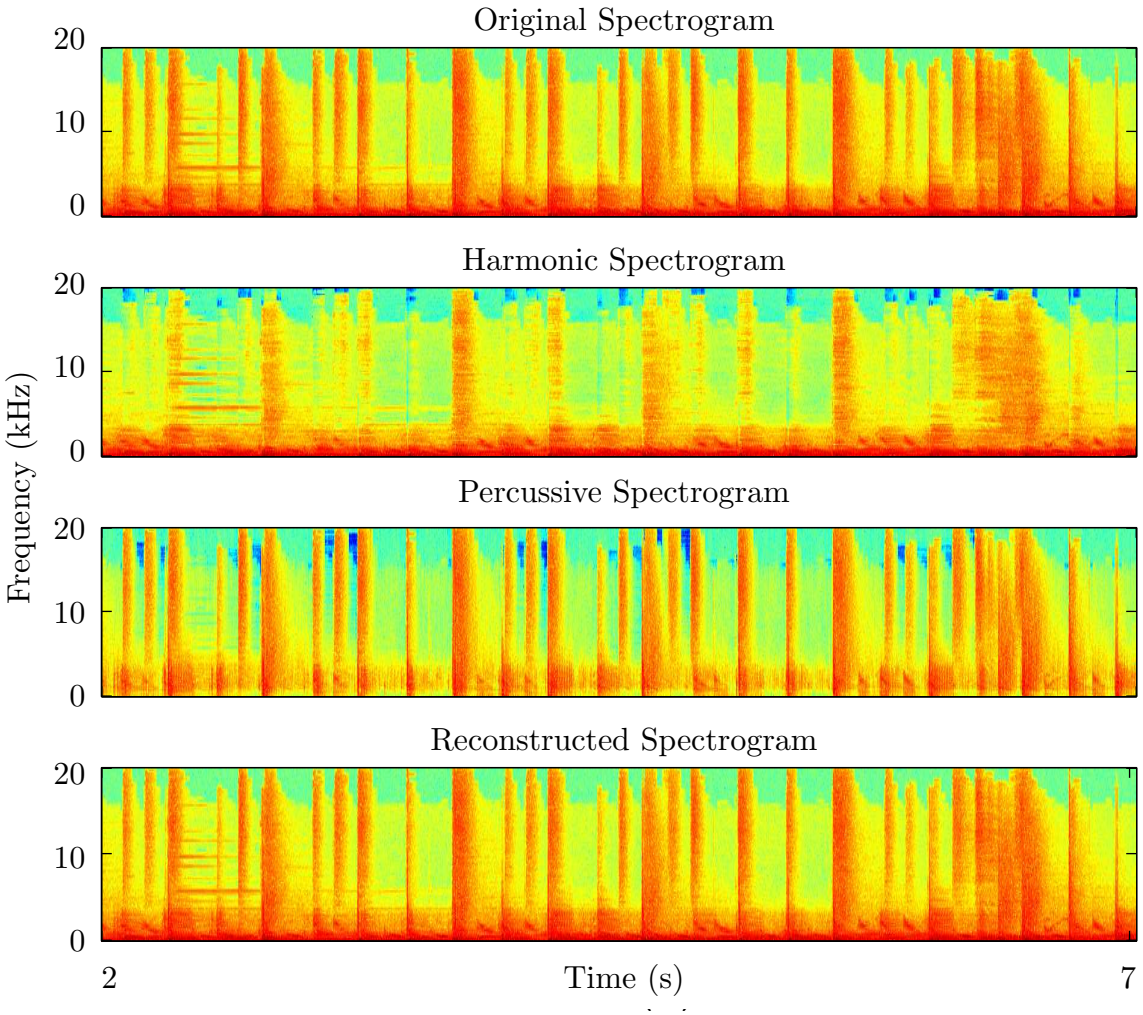

Figure 6.3: Original spectrogram, harmonic and percussive spectrograms, and reconstructed spectrogram formed from an excerpt of 'Fame' by Irena Cara.

6.3 Impulse Train Correlation for Periodicity Determination

After performing HPSS and PLCA on the noisy musical signal, the system has knowledge of a set of components that comprise the signal, as well as the corresponding activation probabilities over time for each component. For the task of determining beat locations, the activation probability of the component most closely corresponding to the beat can be used as an accent signal in a beat tracker, as that signal would be likely to have local maxima during frames which contained beats. However, the

determination of which component is the most likely to correspond to beats (i.e., the *beat component*) is a non-trivial task, and as there is no guarantee that any other components will have local extrema on or near beat locations, if the system chooses the wrong component as the beat component then it will be likely to perform inaccurately. The proposed beat learning algorithm therefore implements a system to identify which component most likely corresponds to the beat, and also obtains information about the periodicity of the signal, which it needs to estimate the music's tempo, at the same time.

One possible technique that could be used for this purpose is to analyze the spectral characteristics found via PLCA and to pass them either through a template-matching system or through an heuristic test in order to find which component most conformed to expectations about what a beat 'should' look like. For instance, because many pop songs include heavy beats in the low frequencies, an algorithm of this sort could scan the spectral characteristics of each component and categorize those with high energy values in the low frequencies as being more 'beat-like' than those with lower energy values in that range. However, such a system risks biasing the algorithm towards preconceived ideas of what beats should look like which may not match up properly with any particular song. There is music that has beat information in higher frequencies, as well as music whose low-frequency energy has little to do with beats, and the 'energy in low frequencies' heuristic would be liable to produce inaccurate results when presented with such music. As such, template- and rules-based systems for determining the beat candidate were not used.

Another potential method for this task is to perform autocorrelation on the activation probabilities. Autocorrelation is a mathematical operation calculated with:

$$R_{xx}[j] = \Sigma_n x[n]\bar{x}[n-j] \tag{6.9}$$

Where $x[n]$ is a signal and $\bar{x}[n]$ is its complex conjugate. In this case, the signal being autocorrelated is a probability and is strictly real (and, in fact, is specifically within the range [0,1]), and so the autocorrelation becomes:

$$R_{xx}[j] = \Sigma_n x[n]x[n-j] \tag{6.10}$$

This is summing a signal with a copy of itself that is reversed in time and is delayed by j. If the signal is periodic, then it is likely to have constructive interference when j is a multiple of that period. Furthermore, the activation probability of a beat component is likely to be roughly periodic. If the beat occurs at a steady rate, then the component should have local maxima at that same rate and should be relatively small elsewhere. The choice of beat component can thus be estimated by autocorrelating the activation probabilities of all the components and then finding the one with the largest values in the (normalized) autocorrelations. That activation probability is likely to correspond to the beat component, and furthermore, the location of that maximum value within the autocorrelation is likely to be a multiple of the tempo.

Unfortunately, while the autocorrelation-based system can produce powerful results on clean audio, it does not work nearly as well on noisy audio. If the noise is strong enough, then even after PLCA, there may still be some distortion in the activation probabilities. The subsequent correlation of two noisy signals, the noisy activation probabilities with themselves, occasionally results in another noisy signal which is difficult for the system to process. Certain types of noise, such as the sounds of the robot's motors when they steadily move up and down, can also look roughly periodic, which can cause the unfortunate situation of the algorithm beginning to track the robot's noise instead of the music. As such, autocorrelation was found to be a suboptimal choice. Results evaluating an autocorrelation-utilizing variant of the proposed work are shown in Tables 7.3 and 7.4.

Ultimately, the algorithm used in the proposed beat learner still utilized correlation, but instead of correlating the probabilities with themselves, they are correlated with signals similar to impulse trains. The algorithm creates a series of periodic signals for correlation according to the formula:

$$C_T[n]\} = \begin{cases} 1, n = kT \\ c, n \ = kT \end{cases}$$

Where C_T is one of the correlation signals with a period equivalent to a tempo, T, n is the frame, k is any positive integer, and c is a small constant to represent the fact that elements of the beat component, such as frequencies that are strong for that component, may also be present in non-beat frames. Experimentally, c was set to 0.1. Initially, correlation signals were calculated with periods corresponding to tempi in the range of [80, 160], which encompasses 95% of popular music [93]. However, some pieces of music had different-sounding beats on the downbeats and offbeats, and if the system only identified one of those types of beats, the strongest correlations would be at half the true tempo value. To accurately process these signals as well, 74 correlation signals were ultimately created with periods equivalent to tempi in the range [51, 188]. An example of one such signal is shown in Figure 6.4.

In order to designate the appropriate component as the beat component, the activation probabilities of each component were correlated with all of the correlation signals, and the single largest correlation was found. The activation signal for that correlation was judged to correspond to the beat component, and the correlation signal used to generate that correlation was judged to have a period equivalent to the signal's tempo. There was also a post-processing step added in which the tempo was doubled if it was estimated to be less than 80 BPM; the vast majority of the time,

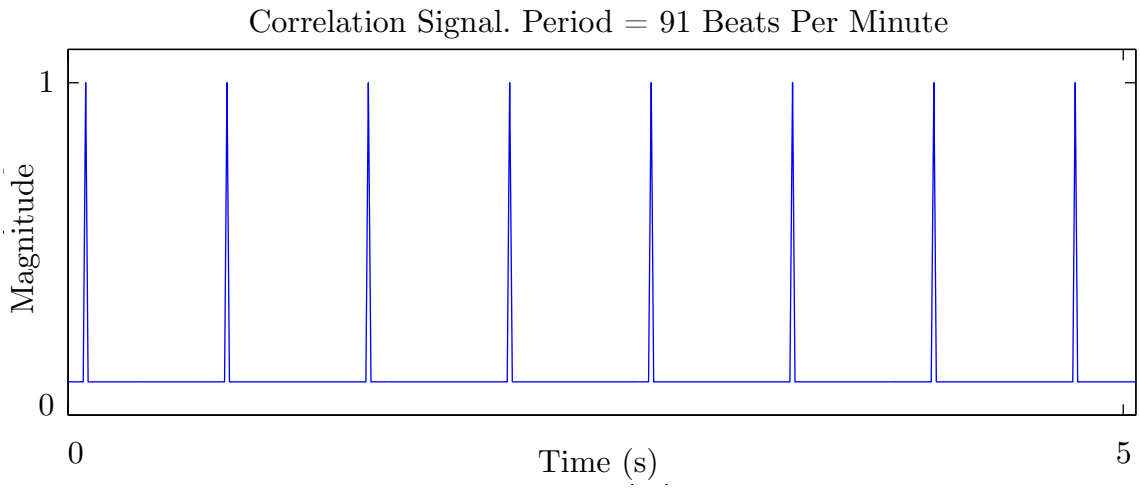

Figure 6.4: A correlation signal.

this was because the system had identified every other beat, so the true tempo was twice that which was marked. At the end of this process, both the beat component and the tempo had been estimated.

An example of this process used on an excerpt from the Dance Dataset is shown in Figure 6.5. On the top is an image of the correlations of the activation probabilities of each component with all the correlation signals. The maximal value occurs when component #2 is correlated with a correlation signal with a period of about .92 seconds, equivalent to a tempo of about 65 BPM. Because 65 BPM is less than 80, the tempo estimate is doubled to 130 BPM. Below the correlation image is a plot of the activation probability of component #2; the peaks in the signal are .45 seconds, apart, which is equivalent to a period of about 133 BPM. Both of these are very close to the true tempo value shown in Table B.1, 132 BPM.

6.4 Overall System

A flowchart of the overall system is shown in Figure 6.6.

The first several blocks of the flowchart show the portion of the system which

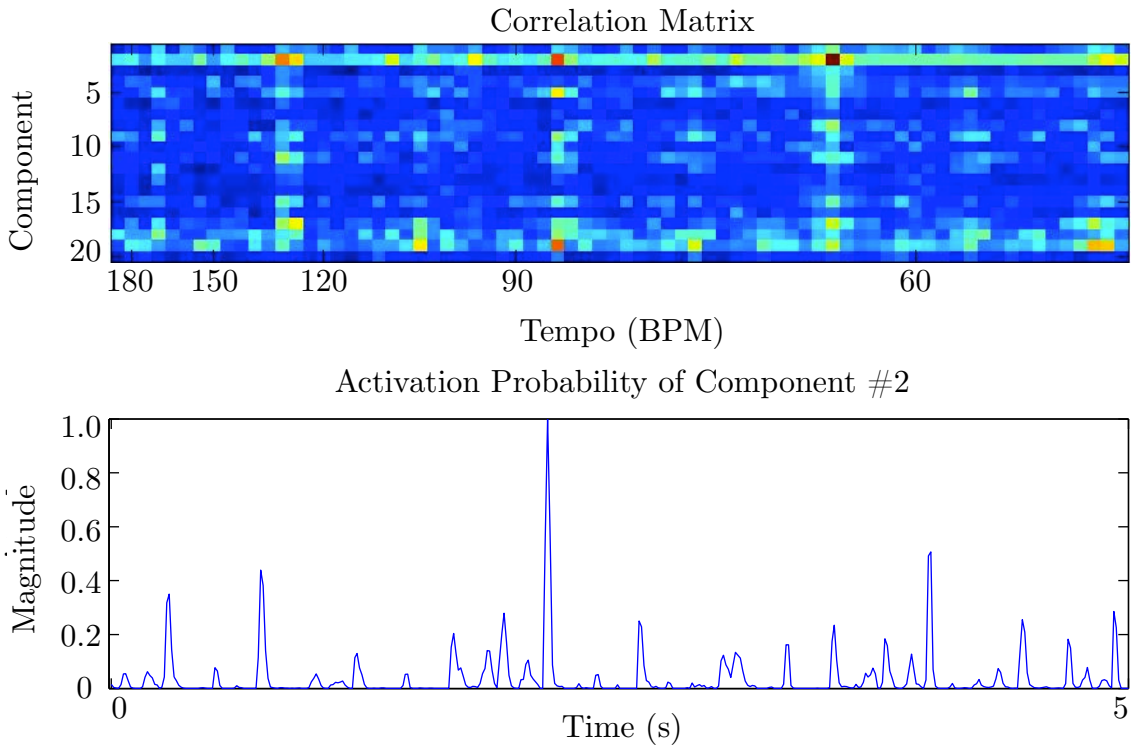

Figure 6.5: The correlation values between all activation probabilities from an excerpt of music and all correlation signals (top), and the activation probabilities of the component with the highest maximum correlation value over all correlation signals (bottom). Music was taken from 'Fame' by Irena Cara.

determines accent signal candidates. First, noisy audio is recorded by the system's microphones, such as the iRig MIC Cast or the Hubo lapel mikes. If multiple microphones are used, the signal is averaged into a monaural audio stream. The audio is represented as a stacked spectrogram, then passed into the HPSS and PLCA algorithms in order to first remove the harmonic part of the signal and then to decompose the percussive portion of the audio into its component elements. The spectral characteristics for these elements are potential beat spectral characteristics, and the activation probabilities are potential accent signals. Those signals are passed on to the next portion of the algorithm for further processing.

Both the HPSS and the PLCA algorithms require some parameterization. For

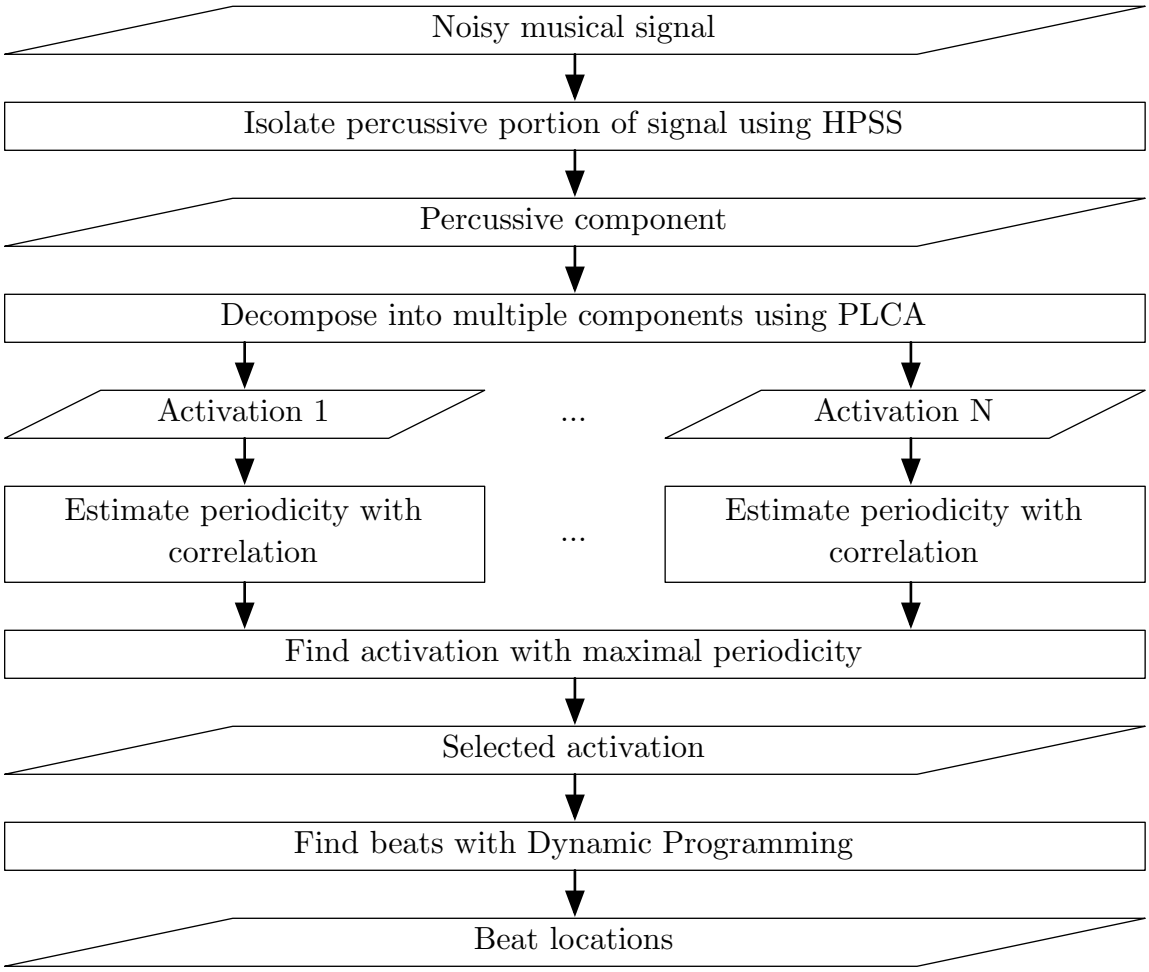

Figure 6.6: Flowchart of the final beat tracking algorithm.

HPSS, one parameter that can be varied is the power to which the harmonic- and percussive-suppressed spectrograms are raised when determining the soft masks. The mask equations can more generally be written:

$$M_{H_{f,t}} = \frac{H_{f,t}^p}{H_{f,t}^p + P_{f,t}^p} \tag{6.11}$$

$$M_{P_{f,t}} = \frac{P_{f,t}^p}{H_{f,t}^p + P_{f,t}^p} \tag{6.12}$$

Where p is any positive number. Smaller values of p result in a 'flattening' of the

signal, in which the distance between the largest and smallest values of H and P are reduced. This helps ensure that subtle, yet significant portions of those spectrograms are not overshadowed by more prominent parts. Larger values of p render the maximum values of H and P much more influential on the masks than the smaller values, which can help to overcome noise. This specific algorithm sets $p = 2$ empirically in order to allow the largest values in H and P, which are often attributable to the music and not the noise, to have heavy influence on the masks, while also not totally dismissing smaller values [46].

As for PLCA, the two most important parameters to set are the number of components, Z, and the number of iterations of the EM algorithm that run for each frame of audio. More components can result in a more accurate representation of each individual component, since there is less risk of multiple components being combined, but also linearly increases the amount of computation that must be performed. While some papers argue that setting $Z = 60$ is a good choice when using PLCA on audio spectrograms, this particular algorithm sets $Z = 20$, reducing computational requirements to one-third of the $Z = 60$ case [118]. As the results in Section 7 show, this still achieves state-of-the-art accuracy. The number of iterations through the Expectation and Maximization equations of the PLCA algorithm, N, was empirically set to 40. This was also found to provide accurate results while not wasting computational resources on additional, unneeded trips through the EM loop.

Subsequently, the accent signals are passed through the correlation portion of the system. The correlation values are considered to be proportional to the periodicity of the signal, and the activation probability with the highest correlation is judged to be the most periodic and thus the most beat-like. That signal, as well as its corresponding component, are passed on to the final portion of the algorithm, where beat locations themselves are estimated. This is done via dynamic programming,

which looks for maxima in the accent signal that are spaced according to the tempo. Such maxima are recorded as potential beat locations. Finally, as mentioned in Section 6.3, if the tempo is found to be below a certain threshold, it is doubled and beat locations are marked between every two beats that were already found. This helps prevent the system from finding only the downbeats or the offbeats in a musical work.

This portion of the system is also subject to parameterization. One important parameter is the neighborhood around the 'expected' next beat location where the system searches for a maximum value. Because each beat will not necessarily fall cleanly into a specific sample, and also because musicians will sometimes vary their tempo slightly from beat to beat (such as when classical musicians add *rubato* to their music), the system cannot simply take a beat location and skip exactly one period ahead to place the next beat. Instead, the system must search around the position that is one period ahead of the next beat, in case the actual frame containing a beat is slightly ahead or behind the expected value. The proposed system searches in a neighborhood that is 70ms wide, which was found to provide sufficient robustness against minor tempo variations. However, because it is still more likely that the next beat will be found exactly in the center of that window than at the edges, the system also includes a bias parameterization. The value in the center of the window will be chosen unless another frame is the maximum within the window by at least 3%. If the other frame is only just barely a maximum, then it is likely due to jitter or noise and not because that frame truly has a higher probability of possessing a beat than the center frame.

Because one goal for this system is that it be used to assist machines such as robots in responding to music, it is crucial that this system be able to function on short excerpts of music. If it needed to wait for the entire song, then the robot or

other device would not be able to do anything until the music was already over and it was too late. As such, the proposed algorithm system first observes the audio for five seconds and then performs the entire beat learning algorithm. Once the beat components have been identified and the beat locations for the first five seconds have been found, the system then listens to the rest of the piece and continues to perform the algorithm. As each new frame is added, the earliest remaining frame is discarded, maintaining a five-second window to be passed through the algorithm. Otherwise, the only difference between the initial 'training' step and the subsequent 'updating' step is that the frequency components, $P(f|z)$, are iterated over 40 times originally but only 2 times per new frame subsequently. This reduces computation and reflects the assumption that the frequency components comprising the audio are slowly varying.

In sum, the above algorithm is a novel system that takes in a musical signal over a noisy channel and processes it through the above steps in order to determine both beat locations and the spectral characteristics of the beats themselves. This system can form estimates of the beat spectral characteristics based on just five seconds of audio, enabling it to operate quickly on novel and unknown music. It is also computationally efficient and does not assume *a priori* what the spectral characteristics of either the noise or the beat components look like. It is therefore a flexible and robust system for the challenging task of beat learning in noisy environments.

7. Experiments and Results

The proposed beat-learning algorithm described in Section 6 was subsequently tested on the noisy audio datasets described in Section 4.1. By using the algorithm on different types of audio that were recorded in the presence of a variety of types of noise, these experiments could determine the accuracy of the algorithm in a wide variety of circumstances. This in turn would help show if it is robust to many types of noise or just a few specific nose sources. The metrics described in Section 5.2 were used in order to determine the overall accuracy of the algorithm.

7.1 Dance Dataset, PLCA Alone

The first experiments were to determine if PLCA itself was a viable procedure for decomposing a noisy musical audio stream into its component elements. As such, a stripped-down version of the algorithm was evaluated on noisy audio. Rather than using both HPSS and PLCA in order to identify the component elements of the audio, this reduced version of the algorithm only utilized PLCA in that first step. Additionally, it did not consider the 'update' portion of the algorithm and so only learned the beat component all at once using five-second clips of audio.

The stripped down algorithm was thus run on five-second excerpts from music in the Dance Dataset [63]. This dataset was considered to be easier to track than the General Dataset, with all the state-of-the-art algorithms achieving near-perfect accuracy when they tracked on clean audio from that set, and so was useful in order to determine if the proposed algorithm could work in the best case. The audio also contained noise from the Hubo robot as it moved its arms, in order to simulate the scenario in which the tracker was being used to synchronize a robot with a piece of

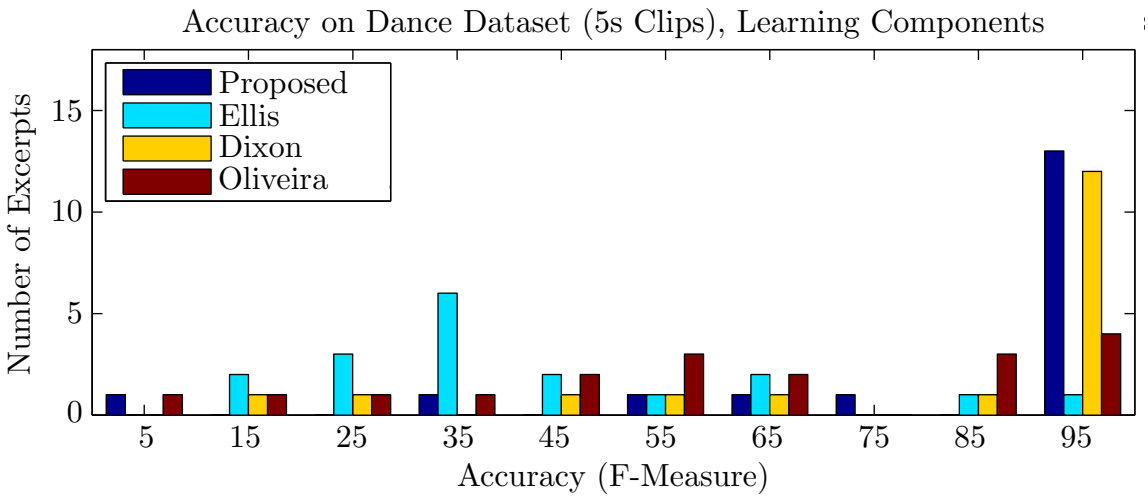

Figure 7.1: Results of the proposed and the comparison algorithms trained and run on 5-second excerpts from the Dance Dataset recorded in the presence of robot noise, in F-Measure.

music. The same audio was tracked by the state-of-the-art beat tracking algorithms denoted as *Ellis* and *Dixon* in Section 5.1 [34, 40]. Because this experiment would be run on audio played in the presence of a robot, another state-of-the-art beat tracker that was developed by Oliveira and Davies with the specific goal of enabling musical robots was compared to the proposed work as well [99]. This system, denoted as *Oliveira*, used the same accent signal and multiple-agent architecture as the Dixon tracker and also incorporates the state-recovery abilities of the Davies tracker. Because it was developed for robot audition, it was included as a fair comparison for this round of tests.

The results for this first round of experiments are shown in Figures 7.1 and 7.2 as well as Table 7.1 [63]. It is clear that the proposed system, even without HPSS, is able to outperform the off-the-shelf algorithms that it is being compared against. It surpasses the next most accurate system by over 4 points in F-Measure, and it outperforms the other two systems by far more than that. It does similarly well using

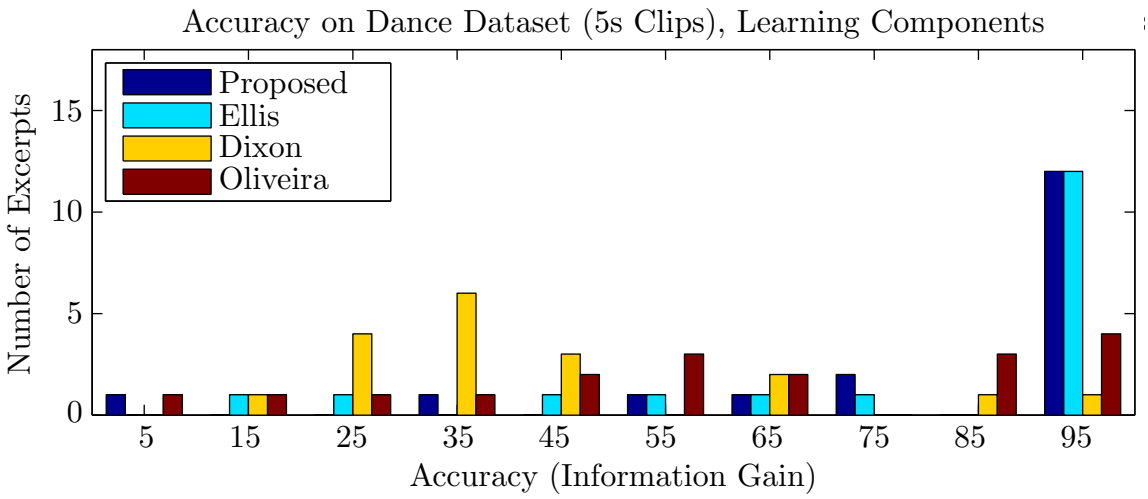

Figure 7.2: Results of the proposed and the comparison algorithms trained and run on 5-second excerpts from the Dance Dataset recorded in the presence of robot noise, in Information Gain.

Table 7.1: Average results of the proposed and the comparison algorithms trained and run on 5-second excerpts from the Dance Dataset recorded in the presence of robot noise.

Metric	Proposed	Ellis	Dixon	Oliveira
F-Measure	85.8	43.1	81.7	60.1
Information Gain	84.3	80.6	42.5	60.1

the Information Gain metric. While all the state-of-the-art algorithms can easily track the music with near perfect accuracy when it does not contain noise, the presence of noise degrades their accuracy substantially. Meanwhile, the proposed system remains relatively accurate. It is thus evident that the proposed system is better able to deal with noisy music.

The next evaluation was to determine whether or not the components were useful for subsequent frames in the audio. The basis of the updating portion of the system is that the components can be assumed to be slowly varying, but this assumption

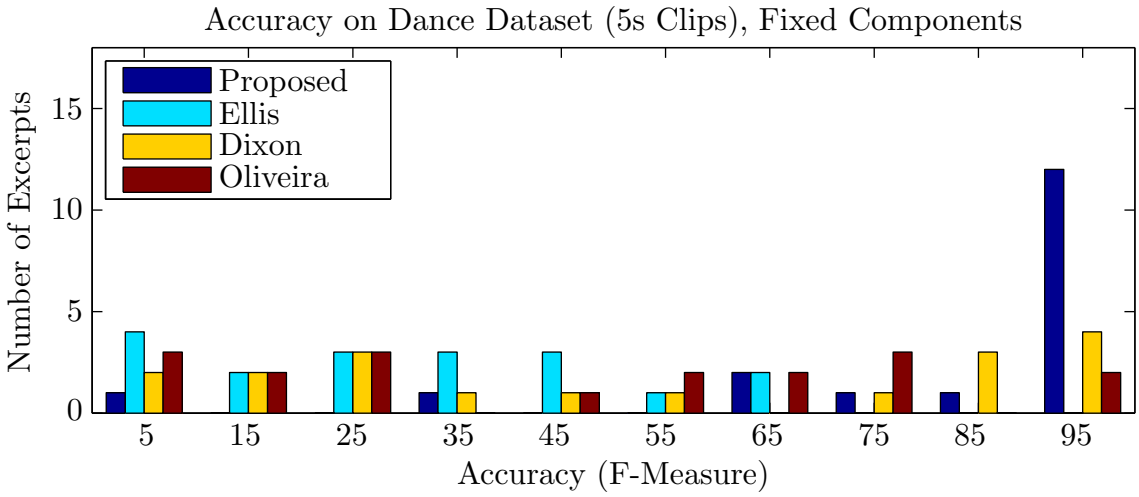

Figure 7.3: Results of the algorithms on 5-second excerpts from the Dance Dataset recorded in the presence of robot noise, in F-Measure. The proposed algorithm's components were learned on preceding 5-second excerpts.

Table 7.2: Average results of the proposed and the comparison algorithms run on 5-second excerpts from the Dance Dataset recorded in the presence of robot noise. Components were learned on preceding 5-second excerpts.

Metric	Proposed	Ellis	Dixon	Oliveira
F-Measure	84.2	29.7	53.3	44.2
Information Gain	81.0	50.2	35.1	47.3

had to be tested. As such, the components found in the first experiment were fixed, with $P(f|z)$ no longer updating, and the stripped-down algorithm was then run on the next five seconds of audio. The other trackers were tested on this audio as well. If accuracy degraded significantly, that would indicate that the components varied greatly even within a few seconds of audio, and so a more sophisticated updating algorithm would be needed later.

This second experiment's results are shown in Figures 7.3 and 7.4 and Table 7.2 [63]. Accuracy for the proposed algorithm only decreases by about 1.6 points in the

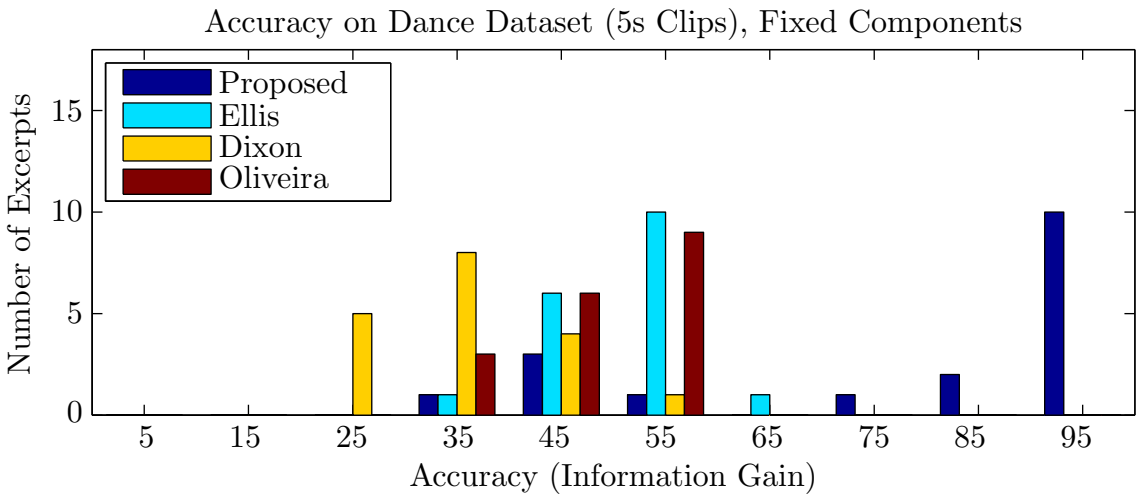

Figure 7.4: Results of the algorithms on 5-second excerpts from the Dance Dataset recorded in the presence of robot noise, in Information Gain. The proposed algorithm's components were learned on preceding 5-second excerpts.

F-Measure metric and 3 points in the Information Gain metric, while accuracy of the other systems falls precipitously. Even the Dixon tracker, which performed relatively well initially using the F-Measure Metric, and the Ellis tracker, which did well using the Information Gain metric, fails on music later in the piece. This demonstrates how the components that are learned by the proposed algorithm can indeed be used with substantial accuracy later in the music, which would indicate the power of the slowly-updating approach used in the full algorithm. Furthermore, the ability of the proposed algorithm to surpass other systems is again displayed.

7.2 Dance Dataset, Entire Algorithm

After verifying that PLCA was a viable option for identifying components, the full algorithm was evaluated. In particular, the HPSS algorithm described in Section 6.2 was incorporated into the algorithm being tested, as was the updating system described in Section 6.4. In order to verify the ability of the full algorithm to learn

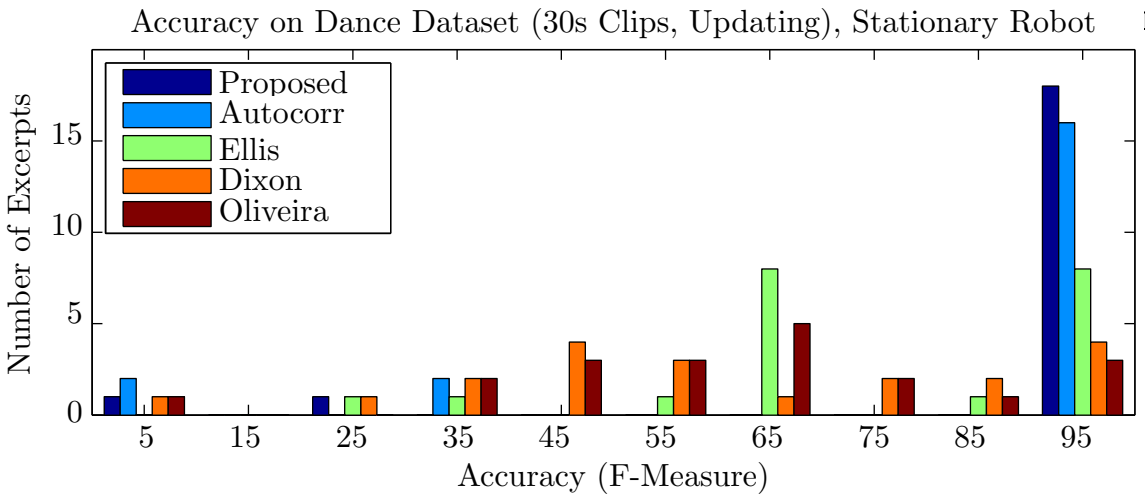

Figure 7.5: Results of the proposed and the comparison algorithms trained and run on 30-second excerpts from the Dance Dataset recorded in the presence of stationary robot noise, in F-Measure.

and locate beats in larger pieces of music, it was then run on larger sections of the noisy audio in the Dance Dataset.

Two sets of noisy audio were obtained, one in which the Dance Dataset contained both room noise and the base noise of the robot (such as fans and computers), and one in which the robot was also moving and contributing ego noise, as in the previous experiment [57]. The audio was then passed through the same trackers as in the previous round of experiments and analyzed according to the specified metrics. Additionally, in order to evaluate the decision to use a correlation-based method to find the beat component rather than an autocorrelation-based method, a version of the algorithm using autocorrelation instead of correlation with an impulse train was also implemented and tested with the others. This algorithm is denoted *Autocorr* in this paper.

The results of the experiments conducted on the audio containing noise from the stationary Hubo robot and the room, but not moving motors, are shown in Fig-

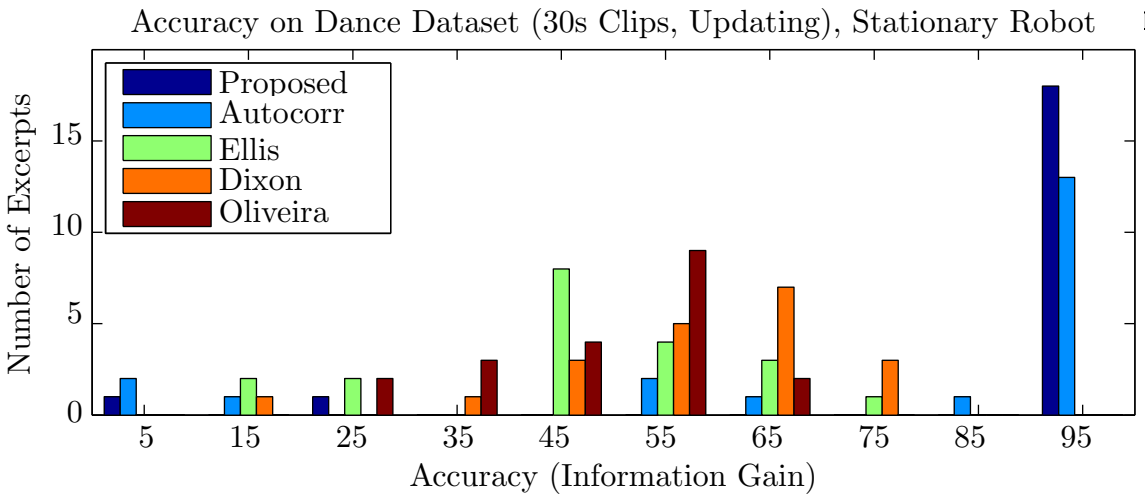

Figure 7.6: Results of the proposed and the comparison algorithms trained and run on 30-second excerpts from the Dance Dataset recorded in the presence of stationary robot noise, in Information Gain.

Table 7.3: Average results of the proposed and the comparison algorithms trained and run on 30-second excerpts from the Dance Dataset recorded in the presence of stationary robot noise.

Metric	Proposed	Autocorr	Ellis	Dixon	Oliveira
F-Measure	91.4	82.9	77.1	61.1	60.9
Information Gain	91.6	78.8	47.1	55.9	48.1

ures 7.5 and 7.6 as well as Table 7.3 [57]. The proposed system as well as the autocorrelation-based variant of that system again outperform the comparison trackers on this dataset, achieving F-Measure scores of at least 5 points above the next best tracker, and Information Gain scores even better in comparison to the results of the others. Additionally, the proposed system does consistently outperform the autocorrelation-based system. This is likely due to several factors, including the noise distorting the activation probabilities of the components to such a degree that the autocorrelations are themselves noisy and unreliable.

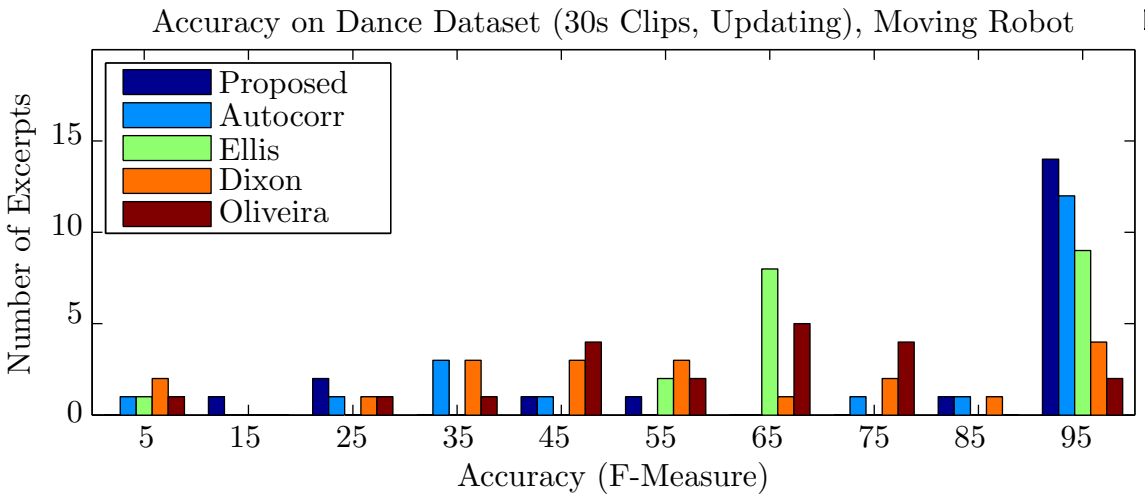

Figure 7.7: Results of the proposed and the comparison algorithms trained and run on 30-second excerpts from the Dance Dataset recorded in the presence of moving robot noise, in F-Measure.

Table 7.4: Average results of the proposed and the comparison algorithms trained and run on 30-second excerpts from the Dance Dataset recorded in the presence of moving robot noise.

Metric	Proposed	Autocorr	Ellis	Dixon	Oliveira
F-Measure	82.5	77.1	76.3	55.7	58.5
Information Gain	75.4	65.5	44.7	48.5	44.5

The superior performance of the proposed algorithm, as well to a certain extent the autocorrelation-based algorithm, demonstrates the utility of a component-based approach to finding beats in noisy audio. The other systems all use hand-crafted features which can very easily be thrown astray by the presence of loud noises, with the result that the systems produce inaccurate results. By decomposing the audio and attempting to locate a particular component which corresponds to a beat, though, more accurate results are achievable, even in the presence of noise.

The results for the same tests run on the audio that was recorded while the robot

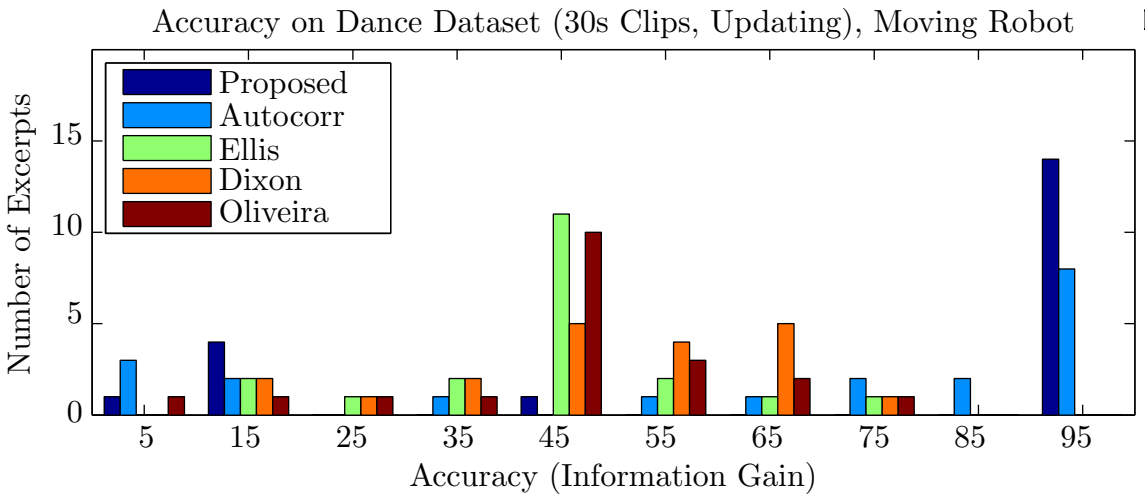

Figure 7.8: Results of the proposed and the comparison algorithms trained and run on 30-second excerpts from the Dance Dataset recorded in the presence of moving robot noise, in Information Gain.

was moving its arms are displayed in Figures 7.7 and 7.8, and also Table 7.4. All the algorithms perform worse on this audio, which is expected because the noise is more significant. Not only are the robot motors noticeably loud, but because the arms are moving periodically, they provide another signal which a beat learning and tracking algorithm might pick up instead of the desired musical signal. However, the proposed algorithm still does better than all the others, followed closely by the autocorrelation variant of the proposed system.

Once again, the results show that the usual method of using hand-crafted features does not work as well when used on noisy audio. The features in the off-the-shelf algorithms are less able to function as accent signals with extrema close to the beats than the proposed system, which learns the beats from the audio directly and derives an accent signal from the learned spectral characteristics. Furthermore, as the algorithm is powerful enough to learn the beats even in the presence of substantial noise, overall accuracy, while not quite at the level which could be achieved on clean audio,

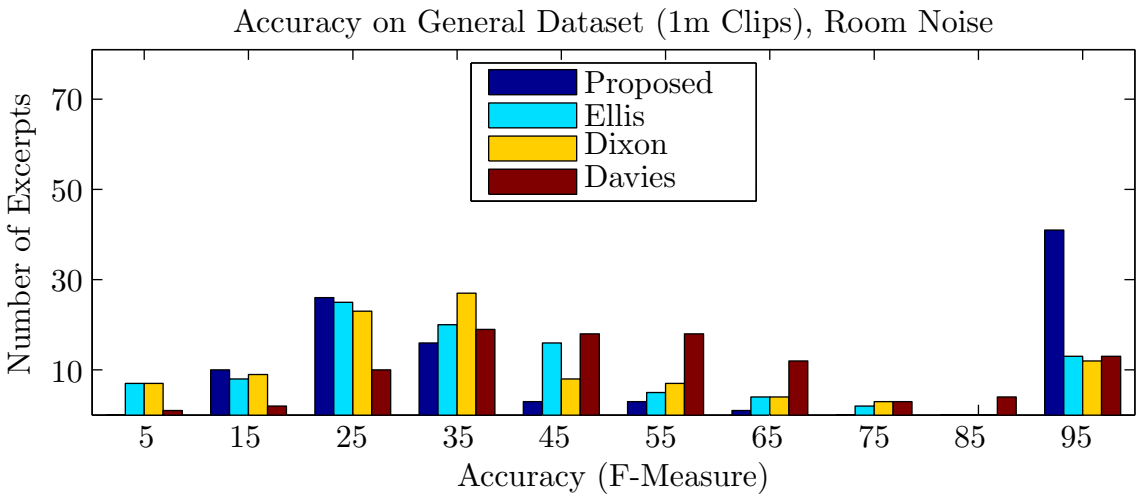

Figure 7.9: Results of the proposed and the comparison algorithms trained and run on 60-second excerpts from the General Dataset recorded in the presence of room noise, in F-Measure.

is still much better than if another system was used.

This round of experiments demonstrated that the proposed algorithm produces state-of-the-art results for beat tracking, at least for dance music with beats that are steady and prominent. In order to determine if this system could perform well on other types of music, however, further experiments were conducted.

7.3 General Dataset, Room Noise

The proposed algorithm was used to locate and learn beats for 1-minute excerpts of the 100 songs in the General Dataset. Additionally, the general-purpose beat trackers described in Section 5.1 (*Ellis*, *Dixon*, and *Davies*) were also run on the same data. In this way, not only could the system be evaluated on still longer excerpts of music in order to test the robustness of the algorithm, but the multiple genres allow for the examination of which types of music the proposed algorithm works well on, and which it does not.

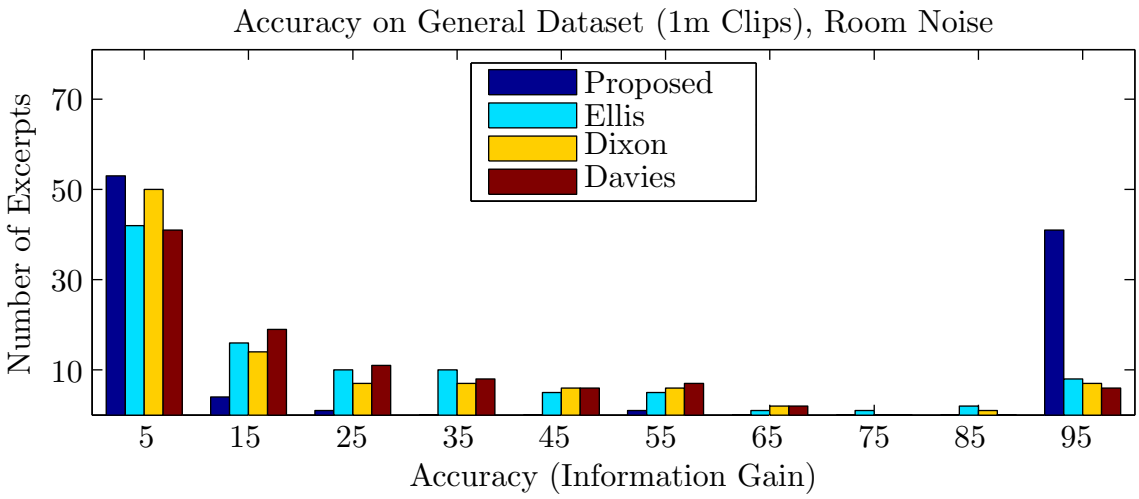

Figure 7.10: Results of the proposed and the comparison algorithms trained and run on 60-second excerpts from the General Dataset recorded in the presence of room noise, in Information Gain.

The overall results of this experiment are shown in Figures 7.9 and 7.10. While performance is not as great as it is for the Dance Dataset, it can still be observed that the system continues to outperform the off-the-shelf algorithms. Some of this music is very challenging to track, particularly music from the classical and jazz genres, which lack many of the elements that made the Dance Dataset more tractable such as consistent, steady, and prominent beats. However, despite these difficulties, the proposed algorithm is still superior to the others. This demonstrates the overall utility of the system, as it can still function more accurately than others while tracking challenging music in noisy environments.

Additionally, the results broken down by genre are shown in Tables 7.5 and 7.6. These tables make it clear that the proposed algorithm is better at tracking some genres than others. Hip-hop and soul do particularly well, with F-Measure scores of 84.5 and 72.6 and Information Gain scores of 81.6 and 67.4 respectively. The music found for these genres tends to include prominent steady beats without too many

Table 7.5: Average results of the proposed and the comparison algorithms trained and run on 60-second excerpts from the General Dataset recorded in the presence of room noise. Results are in F-Measure.

Genre	Proposed	Ellis	Dixon	Davies
Classical	**44.1**	37.8	34.0	39.3
Country/Western	**49.7**	26.1	27.9	41.7
Dance	64.0	62.6	58.6	**72.8**
Folk	40.4	34.2	47.0	**57.3**
Hip-Hop	**84.5**	40.8	32.0	56.7
Jazz	37.1	34.5	30.6	**45.2**
Metal	**65.5**	43.8	46.4	48.9
Pop	**61.8**	41.3	40.5	60.7
Rock	**66.5**	46.8	53.2	59.4
Soul	**72.6**	46.4	40.0	52.7
Total	**58.6**	41.4	41.0	53.5

other strong sound sources to detract from them. For much of the selected music of these genres, the beats are the loudest and most prominent parts, which allows for the proposed system to more easily separate the beats from everything else and successfully perform beat tracking. The system still does relatively well for other genres that tend to have steady beats, such as Dance, Metal, Pop, and Rock. It does the most poorly on genres such as Classical and Jazz, which often have irregular beats, as well as Country/Western and Folk, which tends to have softer beats than the other genres.

It is worth noting that the proposed system does not win for every single genre. In terms of F-Measure, for example, the Davies tracker outperforms it for the Dance genre. A likely explanation for this is that the feature used in the Davies system, the complex spectral difference, was especially impacted by the beats in those songs, which tend to have a particularly bombastic quality. This feature was still an accurate indicate of the presence of beats even after noise was added due to the beats still being

Table 7.6: Average results of the proposed and the comparison algorithms trained and run on 60-second excerpts from the General Dataset recorded in the presence of room noise. Results are in Information Gain.

Genre	Proposed	Ellis	Dixon	Davies
Classical	**23.2**	13.7	13.9	3.1
Country/Western	**34.1**	9.3	8.3	14.6
Dance	**54.5**	47.1	44.6	42.3
Folk	14.2	8.5	**23.9**	21.4
Hip-Hop	**81.6**	44.0	28.8	29.7
Jazz	14.0	18.3	6.9	**18.0**
Metal	**54.1**	35.3	24.9	22.0
Pop	**53.1**	31.7	33.6	32.2
Rock	**54.1**	33.8	30.7	32.0
Soul	**67.4**	26.2	22.0	21.6
Total	**45.0**	26.8	23.7	23.7

loud and that feature being so heavily influenced on those particular beats. However, the proposed system does win for a majority of the genres, and furthermore has the highest average accuracy overall. Also, the proposed system wins more consistently in terms of Information Gain, which tends to be more meaningful for lower scores (while F-Measure is more meaningful at higher scores). Since the achieved scores of the system are generally substantially less than 100, Information Gain is likely the more meaningful metric in this situation, and by that standard, the proposed system achieves superior results for almost every genre.

The results of this experiment show that the proposed system produces state-of-the-art results on an audio corpus taken from a wide variety of genres. It is thus more useful in cases where the exact genre of the music that will be played is not known, as might be the case in a bar or club where the music being played through the speakers can vary dramatically from song to song. This also indicates that the system is robust enough that it can learn and locate different types of beats, instead

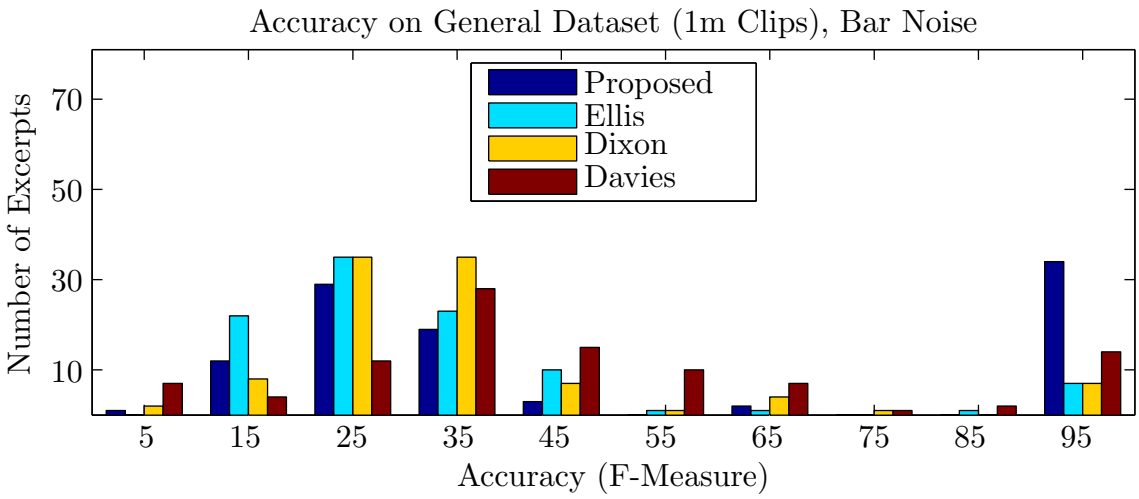

Figure 7.11: Results of the proposed and the comparison algorithms trained and run on 60-second excerpts from the General Dataset recorded in the presence of bar noise, in F-Measure.

of being restricted to only being able to find beats that heavily influence a particular hand-crafted feature. In short, these results demonstrate that the system is robust both to noise and to the variations that occur between songs from several different genres. Furthermore, for genres such as Hip-Hop and Soul, these results demonstrate that the system is highly accurate and can already track such music well even in these noisy conditions.

7.4 General Dataset, Bar Noise

Finally, the same trackers are tested on a much harder dataset. The General Dataset was recorded in the presence of loud noises recorded at a bar. This scenario most closely mimicked one of the conditions for which this system is intended to be used, namely, analysis of noisy music played at a public venue where many people are gathered, such as a club or restaurant. Such noises would include not just room effects, but also people talking, silverware and plates clattering, and a myriad of other

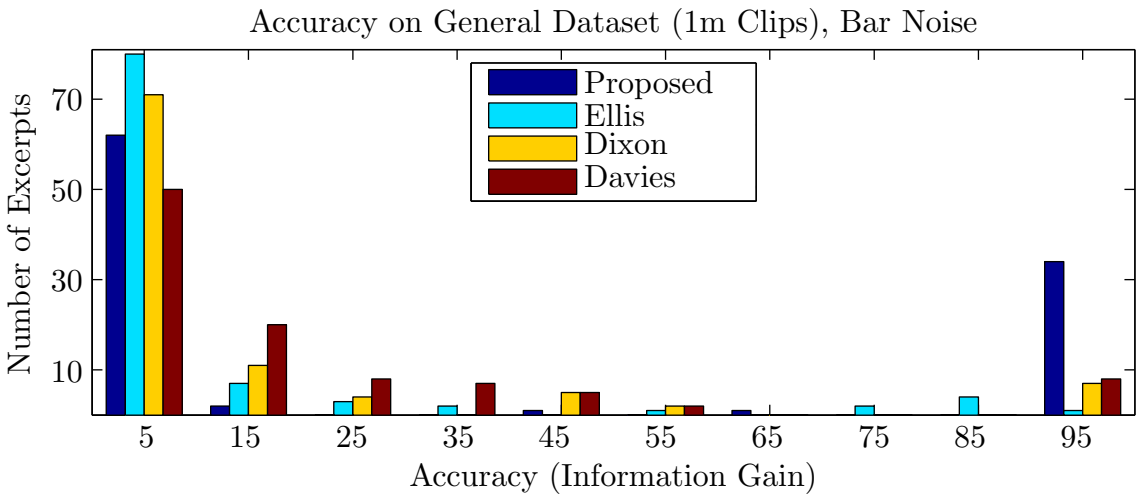

Figure 7.12: Results of the proposed and the comparison algorithms trained and run on 60-second excerpts from the General Dataset recorded in the presence of bar noise, in Information Gain.

sounds which were not present in the other experiments. This test was performed to determine if the system could produce state-of-the-art results in this particular case, and thus if it was effective at one of the overall goals of the project.

The results are shown in Figures 7.11 and 7.12. It is clear that the accuracies of all of the algorithms have been reduced compared to the other cases; the bar noise is louder and more obstructive than the noises produced by the room itself, as indicated by the Sound Pressure Level of the recorded bar sounds being greater than the other noise sources, and so accuracy falls in response. However, the proposed system is still able to achieve better results than the others that are tested. For both Information Gain and F-Measure, roughly 30 of the 100 songs produce scores in the top range of the presented histograms, which represent scores between 90 and 100. None of the other algorithms even come close. This shows that the proposed algorithm, though not as accurate as before, is still of utility for this very noisy dataset. Thanks to the process of performing decomposition operations on the noisy music in order to

Table 7.7: Average results of the proposed and the comparison algorithms trained and run on 60-second excerpts from the General Dataset recorded in the presence of bar noise. Results are in F-Measure.

Genre	Proposed	Ellis	Dixon	Davies
Classical	27.8	**46.6**	39.1	39.5
Country/Western	**41.1**	32.8	28.8	35.2
Dance	**70.6**	36.9	44.8	68.9
Folk	45.7	31.5	40.1	**54.4**
Hip-Hop	**77.5**	37.9	40.7	45.9
Jazz	42.1	34.6	31.6	**49.3**
Metal	**57.8**	25.6	38.9	43.3
Pop	**53.5**	29.4	40.9	45.3
Rock	**53.7**	30.1	25.1	37.0
Soul	**54.2**	33.6	36.4	46.8
Total	**52.4**	33.9	36.6	46.6

learn the beats, it can surpass the systems using hand-crafted features, even in the presence of intense noise.

The genre breakdown of these same results is shown in Tables 7.7 and 7.8. As before, the system performs better in some genres than others. Hip-Hop continues to track very well with this algorithm, as does Dance, and genres such as Metal, Pop, Rock, and Soul are not far behind. Furthermore, for these genres the proposed system outperforms the off-the-shelf trackers in every case. This shows that, even in the presence of heavy noise, the algorithm can still produce results that are better than what currently exists, and thus is more useful for the purpose of accurately tracking and analyzing noisy music, the type that a great many humans listen to frequently in restaurants, bars, and clubs.

It is true that the results are less accurate in this test than in the preceding experiment, both for the proposed tracker as well as for the off-the-shelf systems that my algorithm is being compared to. However, though the increased noise does in

Table 7.8: Average results of the proposed and the comparison algorithms trained and run on 60-second excerpts from the General Dataset recorded in the presence of bar noise. Results are in Information Gain.

Genre	Proposed	Ellis	Dixon	Davies
Classical	4.6	**29.9**	15.2	4.6
Country/Western	**24.2**	12.3	5.6	16.0
Dance	**62.7**	15.5	22.6	40.9
Folk	22.9	4.7	14.0	**24.5**
Hip-Hop	**73.2**	19.2	21.9	19.3
Jazz	23.9	13.6	4.9	**24.6**
Metal	**42.5**	4.9	14.1	9.0
Pop	**43.0**	8.2	30.5	19.2
Rock	**43.5**	15.8	11.5	26.9
Soul	**44.1**	8.9	23.7	23.4
Total	**38.5**	13.3	16.4	20.8

turn force a decrease in accuracy, the proposed system is still able to operate more accurately than the other algorithms. This is further support of the fundamental idea of the system, that it is useful to decompose an acoustic stream and attempt to find one or more components representing the beat, rather than simply relying on hand-selected features which, in clean audio, tend to correlate well with beat locations. Since even the heavy noise of the bar was unable to completely erase all of the beat information from the signal, the proposed system was able to extract that information out for some of the songs and use it both to learn the beats and to find them in the music.

In summary, these results as well as all the others indicate that the proposed algorithm is able to achieve state-of-the-art accuracy on many different types and genres of audio in a wide variety of acoustic conditions. Whether the system processes music that is recorded in the presence of room noises, a stationary robot, a moving robot, or the loud and chaotic sounds of a bar, it is able to outperform other algorithms

and achieve superior results. As a result, for the vast amount of audio that humans listen to which is noisy–ranging from music played in bars to music used in conjunction with loud machines–the proposed algorithm is more suitable for learning the beats and locating them than the other evaluated systems.

8. Conclusions

This chief contribution of this thesis to the scientific literature is the proposed beat learning algorithm, which has been demonstrated to be capable of locating and learning beats in noisy acoustic environments more accurately than current systems. These results have proven to be the case over a wide variety of noise sources that range from the chatter at a bar to the motors of a moving robot. The algorithm can also operate on as little as five seconds of audio, making it useful for live environments. Lastly, the system was proven to perform at this level over several different genres, demonstrating its flexibility.

One explanation for the proposed algorithm's success could be that it considers the problem of locating beats in a fundamentally different way than other state-of-the-art beat trackers. Most modern beat trackers use hand-crafted features, such as the spectral energies at different subbands with specified widths and central frequencies, which they assume *a priori* are good indicators as to whether or not a beat exists in any given frame. When they encounter audio signals for which those features are poorly correlated with the presence of beats, the trackers are less likely to work. The proposed algorithm, by contrast, makes fewer assumptions about what a beat 'should' look like. It instead uses PLCA to learn the spectral characteristics of the beat directly from the audio. This allows for the system to be more flexible than other beat trackers, and may help explain why it performs better than the others.

It should be noted that this particular system may more closely parallel with how humans learn beats than the conventional methods. Humans can often listen to completely novel music and pick out beats even if they do not know beforehand what instrument is keeping the beat or which features are most strongly correlated with beats in that music. Rather than approaching the music with some exact specification

of what features a beat must have, they can often listen to the audio and identify consistent repeated elements which have a temporal structure, and then subsequently work out what beats sound like in that particular piece. Humans are often able to perform this task even while in noisy environments, which is another indicator that part of the reason for this algorithm's success may be because it parallels this strategy.

Another contribution to the scientific literature of this thesis is the dataset of annotated audio recorded in the presence of noise. No such dataset existed before this project, with the result that beat detection algorithms being developed instead used audio taken directly from digital files that contained very little noise. The existance of this dataset will allow myself as well as other researchers to continue developing systems that are robust to noise and thus able to process more of the audio that humans listen to and enjoy than existing systems. The audio recorded in conjunction with robot noise could be used in particular to assist in developing Music-IR algorithms that can function in the presence of noisy robots.

Lastly, the evaluation of the proposed system and others on the annotated audio dataset is another contribution to the scientific literature. By establishing a useful baseline, these results can be used to benchmark further algorithms that attempt to solve this problem. The breakdown of the results by genre and noise source also help to identify in exactly which situations further improvement is most needed.

8.1 Future Work

One direction to look for future work is to incorporate multiple components into the analysis. If a piece of music contains polyrhythms, for example, then a single beat component might either not capture all of the beats. This could be due to multiple instruments playing the different beats, the unusual rhythm appearing less periodic than more traditional rhythmic structures, or other factors. By using more

components and finding more potential beat candidates, a system that is more robust to polyrhythms could be developed.

Another possibility would be to perform more analysis of the spectral characteristics of the beats in order to analyze the rhythmic patterns of the music. For instance, if a piece of music has different instruments playing different beats, the spectral characteristics would be likely to reflect this and could be used to help classify beats or determine the higher-level rhythmic structure of the audio. This information in turn would be useful for a variety of purposes, from automatic segmentation and analysis of music to automatic remixing, and so exploiting the spectral characteristics in this manner could yield useful results.

A third option for future work is to implement this system as a Robot Operating System (ROS) module.[1]. ROS is a framework for writing robot software that is used for a variety of robots. Adding this system to ROS would enable it to be more easily used by various robotics researchers, enabling them to obtain the benefits from this work. Additionally, the algorithm could be paired with other ROS modules, such as systems enabling a robot to turn its head and direct its ears to focus at a sound source, further increasing the accuracy of the system.

8.2 Potential Applications

There are numerous applications for this work. One such application is to allow for standard Music-IR tasks, such as playlist generation, to be performed on audio that was recorded in relatively noisy environments. Databases like the Live Music Archive, which includes over 100,000 songs, contain a great deal of music that was recorded in live, noisy conditions.[2] Noise-robust beat tracking would open all this music up for the various Music-IR tasks for which beat tracking is essential, from automatically

[1] `http://www.ros.org/`

[2] `https://archive.org/details/etree`

selecting music at a particular tempo to automatic remixing. This music could also become available for use by DJs and other performers who need algorithms to perform tasks such as automatic beat synchronization, but who couldn't perform that task accurately on live audio with prior systems.

This algorithm could also be used in a variety of circumstances in bars, clubs, and other noisy venues in which music is played. As an example, many bars host live music performances that can be augmented with backing tracks or computer-generated effects, but manually synchronizing the tracks and effects to the human musicians can be difficult and tedious. The use of this system could determine the beat positions in the live music and communicate this to the computer, allowing for automatic synchronization.

To further augment performances, this algorithm could increase the ability of robots to interact with musical performances. In order to accompany or otherwise react to human musicians, such robots would need to be able to listen to those musicians and find the beat locations in their music to synchronize themselves with the humans. This beat learning system, which is more robust to robot noise than other algorithms, could help enable robots to suss out the beat locations even in the presence of their own noise and thus play more coherently with humans.

Ultimately, the major advantage of this system is that it enables us to perform Music-IR tasks on more of the music that humans already hear. We don't only listen to music that was recorded in pristine conditions and supplied to us through high-quality speakers in quiet rooms, so our systems that help us enjoy our music more, such as by automating tedious chores such as beat synchronization, intelligently recommending new music to us, or automatically adding effects and augmentation to make our musical performances more interesting, shouldn't be restricted to studio-quality audio either. Just as we humans often enjoy noisy audio, our tools should be

able to function on that audio and help us to derive even more enjoyment from that music. This algorithm is a key component in enabling Music-IR on such audio.

Bibliography

[1] Alex Acero, Neal Bernstein, Rob Chambers, Yun-Cheng Ju, Xiao Li, Julian Odell, Patrick Nguyen, Oliver Scholtz, and Geoff Zweig. Live search for mobile: Web services by voice on the cellphone. In *Proceedings of the International Conference on Acoustic, Speech, and Signal Processing*, Las Vegas, USA, March-April 2008.

[2] Masoud Alghoniemy and Ahmed Twefik. Rhythm and periodicity detection in polyphonic music. In *Proceedings of the IEEE Workshop on Multimedia Signal Processing*, pages 185–190, Copenhagen, Denmark, September 1999.

[3] Paul Allen and Roger Dannenberg. Tracking musical beats in real time. In *Proceedings of the International Computer Music Conference*, pages 140–143, Glasgow, United Kingdom, September 1990.

[4] Miguel Alonso, Roland Badeau, Bertrand David, and Gael Richard. Musical tempo estimation using noise subspace projections. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 95–98, New Paltz, NY, USA, October 2003.

[5] Miguel Alonso, Gael Richard, and Bertrand David. Accurate tempo estimation based on harmonic + noise decomposition. *EURASIP Journal on Applied Signal Processing*, 2007, January 2006.

[6] Janet Baker, Li Deng, James Glass, Sanjeev Khudanur, Chin-Hui Lee, Nelson Morgan, and Douglas O'Shaughnessy. Research developments and directions in speech recognition and understanding, part 1. *IEEE Signal Processing Magazine*, 26(3):75–80, May 2009.

[7] Shumeet Baluja and Michele Covell. Content fingerprinting using wavelets. In *Proceedings of the 3rd European Conference on Visual Media Production*, pages 198–207, London, United Kingdom, November 2006.

[8] Mark Bartsch and Gregory Wakefield. Audio thumbnailing of popular music using chroma-based representations. *IEEE Transactions on Multimedia*, 7(1):96–104, 2005.

[9] Alyssa Batula, Manu Colacot, David Grunberg, and Youngmoo Kim. Using audio and haptic feedback to detect errors in humanoid musical performances. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 295–300, Daejeon, South Korea, May 2013.

[10] Juan Bello and Jeremy Pickens. A robust mid-level representation for harmonic content in music signals. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 304–311, London, United Kingdom, September 2005.

[11] Michael Berouti, Richard Schwartz, and John Makhoul. Enhancement of speech corrupted by acoustic noise. In *Proceedings of the International Conference on Acoustic, Speech, and Signal Processing*, Washington, D.C., USA, April 1979.

[12] Steven Boll. Suppresion of acoustic noise in speech using spectral subtraction. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(2):113–120, April 1979.

[13] Andrei Broder. On the resemblance and containment of documents. In *Proceedings of the Conference on Compression and Complexity of Sequences*, pages 21–29, Salerno, Italy, June 1997.

[14] Judith Brown. Determination of the meter of musical scores by autocorrelation. *Journal of the Acoustic Society of America*, 94(4):1953–1957, October 1993.

[15] T. Tony Cai. Adaptive wavelet estimation: A block thresholding and oracle inequality approach. *The Annals of Statistics*, 27(3):898–924, June 1999.

[16] Olivier Cappe. Elimination of the musical noise phenomenon with the ephraim and malah noise suppressor. *IEEE Transactions on Speech and Audio Processing*, 2(2):345–349, April 1994.

[17] Chris Chafe, Bernard Mont-Reynaud, and Loren Rush. Toward an intelligent editor of digital audio: Recognition of musical constructs. *Computer Music Journal*, 6(1):30–41, March 1982.

[18] Praphul Chandra and David Lide. *Wi-Fi Telephony: Challenges and Solutions for Voice over WLANs*. Newnes, 2007.

[19] Vijay Chandrasekhar, Matt Sharifi, and David Ross. Survey and evaluation of audio fingerprinting schemes for mobile query-by-example applications. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 801–806, Miami, USA, October 2011.

[20] Vimala Chinnaraj and V. Radha. A review on speech recognition challenges and approaches. *World of Computer Science and Information Technology Journal*, 2(1):1–7, 2012.

[21] John Chowning, Loren Rush, Bernard Mont-Reynaud, Chris Chafe, W. Andrew Schloss, and Julius Smith. Intelligent systems for the analysis of digitized acoustic signals. Technical Report STAN-M-16, Stanford University: Center for Computer Research in Music and Acoustics: Department of Music, January 1984.

[22] Joseph Chung. *An Agency for the Perception of Musical Beats, or If I Only Had a Foot...* PhD thesis, Massachusetts Institute of Technology, June 1989.

[23] Israel Cohen. Speech enhancement using a noncausal a priori SNR estimator. *IEEE Signal Processing Letters*, 11(9):725–728, September 2004.

[24] Israel Cohen and Baruch Berdugo. Noise estimation by minima controlled recursive averaging for robust speech enhancement. *IEEE Signal Processing Letters*, 9(1):12–15, January 2002.

[25] Thomas Cover and Joy Thomas. *Elements of Information Theory*, chapter 2. John Wiley & Sons Ltd, 1991.

[26] George E. Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42, January 2012.

[27] Matthew Davies, Norberto Degara, and Mark Plumbley. Evaluation methods for musical audio beat tracking algorithms. Technical Report C4DM-TR-09-06, Queen Mary, October 2009.

[28] Matthew Davies and Mark Plumbley. Context-dependent beat tracking of musical audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(3):1009–1020, March 2007.

[29] Steven Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, August 1980.

[30] Bert de Vries, Chi Wei Che, Roger Crane, Jim Flanagan, Qiguang Lin, and John Pearson. Neural network speech enhancement for noise robust speech recognition. In *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications*, Sweden, May 1995.

[31] Li Deng, Alex Acero, Mike Plumpe, and Xuedone Huang. Larve-vocabulary speech recognition under adverse acoustic environments. In *Proceedings of the International Conference on Spoken Language Processing*, pages 806–809, Beijing, China, October 2000.

[32] Simon Dixon. A beat tracking system for audio signals. In *Proceedings of the Conference on Mathematical and Computational Methods in Music*, pages 101–110, Vienna, Austria, December 1999.

[33] Simon Dixon. Onset detection revisited. In *Proceedings of the International Conference on Digital Audio Effects*, pages 133–137, Montreal, Canada, September 2006.

[34] Simon Dixon. Evaluation of the audio beat tracking system BeatRoot. *Journal of New Music Research*, 36(1):39–50, March 2007.

[35] Simon Dixon, Fabien Gouyon, and Gerhard Widmer. Towards characterisation of music via rhythmic patterns. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 509–516, Barcelona, Spain, October 2004.

[36] J. Stephen Downie. Music information retrieval. *Annual Review of Information Science and Technology*, 37(1):295–340, 2003.

[37] Anthonie Driesse. Real-time tempo tracking using rules to analyze rhythmic qualities. In *Proceedings of the International Computer Music Conference*, Montreal, Canada, October 1991.

[38] Douglas Eck. Beat tracking using an autocorrelation phase matrix. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages IV1313–IV1316, Honolulu, USA, April 2007.

[39] Robert Ellenberg, David Grunberg, Paul Oh, and Youngmoo Kim. Using miniature humanoids as surrogate research platforms. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2009.

[40] Daniel Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, March 2007.

[41] Yariv Ephraim and David Malah. Speech enhancement using optimal non-linear spectral amplitude estimation. In *Proceedings of the International Conference on Acoustic, Speech, and Signal Processing*, Boston, USA, April 1983.

[42] Yariv Ephraim and David Malah. Speech enhancement using a-minimum mean-square error short-time spectral amplitude estimator. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(6):1109–1121, December 1984.

[43] Yariv Ephraim and David Malah. Speech enhancement using a minimum mean-square error log-spectral amplitude estimator. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 33(2):443–445, April 1985.

[44] Antti Eronen and Anssi Klapuri. Music tempo estimation with k-NN regression. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(1):50–57, January 2010.

[45] William Fisher, George Doddington, and Kathleen Goudie-Marshall. The DARPA speech recognition research database: Specifications and status. In *Proceedings of DARPA Workshop on Speech Recognition*, pages 93–99, 1986.

[46] Derry FitzGerald. Harmonic/percussive separation using median filtering. In *Proceedings of the International Conference on Digital Audio Effects*, 2010.

[47] Derry FitzGerald, Eugene Coyle, and Matt Cranitch. Using tensor factorisation models to separate drums from polyphonic music. In *Proceedings of the International Conference on Digital Audio Effects*, Como, Italy, September 2009.

[48] Jonathan Foote and Shingo Uchihashi. The beat spectrum: A new approach to rhythm anaysis. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 1088–1091, Tokyo, Japan, August 2001.

[49] Scott Foster, W. Andrew Schloss, and A. Joseph Rockmore. Toward an intelligent editor of digital audio: Signal processing methods. *Computer Music Journal*, 6(1):42–51, April-June 1982.

[50] Mark Gales and Steve Young. Robust continuous speech recognition using parallel model combination. *IEEE Transactions on Speech and Audio Processing*, 4(5):352–359, September 1996.

[51] Mark Gales and Steve Young. The application of hidden markov models in speech recognition. *Foundations and Trends in Signal Processing*, 1(3):195–304, 2007.

[52] James Glass, Jane Chang, and Michael McCandless. A probabilistic framework for feature-based speech recognition. In *Proceedings of the International Conference on Spoken Language*, volume 4, pages 2277–2280, Philadelphia, USA, October 1996.

[53] Masataka Goto and Yoichi Muraoka. A beat tracking system for acoustic signals of music. In *Proceedings of the ACM International Conference on Multimedia*, pages 365–372, San Francisco, USA, October 1994.

[54] Masataka Goto and Yoichi Muraoka. A real-time beat tracking system for audio signals. In *Proceedings of the International Computer Music Conference*, Banff, Canada, September 1995.

[55] Masataka Goto and Yoichi Muraoka. Real-time beat tracking for drumless audio signals: Chord change detection for musical decisions. *Speech Communication*, 27(4):311–335, April 1999.

[56] Fabien Gouyon, Simon Dixon, and Gerhard Widmer. Evaluating low-level features for beat classification and tracking. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages 1309–1312, Honolulu, USA, April 2007.

[57] David Grunberg. Music-information retrieval in environments containing acoustic noise. In *Proceedings of the ACM International Conference on Multimedia Multimedia*, 2014.

[58] David Grunberg, Alyssa Batula, and Youngmoo Kim. Towards the development of robot musical audition. In *Proceedings of the 2012 Music, Mind, and Innovation Workshop*, 2012.

[59] David Grunberg, Alyssa Batula, Erik Schmidt, and Youngmoo Kim. Affective gesturing with music mood recognition. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, pages 343–348, Osaka, Japan, November-December 2012.

[60] David Grunberg, Alyssa Batula, Erik Schmidt, and Youngmoo Kim. Synthetic emotions for humanoids: Perceptual effects of size and number of robot platforms. *Journal of Synthetic Emotions: Special Issue on Music, Robots, and Emotion (invited paper)*, 3(2):68–83, July-December 2012.

[61] David Grunberg, Robert Ellenberg, In Hyeuk Kim, Jun Ho Oh, Paul Oh, and Youngmoo Kim. Development of an autonomous dancing robot. *International Journal of Hybrid Information Technology*, 3(2):33–43, April 2010.

[62] David Grunberg, Robert Ellenberg, Youngmoo Kim, and Paul Oh. From RoboNova to HUBO: Platforms in robot dance. In *Proceedings of the International Conference of Advanced Humanoid Robotics Research*, August 2009.

[63] David Grunberg and Youngmoo Kim. Rapidly learning musical beats in the presence of environmental and robot ego noise. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[64] David Grunberg, Daniel Lofaro, Paul Oh, and Youngmoo Kim. Robot audition and beat identification in noisy environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2916–2921, San Francisco, USA, September 2011.

[65] Isao Hara, Futoshi Asano, Yoshihiro Kawai, Fumio Kanehiro, and Kiyoshi Yamamoto. Robust speech interface based on audio and video information fusion for humanoid hrp-2. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2404–2410, Sendai, Japan, September-October 2004.

[66] Andre Holzapfel, Matthew Davies, Jose Zapata, Joao Lobato Oliveira, and Fabien Gouyon. Selective sampling for beat tracking evaluation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(9):2539–2548, November 2012.

[67] Gokhan Ince, Kazuhiro Nakadai, Tobias Rodemann, Yuji Hasegawa, Hiroshi Tsujino, and Jun ichi Imura. Automatic speech recognition under ego-motion noise of a robot. In *Proceedings of the 30th Meeting of Special Interest Group on AI Challenges*, pages 32–38, 2009.

[68] Gokhan Ince, Kazuhiro Nakadai, Tobias Rodemann, Yuji Hasegawa, Hiroshi Tsujino, and Jun ichi Imura. Ego noise suppression of a robot using template subtracted. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 199–204, St. Louis, USA, October 2009.

[69] Gokhan Ince, Kazuhiro Nakadai, Tobias Rodemann, Yuji Hasegawa, Hiroshi Tsujino, and Jun ichi Imura. A hybrid framework for ego noise cancellation of a robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3623–3628, Anchorage, USA, May 2010.

[70] Gokhan Ince, Kazuhiro Nakadai, Tobias Rodemann, Jun ichi Imura, Keisuke Nakamura, and Hirofumi Nakajima. Assessment of single-channel ego noise estimation methods. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 106–111, San Francisco, USA, September 2011.

[71] Gokhan Ince, Kazuhiro Nakadai, Tobias Rodemann, Hiroshi Tsujino, and Jun ichi Imura. Robust ego noise suppression of a robot. In *Trends in Applied Intelligent Systems, LNAI 6096, Lecture Notes in Computer Science*, pages 62–71. Springer-Verlag, June 2010.

[72] Gokhan Ince, Kazuhiro Nakadai, Tobias Rodemann, Hiroshi Tsujino, and Jun ichi Imura. Whole body motion noise cancellation of a robot for improved automatic speech recognition. *Advanced Robotics*, 25(11):1405–1426, 2011.

[73] Akinori Ito, Takashi Kanayama, Motoyuki Suzuki, and Shozo Makino. Internal noise suppression for speech recognition by small robots. In *Proceedings of the International Speech Communication Association*, Lisbon, Portugal, September 2005.

[74] Kristoffer Jensen and Tue Andersen. Real-time beat estimation using feature extraction. In *Proceedings of the Computer Music Modeling and Retrieval Symposium*, pages 13–22, Montpellier, France, May 2003.

[75] Mao-Yuan Kao, Chang-Biau Yang, and Shyue-Horng Shiau. Tempo and beat tracking for audio signals with music genre classification. *International Journal of Intelligent Information and Database Systems*, 3(3):275–290, August 2009.

[76] Yan Ke, Derek Hoiem, and Rahul Sukthankar. Computer vision for music identification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 597–604, San Diego, USA, June 2005.

[77] Youngmoo Kim, Alyssa Batula, David Grunberg, Daniel Lofaro, Jun Ho Oh, and Paul Oh. Developing humanoids for musical interaction. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[78] Youngmoo Kim, David Grunberg, Alyssa Batula, Daniel Lofaro, Jun Ho Oh, and Paul Oh. Enabling humanoid musical interaction and performance. In *Proceedings of the International Conference on Collaboration Technologies and Systems*, 2011.

[79] Anssi Klapuri, Antti Eronen, and Jaakko Astola. Analysis of the meter of acoustic musical signals. *IEEE Transactions on Speech and Audio Processing*, 14(1):342–355, January 2006.

[80] Jonathan LeRoux, Hirokazu Kameoka, Nobutaka Ono, Alain de heveigne, and Shigeki Sagayama. Computational auditory induction by missing-data non-negative matrix factorization. In *Proceedings of the ISCA Workshop on Statistical And Perceptual Audition*, pages 1–6, Brisbane, Australia, September 2008.

[81] Shimeng Li, Yuan Zheng, and Lei Cao. New approach for modeling and testing of harmonic drive in robotic systems. In *Proceedings of the International Conference on Mechatronics and Automation*, 2014.

[82] Richard Lippmann. Speech recognition by machines and humans. *Speech Communication*, 22(1):1–15, 1997.

[83] Richard Lippmann, Ed Martin, and Douglas Paul. Multi-style training for robust isolated-word speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 705–708, Dallas, USA, April 1987.

[84] Daniel Lofaro. *Unified Algorithmic Framework for High Degree of Freedom Complex Systems and Humanoid Robots*. PhD thesis, Drexel University, May 2013.

[85] Daniel Lofaro, Sun Chunyang, and Paul Oh. Humanoid pitching at a major league baseball game: Challenges, approach, implementation and lessons learned. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, pages 423–428, 2012.

[86] Hugh Christopher Longuet-Higgins. Review lecture: The perception of music. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 205(1160):307–322, August 1979.

[87] Mrinal Mandel. *Multimedia Signals and Systems*, chapter Audio Fundamentals, pages 11–31. The Springer International Series in Engineering and Computer Science, 2003.

[88] Robert McAulay and Marilyn Malpass. Speech enhancement using a soft-decision noise speech enhancement using a soft-decision noise suppression filter.

IEEE Transactions on Acoustics, Speech, and Signal Processing, 28(2):137–145, April 1980.

[89] Martin McKinney. Extracting the perceptual tempo from music. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 146–149, Barcelona, Spain, October 2004.

[90] Martin McKinney and Dirk Moelants. Ambiguity in tempo perception: What draws listeners to different metrical levels? *Music Perception: An Interdisciplinary Journal*, 24(2):155–166, December 2006.

[91] Paul Mermelstein. Distance measures for speech recognition – psychological and instrumental. In *Proceedings of the Joint Workshop on Pattern Recognition and Artificial Intelligence*, pages 91–103, Hyannis, Massachusetts, USA, June 1976.

[92] Ben Miller and Xu Shao. Speech reconstruction from mel-frequency cepstral coefficients using a source-filter model. In *Proceedings of the International Conference on Spoken Language Processing*, pages 2421–2424, Denver, USA, September 2002.

[93] Dirk Moelants. Dance music, movement and tempo preference. In *Proceedings of the 5th Triennial ESCOM Conference*, Hannover, Germany, September 2003.

[94] Abdel-Rahman Mohamed, George Dahl, and Geoffrey Hinton. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):14–22, January 2012.

[95] Pedro Moreno, Bhiksha Raj, and Richard Stern. A vector taylor series approach for environment-independent speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 733–736, Atlanta, USA, May 1996.

[96] Kazumasa Murata, Kazuhiro Nakadai, Kazuyoshi Yoshii, Ryu Takeda, Toyotaka Torii, Hiroshi Okuno, Yuji Hasegawa, and Hiroshi Tsujino. A robot singer with music recognition based on real-time beat tracking. In *Proceedings of the International Society for Music Information Retrieval Conference*, Philadelphia, USA, September 2008.

[97] Hiroaki Nanjo and Tatsuya Kawahara. Language model and speaking rate adaptation for spontaneous presentation speech recognition. *IEEE Transactions on Speech and Audio Processing*, 12(4):391–400, July 2004.

[98] Yoshitaka Nishimura, Mikio Nakano, Kazuhiro Nakadai, Hiroshi Tsujino, and Mitsuru Ishizuka. Speech recognition for a robot under its motor noises by selective application of missing feature theory and mllr. In *Proceedings of the ISCA Tutorial and Research Workshop on Statistical And Perceptual Audition*, pages 53–58, Pittsburgh, USA, September 2006.

[99] Joao Lobato Oliveira, Matthew Davies, Fabien Gouyon, and Luis Paulo Reis. Beat tracking for multiple applications: A multi-agent system architecture with state recovery. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(10):2696–2706, December 2012.

[100] Joao Lobato Oliveira, Gokhan Ince, Keisuke Nakamura, and Kazuhiro Nakadai. Online audio beat tracking for a dancing robot in the presence of ego-motion noise in a real environment. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 403–408, St. Paul, Minnesota, USA, May 2012.

[101] Nobutaka Ono, Kenichi Miyamoto, Jonathan LeRoux, Hirokazu Kameoka, and Shigeki Sagayama. Separation of a monaural audio signal into harmonic/percussive components by complementary diffusion on spectrogram. In *Proceedings of the EUSIPCO 2008 European Signal Processing Conference*, August 2008.

[102] Douglas O'Shaughnessy. Linear predictive coding. *IEEE Potentials*, 7(1):29–32, August 1988.

[103] Douglas O'Shaughnessy. Improving analysis techniques for automatic speech recognition. In *Proceedings of the Midwest Symposium on Circuits and Systems*, volume 3, pages 65–68, Tulsa, USA, August 2002.

[104] Ill-Woo Park, Jung-Yup Kim, Jungho Lee, and Jun Ho Oh. Mechanical design of humanoid robot platform khr-3 (kaist humanoid robot 3: Hubo). In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, pages 321–326, Tsukuba, Japan, December 2005.

[105] Lucas Parra and Christopher Alvino. Geometric source separation: Merging convolutive source separation with geometric beamforming. *IEEE Transactions on Speech and Audio Processing*, 10(6):352–362, September 2002.

[106] Geoffroy Peeters and Helene Papadopoulos. Simultaneous beat and downbeat-tracking using a probabilistic framework: Theory and large-scale evaluation. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(6):1–17, August 2011.

[107] Stephen Douglas Peters, Peter Stubley, and Jean-Marc Valin. On the limits of speech recognition in noise. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 365–368, Phoenix, USA, March 1999.

[108] Matthew Prockup, David Grunberg, Alex Hrybyk, and Youngmoo Kim. Orchestral performance companion: Using real-time audio to score alignment. *IEEE Multimedia*, 20(2):52–60, April-June 2013.

[109] David Rosenthal. Emulation of human rhythm perception. *Computer Music Journal*, 16(1):64–76, April-June 1992.

[110] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, February 1978.

[111] Eric D. Scheirer. Tempo and beat analysis of acoustic musical signals. *Journal of the Acoustic Society of America*, 103(1):588–601, January 1998.

[112] Jarno Seppanen, Antti Eronen, and Jarmo Hiipakka. Joint beat and tatum tracking from music signals. In *Proceedings of the International Society for Music Information Retrieval Conference*, Victoria, Canada, October 2006.

[113] Klaus Seyerlehner, Gerhard Widmer, and Dominik Schnitzer. From rhythm patterns to perceived tempo. In *Proceedings of the International Society for Music Information Retrieval Conference*, Vienna, Austria, September 2007.

[114] Fei Sha and Lawrence Saul. Large margin hidden markov models for automatic speech recognition. In *Advances in Neural Information Processing Systems*, Vancouver, Canada, December 2007.

[115] Madhusudana Shashanka, Bhiksha Raj, and Paris Smaragdis. Probabilistic latent variable models as non-negative factorizations. *Computational Intelligent and Neuroscience Journal*, 2008.

[116] Paris Smaragdis and Bhiksha Raj. Shift-invariant probabilistic latent component analysis. Technical Report TR2007-009, Mitsubishi Electric Research Laboratories, 2007.

[117] Paris Smaragdis, Bhiksha Raj, and Madhusudana Shashanka. Missing data imputation for spectral audio signals. In *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6, Grenoble, France, September 2009.

[118] Paris Smaragdis, Bhiksha Raj, and Madhusudana Shashanka. Missing data imputation for time-frequency representations of audio signals. *Journal of Signal Processing Systems*, 65(3):361–370, December 2011.

[119] Jorge Solis, Klaus Petersen, Takeshi Ninomiya, Masaki Takeuchi, and Atsuo Takanishi. Development of anthropomorphic musical performance robots: From understanding the nature of music performance to its application to entertainment robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2309–2314, St. Louis, USA, October 2009.

[120] Jason Sroka and Louis Braida. Human and machine consonant recognition. *Speech Communication*, 45(4):401–423, April 2005.

[121] Charles Stein. Estimation of the mean of a multivariate normal distribution. *The Annals of Statistics*, 9(6):1135–1151, November 1981.

[122] Bob L. Sturm. Classification accuracy is not enough: On the evaluation of music genre recognition systems. *Journal of Intelligent Information Systems*, July 2013.

[123] Michael Trompf. Neural network development for noise reduction in robust speech recognition. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, volume 4, pages 722–727, Baltimore, USA, June 1992.

[124] Vivek Tyagi and Christian Wellekens. On desensitizing the mel-cepstrum to spurious spectral components for robust speech recognition. In *Proceedings of the International Conference on Acoustic, Speech, and Signal Processing*, Philadelphia, PA, March 2005.

[125] Jean-Marc Valin, Jean Rouat, and Francois Michaud. Enhanced robot audition based on microphone array source separation with post-filter. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2123–2128, Sendai, Japan, September-October 2004.

[126] Barry Van Veen and Kevin Buckley. Beamforming: A versatile approach to spatile filtering. *ASSP Magazine*, 5(2):4–24, 1988.

[127] Berry Vercoe and Miller Puckette. Synthetic rehearsal: Training the synthetic performer. In *Proceedings of the International Computer Music Conference*, pages 275–278, Vancouver, Canada, August 1985.

[128] Oriol Vinyals and Suman Ravuri. Comparing multilayer perceptron to deep belief network tandem features for robust ASR. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 4596–4599, Prague, Czech Republic, May 2011.

[129] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, Kauai, Hawaii, USA, December 2001.

[130] Avery Wang. The shazam music recognition service. *Communications of the ACM*, 49(8):44–48, August 2006.

[131] Gil Weinberg and Scott Driscoll. Toward robotic musicianship. *Computer Music Journal*, 30(4):28–45, December-February 2006.

[132] Ryuichi Yamamoto, Shinji Sako, and Tadashi Kitarmura. Real-time audio to score alignment using segmental conditional random fields and linear dynamical system. In *Proceedings of the International Society for Music Information Retrieval Conference*, 2012.

[133] Kazuyoshi Yoshii, Masataka Goto, and Hiroshi Okuno. Drum sound recognition for polyphonic audio signals by adaptation and matching of spectrogram templates with harmonic structure suppression. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(1):333–345, 2007.

[134] Kazuyoshi Yoshii, Kazuhiro Nakadai, Toyotaka Torii, Yuji Hasegawa, Hiroshi Tsujino, Kazunori Komatani, Tetsuya Ogata, and Hiroshi Okuno. A biped robot that keeps steps in time with musical beats while listening to music with its own ears. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1743–1750, San Diego, USA, September-October 2007.

[135] Guoshen Yu and Stephane Mallat. Audio denoising by time-frequency block thresholding. *IEEE Transactions on Signal Processing*, 56(5):1830–1839, May 2008.

Appendices

Appendix A. Technical Specifications of Hubo

This thesis made use of the Hubo+ humanoid robot, an advanced humanoid developed by the Korean Advanced Institute of Science and Technology (KAIST) [84]. Technical specifications of Hubo are listed in Table A.1 [84]. Additionally, a skeletal representation of Hubo detailing its dimensions is displayed in Figure A.1.

Table A.1: Hubo+ Technical Specifications.

Height	130cm
Mass	37kg
Degrees of Freedom (DOF)	38
Joint Control Type	High-Gain PID Position
Computer	1.6 GHz Atom 1 GB DDR3-RAM
Balance Sensors	One 4-Axis Inertia Measurement Unit (IMU) Four 3-Axis Force-Torque Sensors Two 2-Axis Tilt Sensors
Vision Sensors	One Stereo, Monocular, RGBD Bumblebee Camera
Auditory Sensors	Two Shure SM93 Microphones

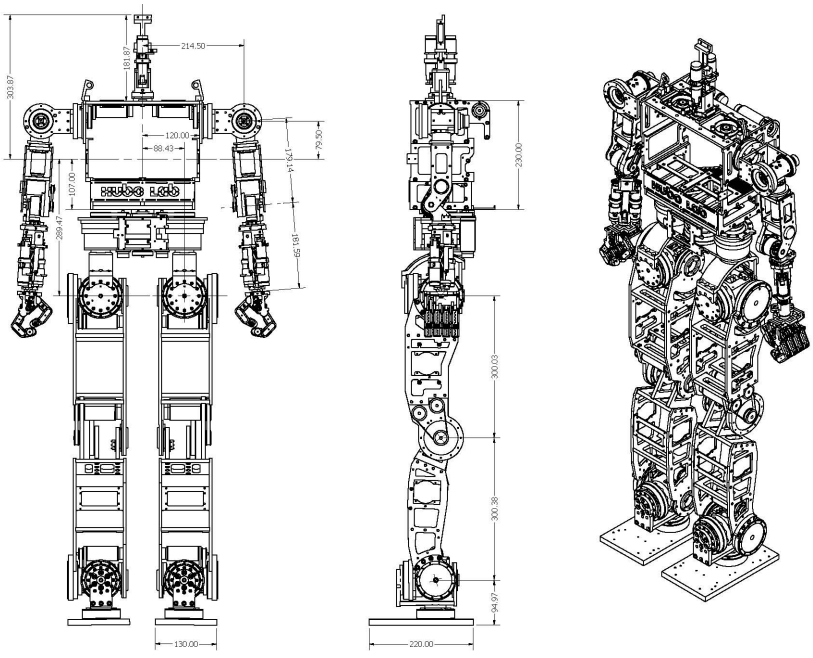

Figure A.1: Skeletal representation of Hubo. All units are in millimeters.

Appendix B. Description of Audio Collections

Below are descriptions of the Dance and the General audio collections, as well as the lists used to identify candidates for the General Dataset.

Table B.1: Description of the Dance Dataset. Length is in the form of minutes: seconds. Tempo is in beats per minute and averaged over the entire piece.

Artist	Title	Length	Tempo
Ace of Base	Waiting for Magic	3:48	130
Billy Joel	A Matter of Trust	4:06	111
Billy Ray Cyrus	Achy Breaky Heart	3:17	122
Blood or Whiskey	King of the Fairies	3:15	139
Bobby Fuller Four	I Fought the Law	2:15	152
Daft Punk	Veridis Quo	3:30	110
Depeche Mode	Master and Servant	3:45	128
Dschinghis Khan	Moskau	4:28	134
Eurythmics	Here Comes the Rain Again	5:01	127
Eurythmics	The King and Queen	4:27	135
Eurythmics	Sweet Dreams	3:30	126
Falco	Der Kommissar	5:36	123
Genesis	That's All	4:20	090
The Human League	Don't You Want Me	3:55	117
Irena Cara	Fame	5:09	132
Jamiroquai	Canned Heat	5:20	125
Johnny Cash	Ghost Riders in the Sky	3:42	108

Continued on next page

Table B.1 – *Continued from previous page*

Artist	Title	Length	Tempo
London Philharmonic	The Arbiter	2:19	133
Men Without Hats	The Safety Dance	2:42	104
Technotronic	Pump up the Jam	5:20	125

Table B.2: Lists used to obtain candidates for the General Dataset

Genre	Title of List	Affiliation
Classical	50 Greatest Pieces of Classical Music	London Philharmonic Orchestra
Country	100 Greatest Songs of Country Music	Country Music Television
Dance	50 Greatest Dance Tracks of All Time	Mixmag
Folk	100 Most Essential Folk Songs	Folk Alley
Hip-Hop	100 Greatest Hip Hop Songs Ever	VH1
Jazz	100 Quintessential Jazz Songs	National Public Radio
Metal	Top 50 Metal Songs of All Time	Gibson's
Pop	Top 100 Pop Songs 1992-2012	Billboard
Rock	Top 1000 Classic Rock Songs of All Time	KZOK-FM
Soul	100 Greatest Soul/R&B Songs of All Time	Soul Bounce

Table B.3: Description of the General Dataset. All excerpts are one minute long. Tempo is in beats per minute and is averaged over the entire piece. Genre is determined based on the lists in Table B.2.

Genre	Artist	Title	Tempo
Classical	Ludwig van Beethoven	Symphony No. 5, Mvt. No. 1	083
	Richard Wagner	Ride of the Valkyries	102
	Gustav Holst	The Planets: Jupiter	127
	Wolfgang Mozart	Eine Kleine Nachtmusik, Mvt. No. 1	147
	Carl Orff	Carmina Burana: O Fortuna	107
	Edward Elgar	Pomp and Circumstance No. 1	120
	Georges Bizet	Carmen Suite No. 2: Habanera	122
	Antonio Vivaldi	The Four Seasons: Spring	164
	Edvard Grieg	Peer Gynt Suite No. 1	113
	Johannes Brahms	Hungarian Dance No. 5	120
Country/ Western	Tammy Wynette	Stand By Your Man	106
	George Jones	He Stopped Loving Her Today	072
	Patsy Cline	Crazy	072
	Johnny Cash	Ring of Fire	104
	Hank Williams	Your Cheatin' Heart	129
	Garth Brooks	Friends in Low Places	109
	Patsy Cline	I Fall to Pieces	111
	Glen Campbell	Galveston	120
	Charlie Rich	Behind Closed Doors	083

Continued on next page

Table B.3 – *Continued from previous page*

Genre	Artist	Title	Tempo
	Waylon Jennings & Willie Nelson	Mammas Don't Let Your Babies Grow Up To Be Cowboys	062
Dance	Daft Punk	One More Time	122
	Tiesto	Adagio for Strings	140
	The Prodigy	Smack My ***** Up	136
	Underworld	Born Slippy .NUXX	086
	Faithless	Insomnia	128
	Stardust	Music Sounds Better With You	125
	Plastikman	Spastik	126
	Paul Van Dyke	For An Angel	140
	Sasha	XPander	137
	Fatboy Slim	Right Here, Right Now	125
Folk	Woody Guthrie	This Land Is Your Land	196
	Bob Dylan	Blowin' in the Wind	168
	Steve Goodman	City of New Orleans	092
	Pete Seeger	If I Had A Hammer	128
	The Kingston Trio	Where Have All The Flowers Gone?	149
	Gordon Lightfoot	Early Morning Rain	104
	Leonard Cohen	Suzanne	132
	Pete Seeger	We Shall Overcome	084
	Ian & Sylvia	Four Strong Winds	105
	Tom Paxton	The Last Thing On My Mind	166

Continued on next page

Table B.3 – *Continued from previous page*

Genre	Artist	Title	Tempo
Hip-Hop	Public Enemy	Fight the Power	105
	The Sugarhill Gang	Rapper's Delight	111
	Dr. Dre & Snoop Dogg	Nuthin' but a 'G' Thang	190
	Aerosmith & Run-D.M.C.	Walk This Way	106
	Grandmaster Flash & the Furious Five	The Message	101
	***	Straight Outta Compton	104
	The Notorious B.I.G.	Juicy	097
	Snoop Dogg	Gin and Juice	095
	Salt-n-Pepa	Push It	129
	Kurtis Blow	The Breaks	115
Jazz	The Dave Brubeck Quartet	Take Five	106
	Miles Davis	So What	137
	Duke Ellington	Take the 'A' Train	160
	Thelonious Monk	Round Midnight	062
	John Coltrane	My Favorite Things	167
	John Coltrane	Acknowledgment	113
	Miles Davis	All Blues	137
	Weather Report	Birdland	086
	Astrud Gilberto with Stan Getz & Joao	The Girl From Ipanema	125

Continued on next page

Table B.3 – *Continued from previous page*

Genre	Artist	Title	Tempo
	Gilberto		
	Benny Goodman	Sing, Sing, Sing	109
Metal	Metallica	Master of Puppets	210
	Motorhead	Ace of Spades	279
	Ozzy Osbourne	Crazy Train	251
	Black Sabbath	Iron Man	070
	Iron Maiden	The Number of the Beast	192
	Black Sabbath	War Pigs	185
	Black Sabbath	Paranoid	164
	Metallica	One	105
	Iron Maiden	Hallowed Be Thy Name	107
	Judas Priest	Breaking the Law	164
Pop	Goo Goo Dolls	Iris	052
	Real McCoy	Another Night	127
	Timbaland & OneRepublic	Apologize	060
	Lifehouse	Hanging by a Moment	125
	Santana w/ Rob Thomas	Smooth	117
	Savage Garden	Truly Madly Deeply	085
	Paramore	crushcrushcrush	137
	3 Doors Down	Here Without You	145
	Goo Goo Dolls	Slide	112
	Kelly Clarkson	Since U Been Gone	131

Continued on next page

Table B.3 – *Continued from previous page*

Genre	Artist	Title	Tempo
Rock	The Beatles	Sgt. Pepper's Lonely Hearts Club Band	094
	Led Zeppelin	Stairway to Heaven	088
	Pink Floyd	Another Brick in the Wall Part 2	106
	The Who	Won't Get Fooled Again	139
	The Rolling Stones	I Can't Get No Satisfaction	138
	Lynyrd Skynyrd	Free Bird	059
	The Eagles	Hotel California	074
	The Beatles	Come Together	166
	Derek and the Dominos	Layla	114
	Jimi Hendrix	Purple Haze	108
	Donny Hathaway	A Song For You	050
	Prince	Adore	068
	Michael Jackson	Rock With You	115
	The Gap Band	Yearning for Your Love	088
	Maxwell	Ascension	097
	Mary J. Bilge	Real Love	096
	Faith Evans	Soon as I Get Home	061
	Boyz II Men	I'll Make Love To You	048
	Meshell Ndegeocello	Outside Your Door	138
	Luther Vandross	Never Too Much	110

Vita

David Kurt Grunberg is a student in the Electrical and Computer Engineering Department of Drexel University. His degrees include:

- Ph. D., Drexel University, December 2014
- M.S., Drexel University, December 2011
- B.S., Drexel University, June 2010

He has received the following honors:

- National Science Foundation Graduate Research Fellowship
- Member of Eta Kappa Nu and Tau Beta Pi (engineering honor societies)
- Graduated Magna Cum Laude
- Drexel Presidential Scholarship

His publications include:

- M. Prockup, D. Grunberg, A. Hrybyk, and Y. Kim, "Orchestral Performance Companion: Using Real-Time Audio to Score Alignment," *IEEE MultiMedia*, vol. 20, no. 2, pp. 52-60, 2013.
- D. Grunberg, A. Batula, E. Schmidt, and Y. Kim, "Synthetic Emotions for Humanoids: Perceptual Effects of Size and Number of Robot Platforms," *Journal of Synthetic Emotions: Special Issue on Music, Robots, and Emotion*, vol. 3, no. 2, pp. 68-83, 2012. (*Invited Paper.*)