ABSTRACT

Title of Thesis: AVISARME: Audio-Visual Synchronization Algorithm for a Robotic Musician Ensemble

David R Berman, Master of Science, 2012

Thesis directed by: Professor Nikhil Chopra
Department of Mechanical Engineering

This thesis presents a beat detection algorithm which combines both audio and visual inputs to synchronize a robotic musician to its human counterpart. Although there has been considerable work done to create sophisticated methods for audio beat detection, the visual aspect of musicianship has been largely ignored. With advancements in image processing techniques, as well as both computer and imaging technologies, it has recently become feasible to integrate visual inputs into beat detection algorithms. Additionally, the proposed method for audio tempo detection also attempts to solve many issues that are present in current algorithms. Current audio-only algorithms have imperfections, whether they are inaccurate, too computationally expensive, or suffer from terrible resolution. Through further experimental testing on both a popular music database and simulated music signals, the proposed algorithm performed statistically better in both accuracy and robustness than the baseline approaches. Furthermore, the proposed approach is extremely efficient, taking only 45ms to compute on a 2.5s signal, and maintains an extremely high temporal resolution of 0.125 BPM. The visual integration also relies on Full

Scene Tracking, allowing it to be utilized for live beat detection for practically all musicians and instruments. Numerous optimization techniques have been implemented, such as pyramidal optimization (PO) and clustering techniques which are presented in this thesis. A Temporal Difference Learning approach to sensor fusion and beat synchronization is also proposed and tested thoroughly. This TD learning algorithm implements a novel policy switching criterion which provides a stable, yet quickly reacting estimation of tempo. The proposed algorithm has been implemented and tested on a robotic drummer to verify the validity of the approach. The results from testing are documented in great detail and compared with previously proposed approaches.

AVISARME: Audio-Visual Synchronization Algorithm for a Robotic Musician Ensemble

by

David R Berman

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2012

Advisory Committee:
Professor Nikhil Chopra, Chair/Advisor
Professor Balakumar Balachandran
Professor Miao Yu

Dedication

To my loving parents, my amazing girlfriend, my sister
my grandparents, and last, but certainly not least,
my advisor, Dr. Nikhil Chopra. If it wasn't for your
love, faith, and support, I would not be here today.

Table of Contents

List of Figures

List of Abbreviations

AVISARME	Audio-Visual Synchronization Algorithm for a Robotic Musician Ensemble
BPM	Beats Per Minute
CFC	Comb Filter Convolution
FFT	Fast Fourier Transform
FPS	Frames Per Second
HMM	Hidden Markov Model
MIDI	Musical Instrument Digital Interface
PO	Pyramidal Optimization
SNR	Signal to Noise Ratio
STFT	Short Time Fourier Transform
TD	Temporal Difference

Chapter 1

Introduction

1.1 Motivation

Music is one of the few mediums which can transcend language barriers while still expressing the original thoughts and emotions of the author. When listening to a piece, most people can pick up on the intricacies of the musicians and immediately comprehend the entirety of the song. But what seems simple to most humans is extremely difficult for computers. Humans, almost instinctually, start tapping their foot or nodding their head to the tempo of their favorite song. What we hear when we listen to music is a series of discrete drum beats and guitar strokes. Computers, on the other hand, aren't so lucky. What the computer reads is one continuous signal, a stream of amplitude modulating data. It is the job of the programmer to convert this stream of data into a form from which a robot will be able to construe the tempo, genre, etc. This is what is referred to as robotic musicianship. The field of robotic musicianship has been widely researched, from Kozima and Michalowski's work at the Massachusetts Institute of Technology and Carnegie Mellon University [1], to Weinberg's work at Georgia Tech University [2], to Murata's and Mizumoto's work at Kyoto University [3, 4]. This issue of continuous signal to discrete beat conversion is the main hardship in developing a robot which can interact musically with a human. Although individual frequency bands, such

as low frequency bass kicks or mid-range guitar strokes, can be extracted from the music, this process consumes a large amount of computational cost, temporal resolution (resolution in the time domain) is lost, and does not always produce the most reliable representation of the tempo of the music.

On the other side, current beat synchronization algorithms have mostly focused on the audio aspect of musicianship, ignoring the fluid motions of the musician. In doing this, the algorithms are using only half the information available when attempting to synchronize with the beat. Musicians instinctually move with the tempo of the music while playing, and they are extremely robust to complicated beat structures. By incorporating visual information into the detection process and combining the signals in a useful manner, it is possible to obtain information greater than the sum of its parts.

Introduce AVISARME, or Audio Visual Synchronization Algorithm for a Robotic Musician Ensemble. Roughly translating to "advise me" or "lead me" in Spanish, AVISARME is our proposal of a simplified method of audio beat detection combined with a first-of-its-kind visual beat detection algorithm. In the proposed method, a computationally efficient approach, referred to as the Overall Amplitude Approach, is provided. The Overall Amplitude Approach is defined as a method for audio beat detection where the solver ignores the individual frequency components (pitches) of the signal, focusing only on the normalized amplitude of the music signal being measured. In other words, the entire signal is located within a single frequency bin. Our method, based on this approach, utilizes a unique convolution algorithm and Pyramidal Optimization scheme to efficiently convert the time-domain signal into

the frequency domain. This allows the robot to quickly and accurately determine the Beats Per Minute (BPM) of the input signal with an extremely high resolution. AVISARME is also one of the early attempts at detecting beats visually. Using the Microsoft Kinect along with optical flow, the motions of the entire human body and instrument are tracked in order to extract information on the music he or she is playing. Due to the efficient depth detection of the Kinect, this tracking is able to be performed at 30FPS (Frames Per Second) allowing for a sufficiently high resolution when converted to the temporal domain. Due to our Full Scene Tracking model, this method is able to be applied to any instrument or musician, a current limitation of the work of Itohara et al. [5], and Lim et al. [6].

As the name implies, AVISARME was designed to be used for robotic musician ensembles. While it may be possible to form an entire orchestra using only robots and a human conductor, the more common application would be to use the robot as a stand-in for an otherwise absent musician. Musical groups meet often to rehearse, and it is common the case that not all members can attend. Having a robot that can track and play any instrument with high precision is extremely valuable in these scenarios. Either in a group setting, or practicing alone, artificial accompaniment which can adapt to one's playing can be crucial to the learning process.

The applications of this type of algorithm is not bound to robotic musicians. When designing concert lighting, Lighting Designers can spend weeks, even months, working with a band to design a light show for their setlist. Designers painstakingly go through recorded versions of the music, labelling each beat and timing cues for different routines. This is all done in advance of the first show, and an operator

is needed at each show to oversee that everything goes as planned. If the band deviates from the setlist, decides to improvise a solo, or simply plays at a slightly different speed, this wreaks havoc on the operator. Incorporating an accurate beat detection algorithm into the design of the controller could eliminate these issues and save millions of dollars in the process.

1.2 Related Work

The field of Robotic Musicianship is fairly new, only dating back to the mid 1980's with Povel and Essens' [7] proposal of a possible method of finding a piece's internal clock. However, it wasn't until the early 1990's that working models began to take shape. In 1993, Goto, et al., [8] (Waseda University, Japan) created a prototype which filters the original signal into multiple frequency bands using Fourier transforms in order to detect the tempo of the bass drum. The approach had two main shortcomings; First, if the music being processed did not contain a bass drum, such as the case of many classical pieces and individual performances, the approach was inapplicable. Secondly, due to stability concerns and lack of computational power at the time, all of the detection was done in post-processing.

In 1996, Scheirer, et al. [9] (MIT Media Laboratory) took this a step further by performing beat detection on each of the frequency bands, and selecting the BPM with the highest energy. Instead of using Fourier transforms, Scheirer used Comb Filter Convolution, or the process of correlating combs of different tempos with the original music signal in order to convert the time domain signal into the

frequency domain. While this was able to be done in real time, the process was quite expensive, taking about 1 second for a 2.5s signal.

In 2004, there was a call for algorithms by ISMIR, the International Society for Music Information Retrieval. The most accurate algorithm, developed by Klapuri [10, 11], was a modified version of Scheirer's approach. In this approach, Klapuri utilized more sub-bands along with a statistical model of beat structure. However, computational complexity and time requirements increased even further.

Since then, there have been a number of other approaches to solve the tempo-matching problem. Georgia Tech has developed two robots, Haile [2] and Shimon [12], which are generally accepted as the first improvisational robotic musicians. Combining beat detection and a deep base of music theory, the robots, Haile (a drummer) and Shimon (a marimba player), can listen to a live piece and simultaneously improvise, responding with an adaptation of the music. The approach to beat detection is radically different than the aforementioned proposals. Instead, it adapts a method proposed by Desain and Honing [13], which applies previous knowledge of instrumental music. Knowing that the preferred distance between beats for this particular type of music is 0.6 seconds, it searches for successive peaks around 0.6s apart, and according to the time between peaks, it finds the corresponding BPM. Although this may not portray the correct tempo of the music, it does produce a stable beat structure which needed for fluent improvisation.

In 2006, Kozima [1], in conjunction with Carnegie Mellon University, introduced KeepOn, a small, yellow robot designed to interact with autistic children. Most studies surrounding the robot are based on the therapeutic assistance it pro-

vides, however, its approach to beat detection is admirable in its own right. Based on simplicity rather than prior knowledge of sophisticated beat structures, the algorithm is computationally negligible. Although it is not referred to such in its literature, it was the first to introduce the Overall Amplitude Approach. KeepOn's beat detection algorithm is unlike the proposed method, however it is similar in that it does not filter the original signal into separate frequency bands. Instead, the algorithm employed by KeepOn looks only at the normalized amplitude of the entire signal, and waits until it hears three equally spaced peaks in power. Studies [1] show that KeepOn is rarely moving synchronously with the music, but it does allow the robot to perform real-time beat detection on a micro-controller. This was the first publication to propose that beat detection could be successfully performed by looking at the overall amplitude of the music.

Figure 1.1: (from left to right) Keepon, Shimon, and Haile

The work of Murata [14], and later Mizumoto [3], is the modern implementation of the original proposals of Goto, and Povel and Essens. It exploits the use of Short Time Fourier Transforms to form a spectrogram of the music in the hope that this will best represent the intricacies of the piece in a graphically simple way.

This process is often used for song/instrument identification as it allows the user to extract spectral (frequency) information, which would otherwise be unavailable. The algorithm then performs Spectral-Temporal Pattern Matching, or STPM, which searches for recurring patterns in the time-frequency domain. The analysis can still be done in real-time, however, it is computationally expensive. Most importantly, as the STFT is only calculated every ∼11.6ms, the resolution in the time domain is drastically decreased.

Lim [6] has expanded on their initial work by including visual cues of a human flutist. Using Hugh lines to track the flute, it requires the flutist to make gestures with the start of each measure in order to signify a beat. Although it is capable of producing extremely accurate results, it is extremely limited in its application.

Recently, Itohara [5] expanded on Mizumoto's work to incorporate visual beat detection without using cues. Using optical flow, the robot is able to track the strumming hand of a human guitarist. Although this is the first publication to provide a working application of visual beat detection without cues, it is still limited to only one instrument, and the resolution issue remains.

Although these sophisticated methods, STPM and the like, are able to obtain a large amount of information about the music, such as individual frequency components of the prominent instruments, it has the major drawback of temporal resolution. While STFTs limit the time resolution to ∼11.6ms, this is much larger than that of the comb filter convolution method proposed by Scheirer which has a resolution of 0.022ms if sampled at the 44.1kHz norm. This may seem negligible, but the 11.6ms resolution has a huge impact when converting into the tempo domain,

limiting the detection resolution to 1.45BPM at 60BPM, and even up to 20.4BPM at 240BPM.

In a recent publication, it was argued that comb filter convolution, the convolution of impulse trains with a time domain signal to convert it into the frequency domain, is non-ideal for beat detection due to the large amount of necessary calculations and lack of tempo resolution [15]. The author states that the disadvantage of CFC is that the possible tempo values (comb frequencies) needs to be assigned in advance, and that the amount of combs should be minimized to decrease computational cost. This decrease in number of combs, in turn, limits the maximum beat resolution significantly. This proposal does not only illustrate a streamlined approach of the comb-filter convolution sufficiently to prove the viability of the technique for live beat detection, but a pyramidal optimization (PO) scheme is also presented in order to break the dependence of resolution on the number of convolving combs.

1.3 Goals

The main goal of the research was to create a beat synchronization method which combined both audio and visual inputs in such a way that it could be applied in real time to any instrument. Furthermore, we intended to maintain the highest temporal resolution possible and minimize tracking errors. Temporal resolution can be used to define the resolution in both the time time and tempo domains. A temporal resolution measured in ms represents the resolution in the time domain

and a resolution measured in BPM represents the resolution in the tempo domain. The goal of the proposed algorithm is to maintain a high resolution in both domains. For both the audio beat detection and the visual beat detection, there was a focus on the following three areas:

Robustness - The robot must be able to use the same method for any instrument and for any musician. Therefore, the robot must not utilize instrument-specific shapes or watch for motions required to play individual instruments. Furthermore, the musician need not perform special gestures and/or wear special markers in order for the method to synchronize with the beat. The beat detection must also be able to be synchronize with all tempos ranging from 60BPM to 240BPM; the current STPM method only searches between 61BPM and 120BPM to avoid the issue of half/double beat detection.

Speed - The program must be able to do the calculations quickly so that the human does not notice a hesitation in motion. As the STPM algorithms takes $\sim$110ms to calculate, the goal was to create an algorithm which can be performed in half the time, $\sim$55ms.

Accuracy - The proposed method must be able to detect the correct tempo of the music, as well as the correct beat onset timing. The tempo of the music refers to the speed of the music in Beats Per Minute, or BPM. Beat onset time refers to the instant right before a note is played; the onset error represents the time difference between when a note is expected to occur and the instant the beat actually occurs. The algorithm must be precise enough so that the human musician perceives that the robot's tempo and beat onset are identical to his/her tempo and offset. The goal

is to reduce the tracking error to less than ± 20ms. Since humans cannot distinguish auditory events which occur less than 20ms apart, the ideal 20ms offset error was chosen. The current most accurate algorithm [5] has an average offset error of ± 150ms.

The initial inspiration for our research can be attributed to KeepOn. Although the beat detection algorithm often did not calculate an accurate tempo, it provided inspiration that beat detection could be performed in an extremely efficient way on the original signal. Instead of carrying out intense and perhaps unnecessary computations, the intention was to combine the simplicity and efficiency of Kozima's algorithm with the accuracy and robustness of Itohara's method, all while maintaining the resolution of Scheirer's proposal.

There was also motivation to become the first group to publish a method on marker-less/gesture-less tracking for visual beat detection. At the time of conception, the most recent publication of visual beat detection was by Lim [6], the first to propose a method for beat detection of a flute using gestures to indicate a beat. KeepOn, published earlier, included a visual beat detection algorithm as well, but details on the method have not been included in a related publication. We were inspired by the possibility of being the first to merge audio and visual signals for seamless beat detection.

It is good to note that the audio beat detection method proposed in this thesis is only useful for tempo synchronization, and is not ideal for score following, beat structure induction, or other types of information retrieval that might be associated with music signals. While the proposed algorithm may be used in conjunction

with other approaches, such as STPM and/or song identification, to enhance the performance of these efforts, the proposed approach, alone, is not sufficient. For these assignments, any binned approach, especially a spectrogram-based approach, would prove more useful.

1.4 Problem Formulation

Since Scheirer's [9] proposal in 1996, there has been a movement to using more and more frequency bins for audio beat synchronization. Along with this, came the introduction of STPM, or Spectral Temporal Pattern Matching, the latest development in the tempo detection and synchronizing field. The problem with using many bins is two-fold: If the detection is done using Comb Filter Convolution, the calculation time becomes multiplied by the number of additional bins. On the other hand, if STPM is used to decrease the calculation time, resolution in both the tempo and time domains are decreased drastically. Therefore, the first problem we set out to solve was to develop a new approach which combines the speed of the STPM approach while maintaining the resolution of the CFC approach.

The second problem was to develop a visual beat detection algorithm which could be used independently of audio beat detection, and which is not based on gestures or cues. While there are quite a few algorithms which incorporate visual input into the beat detection process, they are all either instrument specific or are only used to tell the robot to start, stop, follow, etc. Of the instrument specific algorithms, they either require the musician to perform unnatural gestures while

performing or do not have sufficient resolution to be used as the sole detection method. Therefore, the second problem is to develop a method for visual beat detection which is neither instrument specific nor cue based, and contains sufficient resolution to be implemented independently of an audio-based tempo detection code.

Once these two problems are solved, the last issue is fusing the readings in a useful manner. While Hidden Markov Models and Particle Filters have been used successfully for smoothing and fusion, they are quite expensive computationally. The goal was to develop a fusion algorithm which can perform at the same level of performance, but be calculated in a fraction of the time.

1.5 Contribution

In pursuit of solving these problems that current algorithms face, we have developed numerous novel alterations to previously published approaches. In 1996, Scheirer [9] first proposed the Comb Filter Convolution approach to audio beat detection. In his proposal, the music signal was first split into numerous bands and CFC was performed on each band independently. The proposed Overall Amplitude Approach is one that performs CFC on the original signal as opposed to multiple bins. Utilizing a modified CFC algorithm along with some pre-processing techniques to increase accuracy, multiple bins are no longer needed, decreasing the necessary time to calculate the tempo of the music. Bouguet's 2D Pyramidal Optimization algorithm [16] has also been adapted for the 1D audio signal case, allowing for a resolution gain of 8x and a speed gain of 25%.

We have also successfully implemented a general, gesture-less visual beat detection algorithm that can be run independently as its own form of tempo synchronization. By utilizing a background cancellation scheme, Optical Flow is able to be performed at high speed, allowing for high resolution synchronization. Although Optical Flow is also used in Itohara's proposal to track the motion of a guitarist's hand, the approach is limited in application and can only be performed at a much slower pace (19FPS). The background cancellation proposed in this paper allows for a resolution of 30FPS, translating to a temporal resolution of 3.5 BPM. The proposed approach is also general in nature, and can be used for theoretically any instrument.

We have also implemented the first exemplification of Temporal Difference Learning for Beat Detection. Since learning/smoothing is generally performed after readings from the sensors are made, it is important that this be done quickly. The proposed approach takes only an additional 5ms to compute. We have also implemented a switching policy based on Markov Modeling which allows for an extremely smooth, yet quickly reacting, model.

In the following chapters, more details on the implementation of the audio, video, and fusion algorithms are provided.

1.6 Outline of Thesis

In Chapter 2, the audio beat detection portion of the proposed audio-visual beat detection algorithm (AVISARME) is presented. Chapter 2 begins with a brief

introduction of audio beat detection algorithms, illustrating both the benefits and drawbacks of each method. The proposed method, its approach, and how it is implemented is then described in greater detail. A step-by-step implementation is provided, specifically looking at robustness, speed, and accuracy, and how the algorithm has been optimized using various optimization techniques. Bouguet's [16] pyramidal optimization algorithm for 2D optical flow is presented, and we introduce our adaptation of this technique for the 1D case.

In Chapter 3, the visual portion of the algorithm is analogously studied. The chapter begins with the origins of visual beat detection, and then our first-of-its-kind, full scene visual beat detection algorithm is introduced. As with the audio portion, the proposed method is analyzed in terms of robustness, speed, and accuracy. The optimization performed on the method to increase both accuracy and temporal resolution is subsequently illustrated.

Chapter 4 highlights the integrated sensor fusion/reinforcement learning portion of the algorithm. A short introduction to sensor fusion is presented to provide insight into the different ways information from two different sensor modalities can be utilized coherently. A number of artificial learning algorithms are also presented as they relate to their implementation in beat detection. A step-by-step analysis of this step is then offered, presenting a novel method of policy switching as well as the Markov Modelling used for determining the thresholds. The chapter concludes with a discussion on the effectiveness of this method, as well as a proposal of ways to improve the cohesion of the two sensors.

Chapter 5 presents extensive experimental results of the proposed algorithm,

AVISARME. Experiment 1 tests the audio portion of the algorithm against other popular audio beat detection algorithms. It is shown that on average, the proposed Overall Amplitude Approach method is more accurate than the most popular algorithm currently proposed. Experiment 2 attempts to characterize the circumstances that may result in failure of the audio algorithm. Simulating music signals using trains of pulses, tempo fluctuation is varied and Gaussian noise is added until the system fails. According to the tests, the proposed method is extremely proficient at handling noise at reasonable to high levels before the system fails. The system is more sensitive to tempo fluctuations, yet the error is linearly related to the standard deviation of fluctuation, even at extremely high levels. Under moderate to high levels of fluctuations, the system once again demonstrates robust performance. Comparing the results to previously reported experiments, the proposed audio algorithm proves to handle induced error with more proficiency than other methods. Experiment 3 tests the complete audio-visual integrated algorithm using a series of eight beat patterns, performed and recorded live by both a camera and microphone. For all but one pattern, the proposed method was in better synchronization with the human musician than was the baseline method. The chapter also presents further experimentation which analyzes the computational efficiency of the algorithm, as well as the effectiveness of the fusion and learning algorithms. Using the appropriate variables, the proposed algorithm is twice as efficient as the baseline approach, and the fusion/learning algorithms are shown to be quite effective in increasing the stability of the overall system.

In Chapter 6, we implement Murata's Spectral Temporal Pattern Matching

algorithm for audio beat detection. By following the steps and recommendations outlined by Murata [17], his algorithm is reproduced in order to test the approach more extensively and find ways in which it can be improved upon. In this chapter, it is shown that our implementation performs just as well on noise-free signals as does his. It is then run on the entire testing database to see how it stacks up against the proposed method. Chapter 6 concludes with a discussion of the STPM approach and provides specific recommendations on how to improve its performance.

Chapter 7 provides the conclusion to the thesis, as well as additional proposals for future research.

Chapter 2

Audio Beat Synchronization

2.1 Overview

In this chapter, the audio beat detection algorithm is dissected and explained in greater detail. While it is based off the work of Scheirer [9], specifically the idea of beat detection using Comb Filter Convolution (CFC), the proposed technique is unique in that it is the only viable method which does not utilize filter banks. Comb Filter Convolution is the process of convolving combs of different tempos with a music signal to convert it to the frequency domain. A comb can be thought of as a series of pulses separated by a constant time, representing a certain tempo. One of the major contributions of Scheirer is the idea of splitting the recorded signal into multiple frequency filter banks and running CFC on each bank independently to find the most reliable prediction of tempo. While this preserves the high resolution of the original signal, it requires far too many calculations to be considered for live applications. Murata [17] attempted to remedy this by performing cross-correlation on a lower resolution signal and looking at all filter banks simultaneously in a process called Spectro-Temporal Pattern Matching. However, the lack of resolution is quite troublesome, and this process does not solve the issue of bin separation (i.e. each bin is still considered a separate entity in terms of determining tempo). The proposed solution is starkly different from Murata's approach, and in some respect, multi-

faceted:

The main difference between the proposed algorithm and Scheirer's algorithm is that the proposed beat detection algorithm does not split the original signal into multiple banks, but instead utilizes the Overall Amplitude Approach, or one single bin. With advanced pre-processing techniques to identify and amplify musical beats, CFC needs only to be performed on the one bin as opposed to many. This also allows the detection of beats of all frequencies, not just one. The second main difference is the implementation of a pyramidal optimization scheme to locate the primary tempo. Consequently, the algorithm is able to detect the tempo of the music with a great increase in tempo resolution, all while calculating a much fewer number of CFCs.

The complete process is described more in the following sections.

2.2 Robustness

With the exception of Kozima [1], all of the major publications regarding audio beat detection employed a system of filter banks to split the original signal into separate frequency bands. Although this allows the program to differentiate between different instruments and possibly extract beat structure, it does not guarantee that one of the bands contains a good representation of the beat. For example, as illustrated in Figure 2.1, if the beat was comprised only of bass kicks (a), Scheirer's method [9] would be able to construe the beat. However, if the overall beat was comprised of individual beats in different frequency bands (b) (i.e. a bass drum,

snare drum, and cymbal), this method would fall short.

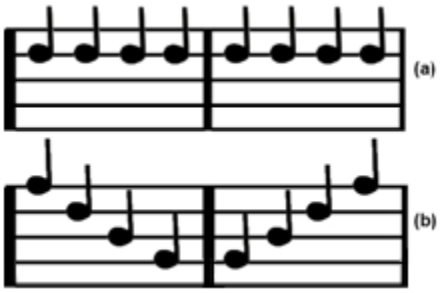


Figure 2.1: An illustration of a beat pattern which would work for Scheirer's method and the STPM approach (a), and one which would not (b).

Murata [17] realized the flaw in Scheirer's proposal, and instead of looking at each band independently, he surmised a method of pattern matching that considered the different bands simultaneously. While this does create a more efficient approach, it does not fully solve the issue, either. In fact, it still does not compare a beat in one bin to a beat in another, and would not be able to detect the pattern in Figure 2.1 (b). In the proposed method, we attempt to solve this flaw by only considering the signal in its original form. By utilizing the Overall Amplitude Approach, it is assumed that each sudden peak in music is a candidate beat, and that given a good filtering technique, it is possible to distinguish between actual beats and false/half beats. In this manner, the technique can be applied to any genre of music, with any beat structure, regardless of its complexity. This is accomplished in the following manner:

The first step in beat detection is clearly to record the music. In the proposed algorithm, a 2.5s recording window is used. This is long enough to get an accurate reading with a 3-tooth comb at 60BPM, but short enough to be considered real-

time. The 2.5s window guarantees that atleast 2 peaks will always be measured. For example, at 60BPM, the slowest tempo being considered, the window must be atleast 2 seconds to make this claim. The added half a second is for added accuracy. The choice of sample rate is also extremely important and can affect the performance of the algorithm greatly. It was found that a sample rate of 10kHz works extremely well, obtaining the best compromise between speed and performance.

Once the music is recorded, the data is first filtered to amplify candidate beats. This is done by bringing the entire signal, piecewise, to the fourth power. This effectively treats all peaks in the music greater than 0.6 (in normalized amplitude, measured in volts), as a candidate beat. This process decreases the non-beat amplitudes by an order of magnitude and appropriately weighting more prominent beats. The effect can be seen in Figure 2.2.

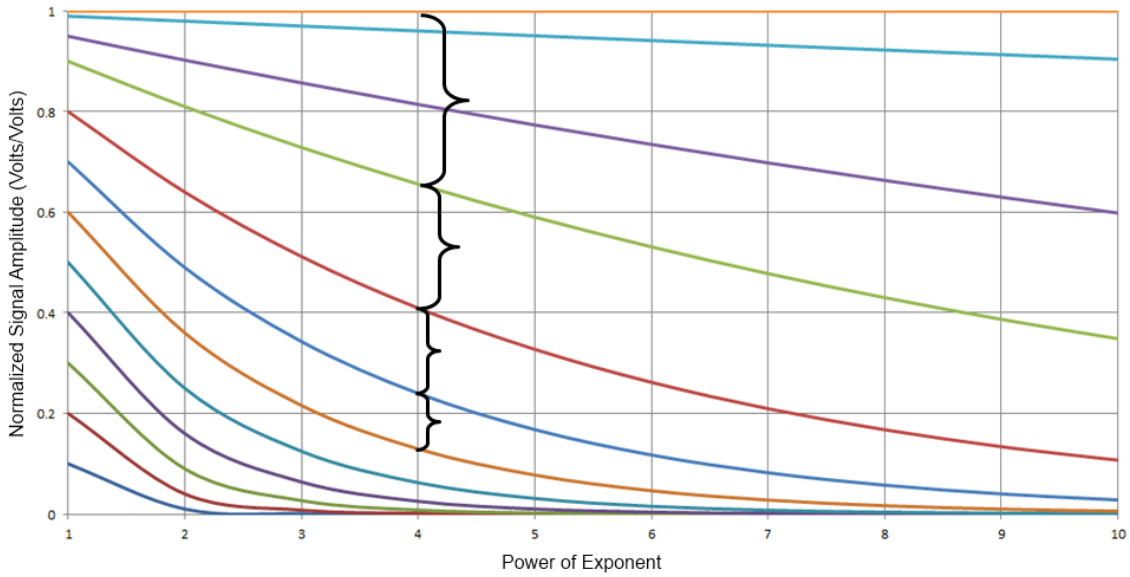


Figure 2.2: Beat amplification by 4th power exponent

A low pass filter is then applied to eliminate random noise and then normalize

high peaks which may dominate the system. The signal is then converted to the time domain using convolution of comb filters, as described in the following equation:

$$S(k) = \sum_{x=1}^{X} \left(\sum_{n=0}^{N} s\left(x + n\left(r\frac{60}{k} \right) \right) \right)^{6} \tag{2.1}$$

and,

$$BPM = arg\max_{k} \; S(k) \tag{2.2}$$

where: S = frequency-domain representation of the data

s = time-domain representation of the data

X = total number of points in the time domain

n = number of teeth per comb

r = sample rate

$k \in K$ = set of tempos in the search domain

Note: $k/60$ = frequency of the comb in Hz

In general, the 6th power exponent in equation 2.1 is normally a squared term to conform to energy conservation. The reason for this change is described in more detail in the Accuracy section.

2.3 Speed

Mizumoto's Spectro-Temporal Pattern Matching [3] is an elegant approach to deducing beat structure and tempo, a shortcoming of Scheirer's work [9]. However,

in doing so, the computations significantly increase in number. The robot not only has to convert each frequency band from the time domain to the frequency domain at each time step, it also must apply an extensive pattern matching algorithm, a 2D Normalized Cross Correlation, to deduce the beat structure.

As Kozima [1] proposed, beat detection could be performed without any knowledge of beat structure. Naïve of the frequencies which comprise the beat, one can look at the overall amplitude of the signal to infer when a beat has occurred and to predict when the next beat is supposed to occur. Doing so, the amount of necessary calculations is limited to a single time-domain / frequency-domain conversion.

To break the dependence of calculation time on tracking resolution, a custom 1D Pyramidal Optimization scheme has been implemented. More information on this scheme is included in the Optimization section.

2.4 Accuracy

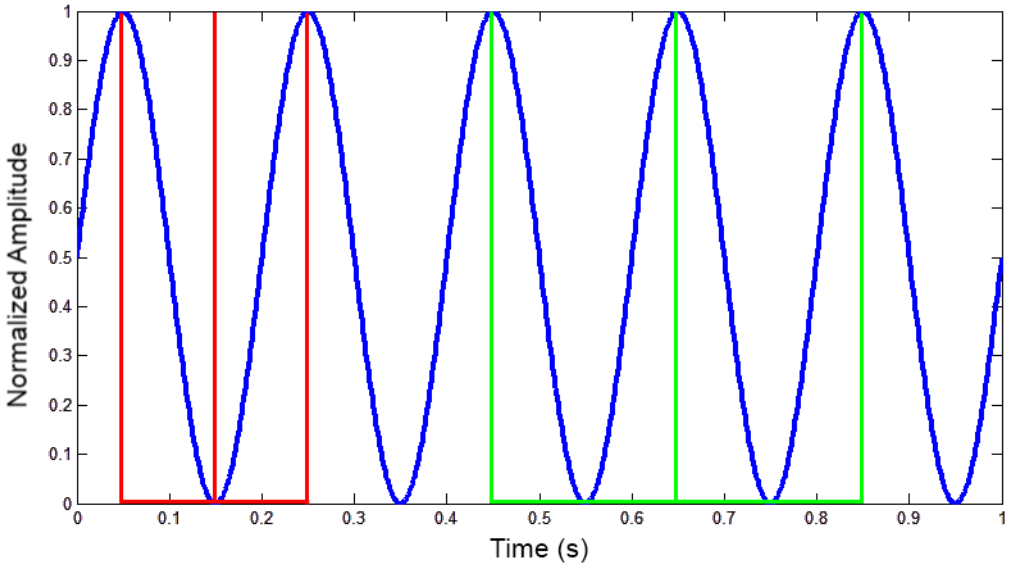
As previously mentioned, the proposed method treats every increase in amplitude as a candidate beat. Although this simplifies calculations drastically, it becomes difficult to determine which peaks represent musical beats and which peaks arise from error. Due to this, a simple Fourier Transform would produce a frequency domain signal which is not representative of the actual tempo. Furthermore, performing an FFT on the signal would yield an extremely low resolution result. Although the FFT would yield a resolution of 0.4 Hz, this translates to a resolution of 24 BPM, far less than what is needed for this application.

After performing preprocessing on the signal to aide in beat detection, the music signal is passed through a unique convolution (2.1), converting it to the frequency domain. In this method, comb filters of varying frequencies are convolved with a filtered version of the signal to get the spectral energy at each of the individual frequencies. In this manner, a resolution as small as 0.02 BPM can be achieved if sampled at the 44.1 kHz norm (ANSI-IEC 60908).

However, as Scheirer [9] noted, there is an issue with double and half beat detection. In other words, if the musician plays a piece at 120 BPM, his method might detect 60 or 240 BPM, along with the correct 120 BPM. Mizumoto [3] avoided this issue by ignoring all tempos faster than 120 BPM. Unfortunately, in practice, if the robotic musician were to play at half or twice the speed as the performer, this would be unacceptable. Our method proposes a way to decrease the occurrence of double and half beat detection rather than avoiding the issue. The results, shown in figure 2.3, demonstrate the method's effectiveness.

According to Parseval's theorem [18], when converting from a time domain signal to its frequency domain representation, energy is related to the sum of the squares of the signals. In applying this theorem to a convolution of comb filters, one sums the square of the individual combs, providing the spectral energy of the system at the corresponding frequency. This approach (the normal approach) is referred to as a second power comb filter. However, there is no evidence to support that this provides the best representation of the tempo of a piece of music, and there is no evidence that the affects of a higher order comb on tempo detection has ever been investigated.

Disregarding the fact that there is no apparent physical meaning (i.e. energy) for higher order combs, it is quite intuitive as to why higher order comb filters weight the correct BPM significantly more than it does integer multiples. For example, assume there is a perfectly sinusoidal signal with amplitude ranging from 0 to 1. A comb of the same frequency of the signal would be weighted much greater as it spans three peaks at one time, as opposed to spanning only two. This can be seen in the ratios in 2.3.



	Base Frequency Comb	2x Frequency Comb	Ratio
Sum	2	3	1.5
Sum^2 (Parseval's Theorem)	4	9	2.25
Sum^6 (Proposed Method)	64	729	11.39

Figure 2.3: Diagram showing the effect of a higher order comb filter on the chance of double/half beat detection.

The difference between a comb which spans three peaks and a comb which spans only two would be greatly exaggerated when taken to a higher power as compared to a second order filter. This becomes extremely important when distinguishing double/half beat detection. The larger the ratio is between the correct tempo and the incorrect tempo, the more reliable the overall detection. Although this example is relatively simple, it can obviously be applied to more complex systems.

In the proposed algorithm, a 6th power comb filter is used. Although the higher the power, the greater the distinction between double/half beat detection, too high a power becomes troublesome for complex beat patterns. At high powers, strong half-beats and intermittent notes may confuse the system and overpower the signal. As seen in Figure 2.4, and backed by experimental data, the 6th power comb filter is large enough to discriminate against most half/double beat detection yet small enough as to be unaffected by strong non-beat notes.

2.5 Optimization

In an attempt to optimize audio beat detection, the comb-filter convolution has been streamlined by decreasing the sampling frequency and by keeping the signal in one broadband frequency bin. More importantly, a pyramidal optimization (PO) scheme has been developed in order to break the dependence of resolution on the number of convolving combs. The idea of using PO to decrease the amount of calculations was inspired by Bouguet's paper on his pyramidal implementation of the Lucas-Kanade feature tracker for OpenCV [16, 19].

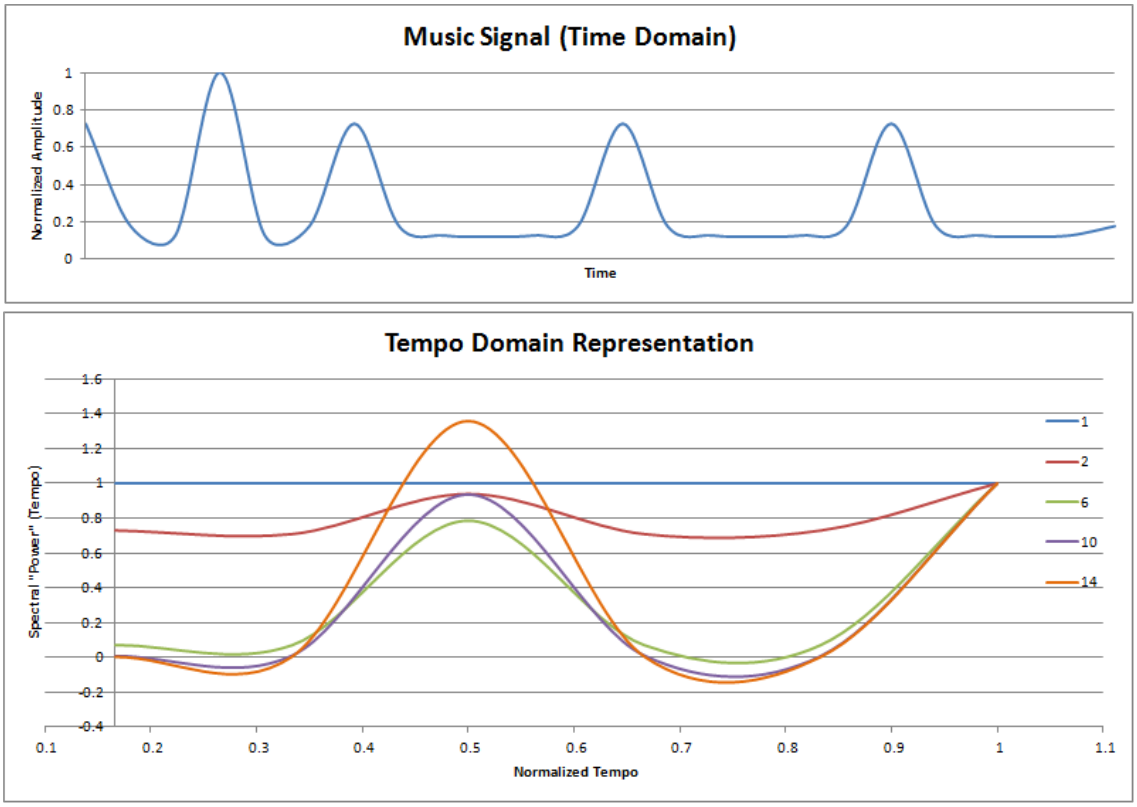


Figure 2.4: Half beat distinction by use of higher power comb filters.

2.5.1 Bouguet's Pyramidal Optimization for 2D Feature Tracking

When performing feature tracking, the main goal is to pick points on an object which are easy to differentiate from their surroundings, and determine how the localized area has changed from the previous frame. If these points are markers, such as white balls as commonly used in cinematography, finding the trajectory of the points is a simple case. Unfortunately, in most cases, clear markers are not available, and instead, we must rely on image processing techniques such as eigenvalues, image gradients, and optical flow.

In order to select adequate points on an object to track when markers are unavailable, programmers turn to computing eigenvalues and/or local gradients of

the image. Although the techniques are quite different, both assign values to each pixel according to how different the intensity of the pixel is with respect to its neighbors. The larger the value, the more distinct the point is, and the easier it will be to track. It is those pixels with the largest assigned values that are selected to be tracked.

Once the points are selected, individual features can be tracked. Although this can be done in many ways, one of the most commonly used techniques, and the one Bouguet uses to implement his pyramidal optimization, is Lucas Kanade optical flow. Lucas Kanade optical flow is generally calculated in the following manner.

First, the frame, I_0, at time t=0 is captured and stored to memory. The location of each point is recorded. A window, w, with width w_x and height w_y is drawn around each point; the area of the original image inside this window will act as a local sub-image for future calculations. The camera then captures the second frame, and the nearest neighbor to the original subimage is found in the new image.

To decrease computations, only a small area of I_{t+1} is searched, or more specifically, we search only within the window surrounding the previous position of the point. The error, ε, or difference between the subimage at time t and the subimage at time t+1, is minimized by using the nearest neighbor rule to find the displacement vector, d, which minimizes the following error function.

$$\varepsilon(d) = \varepsilon(d_x, d_y) = \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{y=u_y-w_y}^{u_y+w_y} \left(I_t(x,y) - I_{t+1}(x+d_x, y+d_y)\right)^2 \qquad (2.3)$$

where: ε = error function

d = residual pixel displacement vector

u = position of point at time t

$w = 1/2$ the window size

I = grayscale image of the frame

Although it is not obvious from the above equation, the number of calculations is related to the fourth power of the window size. (There are $w_x * w_y$ pixels in the sub-images, and in order to search the entire window, each pixel needs to be convolved $w_x * w_y$ times. Assuming the window is square, $w_x * w_y * w_x * w_y = w^4$.) Therefore, assuming one cannot find a sufficient nearest neighbor within the window, it is impractical to simply increase the size of the search window as calculations will increase quartically.

To deal with this, Bouguet has conceived a pyramidal scheme, a form of Direct Search, which first searches a lower resolution representation of the image in order to find an initial, coarse estimation of the new location. The process is best described in Bouguet's own words:

"The overall pyramidal tracking algorithm proceeds as follows: first, the optical flow is computed at the deepest pyramid level L_m [the lowest resolution image]. Then, the result of that computation is propagated to the upper level L_m - 1 in a form of an initial guess [g] for the pixel displacement (at level L_m-1). Given that initial guess, the refined optical flow is computed at level L_m - 1, and the result is propagated to level L_m - 2 and so on up to the level 0 (the original [resolution] image)." [16]

The previous equation then can be modified slightly to fit the new scheme:

$$\varepsilon^L\left(d^L\right) = \varepsilon^L\left(d_x^L, d_y^L\right) = \sum_{x=\ u_x^L-w_x}^{u_x^L+w_x} \sum_{y=\ u_y^L-w_y}^{u_y^L+w_y} \left(I_t^L\left(x,y\right) - I_{t+1}^L\left(x+g_x^L+d_x^L,\ y+g_y^L+d_y^L\right)\right)^2 \tag{2.4}$$

Notice how the size of the window does not decrease with decreasing resolution. With this in mind, the advantage is clear the initial guess can be extremely far from the previous location of the object, allowing the Lucas Kanade optical flow algorithm to detect the large distance without any increase in window size. Astonishingly, it has been shown that a 3-level pyramidal scheme can provide gains in displacement vectors up to 15x, while only increasing calculations by 3x. [16]

2.5.2 Background Information

Bouguet's pyramidal optimization for 2D feature tracking is especially useful for tracking objects with large displacements, and where the location of the object at one point in time is loosely dependant on its position at the previous time step. For human movement, however, this is not the case. Humans inherently move slowly (as compared to the refresh rate of a camera at 30 FPS), and we have significant inertia, forcing us to move at least semi-fluent movements. Therefore, for the case of 2D tracking of the musician, one is able to use a small window, zero-th level (non-pyramidal) optical flow algorithm, decreasing the amount of necessary calculations.

Music, however, is not this way. Although the tempo of a popular song remains fairly consistent throughout its entirety, this is not always the case. Improvisational

music, classical music, etc, can change tempo on a whim, and this provides a huge problem for conventional beat detection since statically converging learning techniques cannot be used. It is preferable to have extremely high resolution for closer synchronization, however, we must also search over a wide range of tempos. It would not be wise to perform beat detection at a high resolution over a short span of tempos, as the tempo at one time step may be independent of the tempo at the previous time step. Each concurrent dataset must then be assumed as independent, and a search over the entire domain must be performed. On the other hand, having to search a large span of tempos (60-240 BPM) at each time step can be costly if performed at a high resolution. This issue is analogous to the problem Bouguet solved with his pyramidal optimization for 2D feature tracking. Therefore, we decided to generalize his method for use with a single dimensional case.

In Bouguet's method, he defines a few terms, such as level, height, window, and guess. He defines level as the current tier in the pyramid, height as the total number of tiers in the pyramid, window as the area of the cross section of the period at height h, and guess as the current estimate of new location which is passed from level to level. For consistency, the analogous properties in the proposed search are referred to by these terms. However, a few more terms are also introduced.

Resolution is defined as the difference in BPM between two consecutive comb filters. For instance, if 5 comb filters are used over a 10 BPM span, the resulting signal would have a resolution of 2 BPM, as it is the smallest increment for which one can declare a tempo to be.

Blur is defined as the order of the low pass filter applied to the signal with

respect to the total number of points in the signal. For instance, if a moving average of 10 points is applied to a data series of 100 points, we say we have a blur of 10%. The higher the number, the more the original signal is distorted.

Slope is also defined to be the rate at which the pyramid will decrease in width. At each step, the window and resolution (widths) of tempos which will be searched will decrease linearly at this rate. The blur (area), will decline in relation to the square of the slope. This notion will be discussed in more detail in the following sections.

With these terms now defined, the implementation process can now be described in detail.

2.5.3 Pyramidal Optimization for Beat Detection

As previously described, once the music is recorded, in the normal, non-pyramidal beat detection, the data would be filtered, normalized, and then converted to the time domain using convolution of comb filters, as described in the following equations:

$$S(k) = \sum_{x=1}^{X} \left(\sum_{n=0}^{N} s\left(x + n\left(r\frac{60}{k} \right) \right) \right)^{6} \tag{2.1}$$

and,

$$BPM = arg\max_{k}\ S(k) \tag{2.2}$$

where: S = frequency-domain representation of the data

s = time-domain representation of the data

X = total number of points in the time domain

n = number of teeth per comb

r = sample rate

$k \in K$ = set of tempos in the search domain

Note: $k/60$ = frequency of the comb in Hz

Although the basic form of (2.1) does not change when converting to the pyramidal version, (2.2) does not come into play until the final level. Instead, in the first and intermittent levels, only the search domain, K, changes and (2.2) gets replaced with (2.5).

$$K = \{arg\max_k \ S(k) - w \ : \ arg\max_k \ S(k) + w\} \tag{2.5}$$

where w is the window size. Both the size of the window and the step size (resolution) are determined by the slope. Since it is difficult to understand the process by examining these three equations, a step by step procedure for implementing 1D pyramidal optimization is presented.

Once the music is recorded, the signal is copied to another array and a low pass filter is applied. This new array will constitute the base level of the pyramid. At this level, a window of 180 BPM (60-240) and 91 combs are used, corresponding to a coarse resolution of 2 BPM. Normally, a resolution of 2 BPM would be insufficient for beat detection; however, as this is merely the first level of the pyramid, we

are simply trying to obtain an initial guess for the true tempo. One important note here when Bouguet compresses the original 2D image into a lower resolution representation, there is an automatic image processing technique which averages neighboring pixels in order to represent multiple pixels as a single one. The more compressed the image becomes, the more averaging that must occur. An example of this can be seen in Figure 2.5. [20]

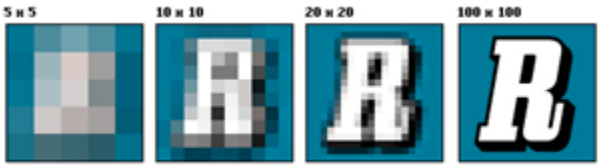


Figure 2.5: Image showing averaging of pixels as a function of image size. Notice how as the number of pixels decreases, the blurring effect increases

Similarly, the music data is also "compressed" by applying a relatively high order moving average to the lowest level. At the first level, a centered moving average filter of order 512 is applied, which corresponds to a blur of roughly 2%. This blur has the effect of smoothing the data, not only in the time domain, but also in the frequency domain. As shown below in Figure 2.6, this aids in providing a reliable initial guess to propagate to the next tier.

There are two things to note in Figure 2.6. First, the distribution corresponding to the filtered data is much smoother. The averaging process widens out the distribution, making it more likely that the coarse comb will detect the hump. In addition, the distribution becomes less sensitive to local disturbances, and will present a much more reliable estimation of the predominant tempo. The second thing to

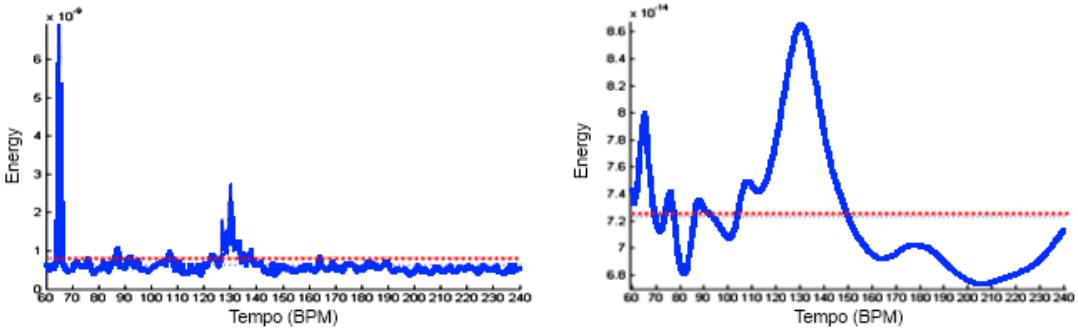


Figure 2.6: Graph showing the effect that the order of the low pass filter has on the spectral energy distribution

notice is that in this particular example, in the case of no smoothing, the half-beat was selected as the predominant frequency. However, when a 2% blur was applied to the data, the area of the correct BPM was discovered. This can be attributed to the imperfections of the artist. Although the artist may intend to play at exactly 130 BPM, he may fluctuate from 127-133 BPM. Although a human listener may not be able to distinguish the difference, the computer can distinguish offset differences down to a tenth of a millisecond. By applying a blur to the music, it induces the uncertainty in offset similar to that of a human. In doing this, the computer will choose to search the span of tempos where there is the largest probability that it contains the correct tempo (124-136 BPM), instead of simply selecting the tempo corresponding to the greatest spectral energy (65 BPM).

The mean of this span (130 BPM in the previous case) is then passed to the second level of the pyramid. In our implementation, a pyramidal slope of 4 is used. Therefore, the resolution of this layer will be 0.5 BPM (2 BPM/4), and a blur of 0.128% ($2\%/4^2$). This is analogous to the 2D version as the length of the window

would decrease linearly with slope of the pyramid while the amount of pixels in the image is correlated with the square of the slope. To cut down on calculations, a window of 12 BPM (±6 BPM) is used as opposed to the 22.5 BPM window as suggested by the slope/2D analogy. The comb filters are then convolved with the music data, and the new guess is passed on to the next level. This process is then repeated until the desired resolution is achieved.

In Figure 2.7, the outputs for each layer of the pyramidal scheme are shown. The widths of the distributions represent the length of the windows used while the coarseness of each distribution can be attributed to the decreasing blur factor and increasing resolution at each layer. Notice how although the initial estimate is extremely close to the true tempo, it was not until the third level was reached that the true tempo was found.

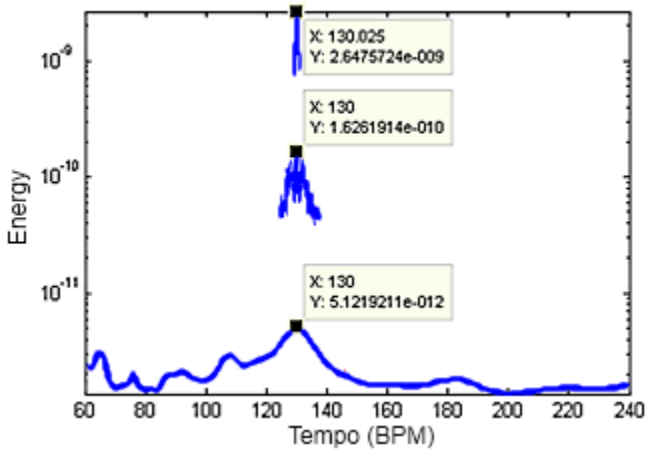


Figure 2.7: Graph showing the tempo distribution of the different layers of the pyramidal scheme for the same song as in Figure 2.6

Under the normal approach, 181 convolutions would have had to be computed

to achieve a resolution of 1 BPM. Amazingly, using the pyramidal optimization, a resolution of 0.125 BPM (8x better) was achieved by computing only 126 convolutions (91 for the base layer + 25 for the second layer + 10 for the final layer). Previously, 1441 convolutions would have needed to be performed to achieve the same resolution. That's a speed gain of 11x!

2.6 Conclusions

While the proposed audio algorithm is based on an old principle, it incorporates new methods which increase both robustness and speed. The Spectro-Temporal Pattern Matching approach described in this section is a quite robust approach, able to handle most types of music, however what maintains is the original issue of bin separation and poor resolution. While differences in bins are all compared simultaneously, each bin is essentially considered a separate entity while performing beat detection. In other words, if a beat pattern consists of notes in the form of Figure 2.1, the beat detection algorithm would surely fail. By analyzing the original signal in its unaltered form, notes of all frequencies are compared at the same time. Along with utilizing a novel 6th power comb filter, the resulting algorithm can handle virtually any genre of music and beat pattern.

A few methods for increasing the efficiency of the algorithm have also been introduced. Since the proposed approach operates in the time domain, the resolution of beat tracking is only limited by the sampling rate of the recorded audio. Though human ears are extremely sensitive, needing every ounce of the 44.1 kHz sample

rate used, this is far more than what is needed to detect tempo. By decreasing the sample rate to 10 kHz, an extremely high resolution is maintained, and the number of calculations is decreased by 77%. However, it is suggested that optimization be performed on the sampling rate. While 10 kHz served us well in terms of accuracy and speed, there is most likely some other optimal value for sample rate which serves as a better compromise between speed and accuracy.

The more important extension to Scheirer's algorithm which affects both speed and resolution is the implementation of the pyramidal optimization (PO) scheme. By implementing an optimization scheme which is both time invariant and largely insusceptible to getting "stuck" in local maxima, the dependence between resolution and number of calculations has been broken. As Hitz [15] argues, the main flaw of the traditional CFC method is the fact that detection resolution is highly dependant on the number of combs convolved with the signal. By utilizing the optimization scheme, a resolution gain of 8x has been achieved while decreasing the number of combs by 30%.

Chapter 3

Visual Beat Synchronization

3.1 Overview

While the field of visual beat detection is still in its infancy stage, there have been a few approaches to integrated beat detection which include visual feedback. Both Hoffman [12] and Mizumoto [3] have incorporated gestures into their robotic musicians to instruct the robot to begin, end, take the lead, etc, however the gestures were never used to aid in tempo detection itself. Lim [6] extended Mizumoto's work to include gestures within the detection algorithm, however the method of implementation only allowed for synchronization with a flutist, and required the flutist to perform special gestures while playing. Itohara [21] later extended the original work of Mizumoto to allow for synchronization of the robot with a guitarist. While this method does not force the guitarist to perform gestures, it can still only be used for a single instrument.

The video beat detection algorithm proposed in this chapter utilizes Lucas Kanade Optical Flow for Full Scene Motion Tracking. In other words, feature tracking is performed on the entire frame; there is no discrimination between body parts, instruments, etc. In this manner, the algorithm is extremely robust, able to detect all types of inputs, such as the nodding of one's head, the tapping of one's foot, the strumming of one's hand, and the movement of the instrument. Utilizing

a Microsoft Kinect (for background cancellation) and clustering techniques, beat tracking can be performed with both higher efficiency and precision than using a traditional camera.

3.2 Robustness

As recent as 2010, there were currently no published beat detection algorithms which incorporated visual inputs directly into the beat detection itself. That changed with a publication by Lim [6]. Enhancing Mizumoto's algorithm [3], she preprogrammed gestures that a flutist must carry out to signify that a beat has occurred. However, her approach could only be applied to flutes, and the flutist was required to have prior knowledge of the gestures. A gesture-based approach is also troublesome in that forcing a musician to perform gestures with the instrument while he or she plays may cause the musician to be distracted or uncomfortable, leading to performance degradation.

Itohara [21] also expanded on Mizumoto's original work by incorporating visual beat detection for a human guitarist. By using optical flow to track the movement of the guitarist's strumming hand, his algorithm is used to form a rough estimate of the musical tempo, and re-weight the particles in his particle filter approach to audio beat detection. Although this approach makes it unnecessary to perform awkward gestures when playing, it is still limited in application to only one instrument. Furthermore, the implementation only allows for a frame rate of 19 FPS which leads to a poor temporal resolution.

In the development phase, OpenNI's [22] full body tracking was initially used to track the motions of the musician. However, this approach was unable to pick up on finer movements such as the bobbing of the head or tapping of the foot. As only a few instruments (i.e. guitar, percussion, etc) require exaggerated motions of the limbs to play, this approach would also be limited in application. Furthermore, it was unable to detect the motions of the instrument and was inaccurate when the musician was visually obstructed by the instrument itself. To remedy this, Lucas Kanade Optical Flow [16] was used to simultaneously track the motion of up to 1000 points on the musician and instrument.

Using the intensity gradient of the video feed, optical flow can markerlessly track the entire body and instrument of the musician, allowing the analysis to detect even the smallest of motion. Once the best 1000 points were chosen, as described in the previous chapter, the movement is tracked by marking the position of each point at every time step. This is done by isolating a small area, w, around each point at time t, and then finding an equally sized area in the image at time $t+1$ which minimizes the error function shown in (2.3).

$$\varepsilon(d) = \varepsilon(d_x, d_y) = \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{y=u_y-w_y}^{u_y+w_y} \left(I_t(x,y) - I_{t+1}(x+d_x, y+d_y)\right)^2 \qquad (2.3)$$

where: ε = error function

d = residual pixel displacement vector

u = position of point at time t

w = 1/2 the window size

I = grayscale image of the frame

The proposed method can therefore be applied to not only guitars or drums, but also to the finer instruments which require little to no movement. A simple nodding of the head or tapping of the foot would suffice for accurate detection of tempo.

3.3 Speed

In order to detect movement in the video feed, optical flow is performed on a large amount of points in the image. However, performing optical flow on the background behind the musician provides data which is irrelevant to beat detection. Since these superfluous calculations lead to degradation in both speed and accuracy, points in the background are not selected. This can be efficiently done using the infrared imaging employed by the Kinect. By obtaining the depth of each pixel in the color image, all of the computing power can be focused on the musician and instrument, ignoring the scenery behind the musician.

Calculation speed is also extremely important because it directly affects the resolution of the beat detection. A large computation time between frames implies a lower number of Frames Per Second, which in turn leads to a lower BPM resolution. In order to get the most out of the 30FPS that the Kinect offers, a non-pyramidal optical flow scheme was selected with a small window size. As humans inherently move slowly and fluently when playing an instrument, this is taken advantage of,

allowing the proposed method to work seamlessly at high refresh rates. Itohara's approach [5] only allows for a 19FPS refresh rate, which significantly reduces the overall resolution of the beat tracking algorithm.

3.4 Accuracy

Although resolution is extremely important to the precision of detection, the proposed method is also extremely accurate due to its ability to pick up on the smallest movements of the musician. Of the 1000 points which track the position of the musician at each time step, only a couple hundred are indicative of the tempo, at best. This assumes that the speed at which the musician is moving is equal to the tempo of the music, which is generally the case. To increase the beat detection accuracy, as well as speed up the analysis, points which do not appropriately represent the tempo of the music are eliminated. This is done by analyzing the variance and frequency domain representation of each point, and performing anomaly detection prior to the beat detection. This includes removing any point moving slower than 1Hz or faster than 4Hz, and any point with a SNR which is deemed too low.

Of the few hundred points left, each of the time-displacement plots are transformed into the frequency-energy domain to find the natural frequency of each point. This is done using a standard FFT. If there are an overwhelming majority of points (greater than 60) moving at the same natural frequency, it is evident that the method had correctly detected the tempo of the music.

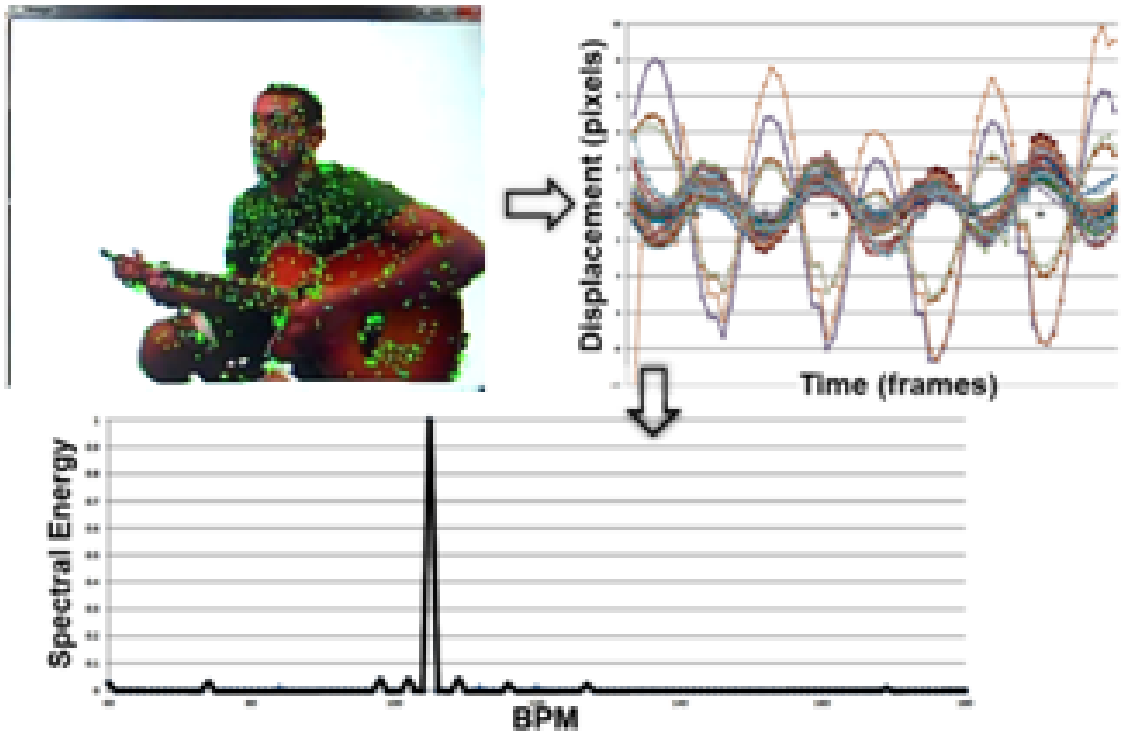


Figure 3.1: AVISARME tracking the motions of a musician. Musician was nodding his head to the beat as he is tracked by the beat detection algorithm.

3.5 Optimization

The entire visual beat detection requires such little computational power that it only takes between 0-10ms to compute. Since the audio beat detection takes roughly five times as long, and the two are computed concurrently, optimization needed only be done on the accuracy and resolution of the technique.

Although the full-scene method of visual beat detection, previously described, provides the correct tempo more often than not, there are cases where there are too few points moving at the same frequency to guarantee sufficient accuracy. These cases include picking on a guitar, foot tapping, etc, where there is only a small amount of mass moving. To remedy this issue, a simple clustering technique is

employed which searches for small groups of points moving in unison.

In order to do this, the ending location of each point is marked. Then, for each point, the distance is found to the ending location of each of the other points being tracked. If both the vertical and horizontal components of the distance are less than some threshold, the frequencies at which they move are compared. If enough neighbors (25 points) are moving at the same frequency, it can be concluded that the beats per minute of the music can be determined through the visual signal.

This is admittedly not the most efficient or elegant method, however, the bottleneck in speed of the overall algorithm is attributed to the audio analysis, not the video, and so the clustering efficiency has no bearing on the performance of the overall beat detection.

3.6 Discussion and Conclusions

The proposed video beat detection algorithm is the first of its kind to employ gesture-less and marker-less motion tracking for multi-instrument beat detection. While both the proposed method and Itohara's method are based on the same principle of optical flow, many optimization methods have been used to allow for more robust and efficient detection.

First is the notion of Full Scene Motion Tracking. Instead of pinpointing one specific body part (the strumming hand of a guitarist or the tip of a flute) and tracking only that motion, even the tiniest periodic motions of every body part can be detected. In experiments, the algorithm has been seen to detect the strumming

hand, the picking hand, the nodding of the head, the movement of the shoulder, and the tapping of one's foot, all at the same time. Relying on all the motions a human makes while playing an instrument produces a much more reliable and robust approach than would relying on just one body part.

Secondly, by utilizing background cancellation via a Microsoft Kinect, all of the computing power can be concentrated on the foreground, that which contains the musician(s) and the instrument(s). This in turn allows for a quicker frame rate of capture, which leads to a higher resolution in the beat detection algorithm. It might, however, prove useful to investigate other forms of video capture and background cancellation, such as stereo vision. While the Kinect can only perform background cancellation at 30FPS, stereo vision (with cameras which can shoot at 60FPS) would allow for higher resolution tempo detection as long as the distance calculation algorithm is efficient enough to perform at that speed. Furthermore, the optical flow which takes place between each frame might also need to be optimized to operate at such high speed. As long as both criterion are satisfied, stereo vision might prove to be a huge advancement of the current proposal. Another option would be to store the frames of the video feed and then performing the distance detection and feature tracking post-facto.

As shown in Experiment 3, the video algorithm tends to produce even a more reliable representation of tempo than the audio algorithm. The main drawback, however, is in the resolution, ± 3.5 BPM. If this resolution can be increased to ± 1 BPM, due to the high accuracy and extremely efficient calculation, visual beat detection may prove to be the foremost method of beat detection for simple robotic

musicians and micro controllers. The issue with visual-only beat detection, especially this method, is that it is impossible to determine what movements of the musician represent beats in the music. For example, on a guitar, both an up-stroke and a down-stroke can compose the beat, and without knowing which strokes caused notes to be played, it is impossible to determine the onset of the beats. Once the tempo is determined via video, it might be possible to determine onset times by quickly analyzing the audio signal for peaks without performing lengthy beat detection. For certain instruments, utilizing only the depth information and instrument detection might be sufficient for this purpose, as well. However, methods such as these has not been implemented in this paper.

Chapter 4

Audio-Visual Fusion and Temporal Difference Learning

4.1 Overview

Sensor Fusion is an emerging issue in today's research. When given two or more noisy sensors, Sensor Fusion attempts to combine the readings into a more dependable sensor for which to base actions on. This is extremely important when the sensors are a heterogeneous set; that is, sensors that measure different entities, such as an accelerometer and gyroscope, or in this case, a microphone and a video camera. This is because while the two sensors might measure the same event, they represent the event in different ways. For example, although both a gyroscope and accelerometer can be used to determine the orientation of a body at rest, one must use sensor fusion to determine the complete state of the body if the object is moving. Assume a robot has both of the aforementioned sensors located at the center of mass. If the robot is linearly accelerating at an angle, from the accelerometer data alone it may appear that the robot is tilted but at rest. It is important to also have the gyroscope to determine the true orientation of the robot in order to adjust for that in the accelerometer readings, and visa versa. Taking both readings independently and choosing the "best signal" would provide nonsensical information about the dynamics of the robot.

Sensor fusion provides a more reliable system for detecting events than would

considering both signals independently. This remains true regardless of whether or not the sensors' measurements are independent of one another. In this case, both a microphone and camera are used to detect beats in music. Furthermore, while both the audio and video signals are dependant on the tempo of the music, the actual measurements may or may not be dependant on one another. For example, suppose there is a drummer hitting a drum at a constant tempo. Each time the drumstick contacts the drum, it corresponds to both a peak in amplitude in the audio stream, and a trough in the position of his hand in the video stream. Both the audio and video analysis should be representative of the tempo at which he is playing. If both sensor readings always provided the same correct reading, sensor fusion would be pointless. There will always be differences in readings due to error, and this must be corrected by fusing the sensors in such a manner that the correct tempo is always provided.

However, suppose that the drummer is now playing a sophisticated 4 over 3 beat, where his right hand plays 4 notes during the same time that his left hand plays 3 notes. In this case, the audio will show the correct tempo (with smaller peaks corresponding to 3x and 4x of the tempo) while the audio will show two peaks at 3x and 4x, neither of which is the correct tempo. This is where sensor fusion is extremely important, and Temporal Difference Learning is utilized.

In 2004, both Takeda [23] and Klapuri [10] proposed a probability based approach to tempo detection using a Hidden Markov Model. While the probabilistic approach was novel, and HMM's prove to supply a stable prediction of tempo, they are inherently only as quick to compute as there are states in the system. Since

tempo is continuous by nature, one must either discretize the tempo into bins, risking losing precision, or use a large amount of bins at the expense of computational efficiency. In 2010, Otsuka [24] proposed a particle-filter based approach to beat detection and score following to break the dependence of computational efficiency on the number of tempo bins. Using the the spectro-temporal approach to beat detection, Otsuka used his estimation of tempo to predict upcoming notes in the piece, re-sampling the weights of the particles with each estimation of tempo based on the normalized cross-correlation output. Even more recently, Itohara [5] has expanded this to include the input of his visual beat detection algorithm. While tracking the hand of the guitarist, Itohara uses the position information to determine a coarse estimation of the tempo in order to re-sample the weights of the particles. However, both algorithms are still highly dependant on the number of particles, which in turn, increases the computational cost of the algorithm.

Temporal Difference Learning [25], on the other hand, is a reinforcement learning technique which is extremely computationally inexpensive, no matter the amount of states in the system. Similar to Markov Chains whose probability density function is only dependant on that of the previous time step, Temporal Difference Learning uses only the predicted value of the previous time step to predict the next value. Depending on the choice of variables, due to it's converging nature, TD learning can either provide a reliable estimate but be slow to react, or quickly reacting but produce an estimate which is sensitive to noise. In order to achieve the optimal compromise between stability and quickness of response, a technique of policy switching inspired by Comanici [26] has been adopted. In this paper, Comanici et. al, pro-

posed a method of policy switching for reinforcement learning based on termination functions determined by the underlying Markov Decision Process, or MDP. For our implementation, the system was modelled as a Markov Decision Process, and when the uncertainty of the new tempo decreases below a certain threshold, TD learning is re-initiated with the new tempo as the initial value. This chapter will describe this part of the algorithm in more detail.

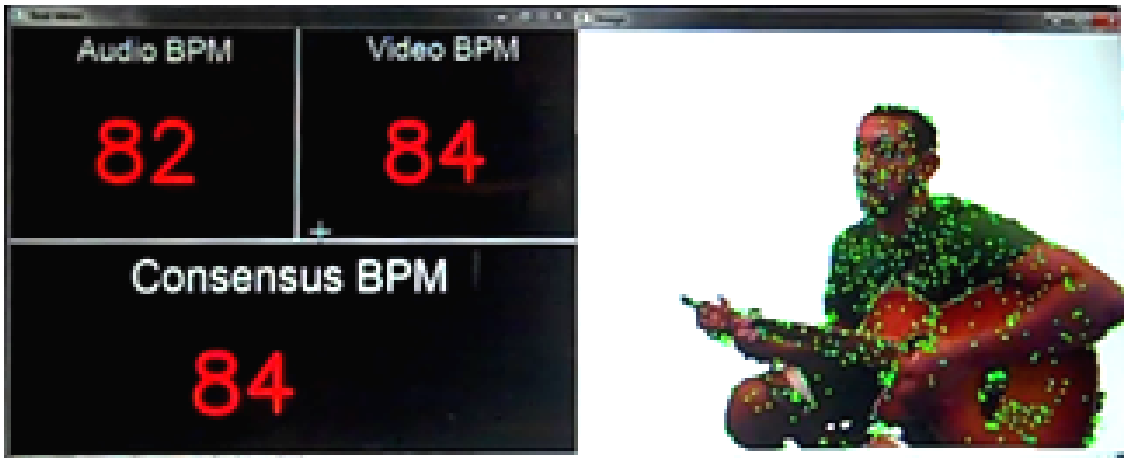


Figure 4.1: AVISARME using both the Audio and Visual Beat detection algorithms to find the tempo of the music

4.2 Sensor Fusion

In 1990, Luo presented a classification of Sensor Fusion techniques by the point at which fusion is applied [27]. According to Luo, there are four levels: signal, pixel, feature, and symbol. The signal level can be thought of as an additional step in signal pre-processing, before it is converted into representable data (i.e. pixels/images). Signal level is generally used to decrease noise and temporal misalignment of data.

The pixel level can be used to extract more information from each individual pixel, such as in segmentation. Feature level fusion can be used to combine features from multiple signals to increase the feature measurement accuracy, or span of features recognized. Symbol level fusion is applied at the highest level of abstraction and is generally used to increase the certainty of probability distribution functions used for decision making [27, 28, 29]. Our sensor fusion is incorporated into both the pixel level and the symbol level.

Sensor fusion is first used in the visual beat detection for background cancellation with the purpose of more efficient tracking. This is used to eliminate the part of the image behind the musician such that the algorithm cannot track movements that do not relate to the music. This is done by fusing the RGB image with the depth-gradient image, overlaying one image on to the other, and deleting any pixel in the RGB image which is above a set threshold in the depth image.

However, the main fusion takes place at the highest level of abstraction, the symbol level. After a tempo is determined from each the audio and video signals, the readings are compared in order to form a more accurate estimate of the tempo. Figure 4.2 shows an example of what is passed into the fusion step of our algorithm. In this example, it is quite easy to see an agreement in approximate tempo. But how is the true tempo obtained from this bimodal distribution?

The first step in our symbol level fusion is to obtain the Signal to Noise Ratio, or SNR of the signals in the frequency domain. For the audio beat detection, this is calculated as a Power-to-Power ratio, by dividing the maximum power by the root mean squared of the span of frequencies calculated in the convolution step. Since the

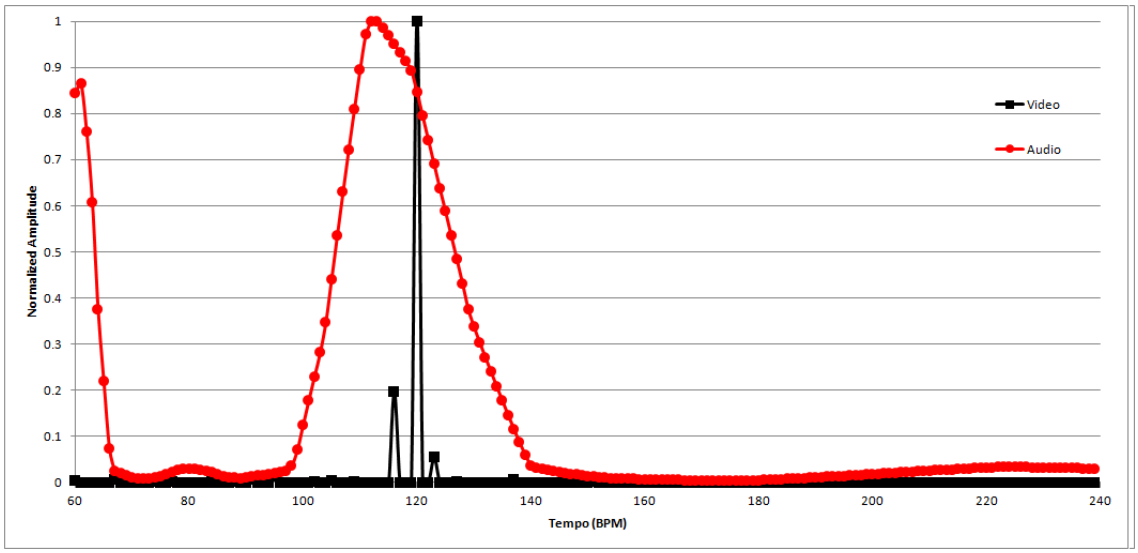


Figure 4.2: Readings from the Audio and Video Beat Detection

span of frequencies over which is searched is far less than the span of frequencies in the entire signal (i.e. the signals to not represent the entirety of the original signal), the SNR is not representative of the music as a whole, just the portion related to tempo. The signal input from the visual beat detection is calculated in a similar manner. For each point tracked, the signal to noise ratio is calculated in this way, and the point is thrown out if it is below a certain threshold. When the BPMs of the individual points are calculated and tallied, the resulting signal is only comprised of clean components and therefore a second SNR test is not needed. Instead, the value of the maximum peak is used to indicate the certainty in the visual tempo measurement.

If the audio tempo measurement passes the SNR test (has a Signal to Noise Ratio of atleast two), but there is uncertainty about the visual tempo (there are an insufficient number of points moving at the same tempo), only the audio prediction

is input into the learning algorithm. On the other hand, if there are sufficient points moving at the same frequency, but the audio signal is too noisy, only the visual tempo prediction is passed. The main fusion occurs only when both signals provide adequate predictions, as seen in Figure 4.2.

In this scenario, the difference in the predictions is checked. If they are within the resolution of the visual bet detection ($\sim$3.5 BPM), the reading determined by the audio is used. If they are outside that span, but are still sufficiently close (within 10 BPM), the two are averaged and passed that into the learning algorithm. If they are significantly different, the certainties of the predictions themselves, as well as the reliability as seen by previous measurements are analyzed. Murata [14] refers to these two reliabilities as the "Neighboring Beat Reliability" and the "Continuous Beat Reliability". The neighboring reliability can be realized by analyzing the SNR, while the continuous reliability can be realized by looking at the variance between the current measurement and previous measurements. This process can be summarized in Figure 4.3.

First, the SNR of the audio signal and the SNR of the visual signal are compared. Whichever has a higher SNR is said to have a higher neighboring beat reliability, and will be analyzed first for continuous reliability. In practice, the visual signal always has a higher SNR if it passes the initial certainty screening. To determine whether or not the prediction is consistent with the tempo of the music, a disparity check is performed between the prediction and the current integrated estimate of the tempo. If they are sufficiently close (within 10 BPM), this new prediction is entered into the learning algorithm. If not, this process is repeated for the

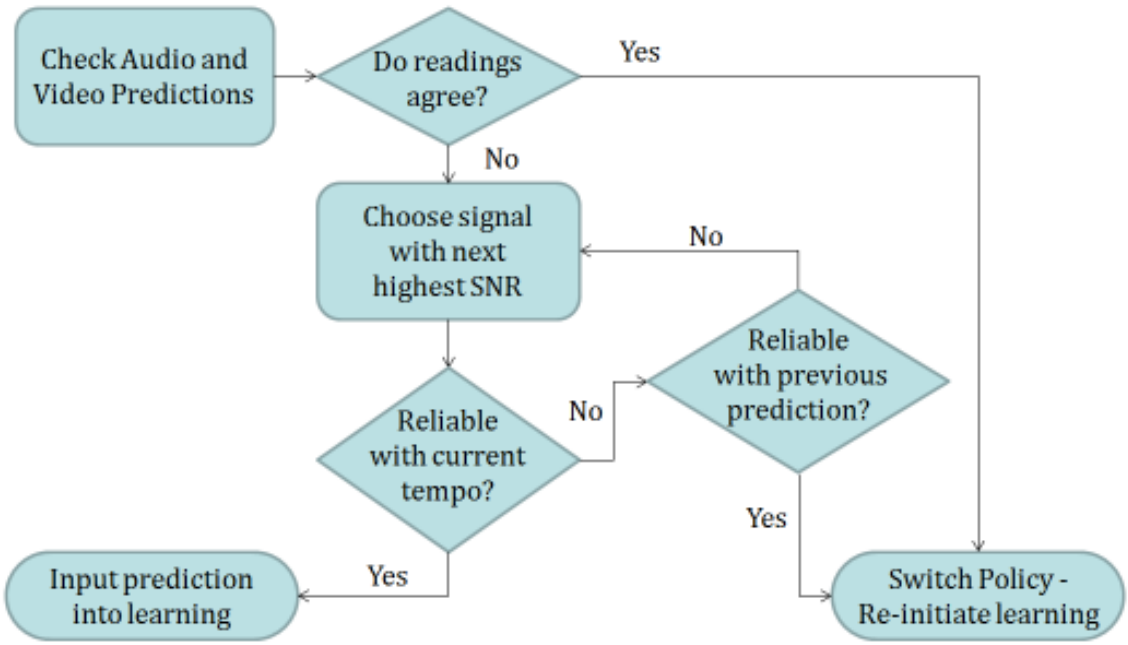


Figure 4.3: Flow Chart of Fusion Technique

prediction with the lower SNR. If there is again a large disparity, both signals are analyzed for consistency with the last prediction that the sensors made, regardless of the conclusions made about the previous readings. This sets the stage for our policy switching. If the prediction from one of the sensors is inconsistent with the integrated estimate but consistent with the last prediction made by either of the sensors ($\pm$ 5 BPM) then the new estimate becomes the average of the two readings. This switching condition was found by modelling the system as an Markov Decision Process, or MDP. According to the modelling, two successive measurements within $\pm$5 BPM is more than sufficient to satisfy the 99.3% certainty threshold required to switch policies.

Note: If both predictions fail the SNR tests, or are continuously unreliable/inconsistent, they are both treated as outliers, and the integrated estimate is not updated.

4.2.1 Modelling as MDP

In order to form a state switching policy, as seen in Figure 4.3, the system was modelled as a Markov Decision Process. This helps to determine an uncertainty threshold for a prediction and what actions may lead to a probability below that threshold. For instance, under most circumstances, the prediction from a sensor is passed into the normal TD learning algorithm. However, if certain events occur (as determined by the MDP model), we will re-initiate the learning process via policy switching. To form this model of the system, all possible states of the system must be known, as well as the transition probabilities between every state. Through extensive experimentation (Chapter 5) it was found that the audio algorithm provides the correct tempo roughly 70% of the time, while the visual detection is only slightly more accurate. Also, since each measurement is treated as independent from one time to the next, these results can also be used to form transition probabilities. If the system is treated as binary, i.e. the only two states are in the form of a correct prediction and a wrong prediction, one can form the simple model in Figure 4.4.

Here, a green arrow represents the transition from any state to the correct state, with P(correct)=70%, and the red arrow represents the transition to an incorrect state, with P(¬correct)=30%, or a 30% chance of incorrectly detecting the tempo. Notice how the probability of "self-transitioning" to the correct state (i.e. predicting the correct tempo if the previous prediction was correct) is the same as the probability of correctly detecting the tempo after incorrectly detecting the tempo. This is due to the independence of each sample.

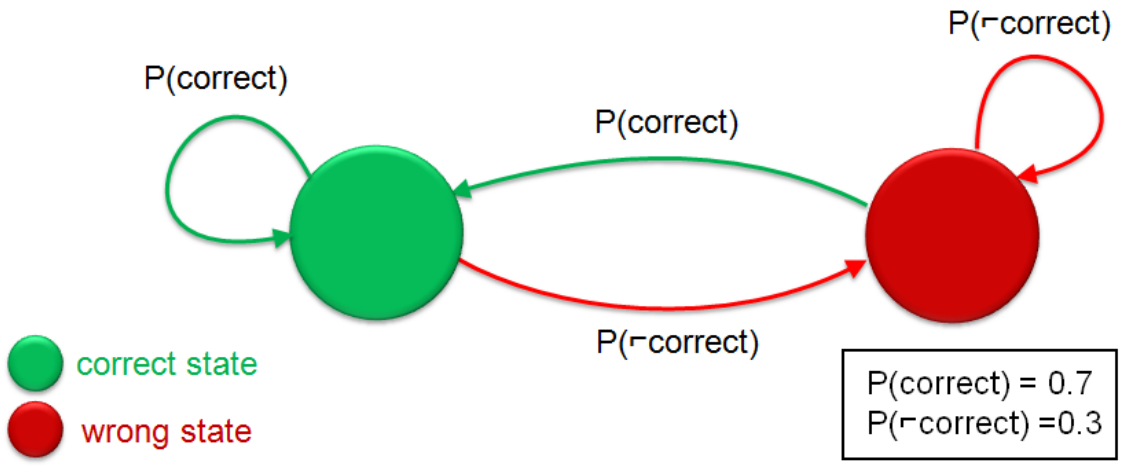


Figure 4.4: Binary Markov Model of the system

According to this model, there are only 2 states in which the robot can be (in complete synchronization with the music, and out of synchronization). The probability of getting two consecutive incorrect readings is calculated in the following manner. Here, S_1 and S_2 represent the first and second states of the robot. In this case, the robot incorrectly detecting the tempo of the music in consecutive measurements.

$$\begin{aligned} P(S_1 = \neg correct, S_2 = \neg correct) &= P(S_2 = S_1|S_1)P(S_1) \\ &= .3 * .3 \\ &= .09 \end{aligned} \tag{4.1}$$

The intention of this process is to find a series of events which would lead to a new estimate of tempo with high certainty. The probability of the algorithm giving an incorrect estimate twice in a row is .09, or 9%, meaning that our confidence of the correct tempo would decrease significantly. However, this does not provide informa-

tion about what the new estimate might be since it does not factor in the relative readings of the supposedly incorrect readings. In order to gain this information in the model, the model was switched from a binary (right/wrong) model to a binned model with many states.

In this model, there are 35 states corresponding to tempo bins of 5BPM, between 60-240BPM. The bin containing the true tempo is treated as the correct state and the 34 other bins are treated as separate, incorrect states. By differentiating the states by estimate as opposed to simply being right or wrong, we gain information on what the tempo will be if we choose to switch to a different state. The new model is seen in Figure 4.5.

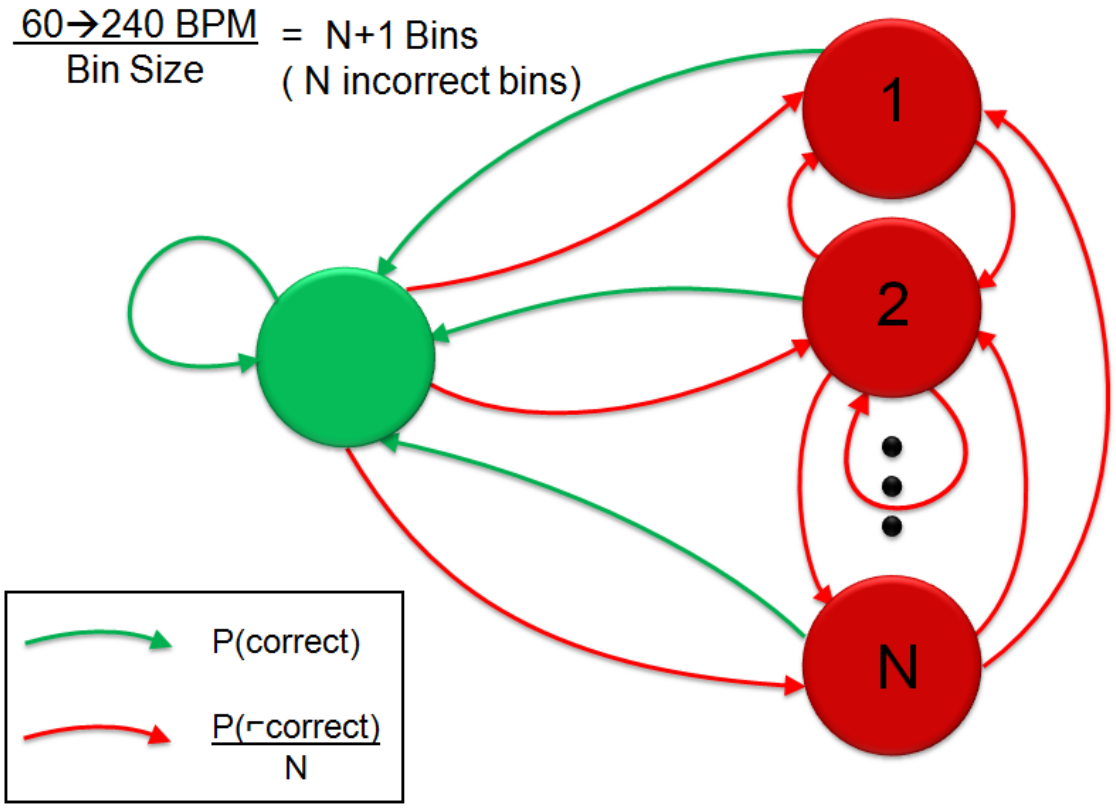


Figure 4.5: Binary Markov Model of the system

Preserving the nature of the system, the probability of getting two incorrect measurements in a row does not change:

$$\begin{aligned} P(S_1 = \neg correct, S_2 = \neg correct) &= \sum^{N} \left(\sum^{N} P(S_2 = S_1|S_1)P(S_1) \right) \\ &= 34 \left(34(\frac{.3}{34} * \frac{.3}{34}) \right) \\ &= .09 \end{aligned} \tag{4.2}$$

However, the probability of obtaining the same (approximately) wrong measurement twice in a row is now known. For example, suppose the algorithm is tracking the tempo of the music at 77 BPM (state 4) , and two measurements of 122 (state 13) are obtained twice in a row. Under the assumption that 77 BPM is the correct tempo, the probability of this occurring is:

$$\begin{aligned} P(S_1 = \neg correct, S_2 = \neg correct) &= \sum^{N} P(S_2 = S_1|S_1)P(S_1 = Bin13) \\ &= 34 \left(\frac{.3}{34} * \frac{.3}{34} \right) \\ &= 0.0026 \end{aligned} \tag{4.3}$$

The probability of obtaining the same wrong answer twice in a row is extremely small (.0026 or 0.26%), much less than the threshold defined for the switching condition of 0.7%. Instead, it would be safer to assume that the "wrong" measurement is actually the correct measurement, and that the song has changed tempo. Therefore, the switching condition is defined to be whenever the algorithm makes two consecutive measurements in the same bin. Under this policy switch, the temporal

difference learning is re-initialized with the new current prediction equal to the mean of those two measurements.

This policy switching technique can be analogized to a die with limited prior knowledge. Suppose we have a 20 sided die in which 19 of the sides contain the same number (either a 0 or a 1), and the 20th side contains the compliment (either a 0 or a 1, whichever is not on the other 19 sides). We do not, however, know which number is on the 19 sides. Suppose we initially assume that 19 sides have a 1. Without looking, we roll the die twice, and both times, our impartial friend tells us that we have rolled a 0. Instead of assuming that our initial assumption is correct and that we have rolled the 0 twice in a row (a 0.25% probability), it would be safer to assume that our initial assumption about the die was incorrect, and that 19 of the sides actually have a 0. Switching our assumption in this scenario is akin to our policy switching condition proposed in this thesis where we switch our assumption about the value of the mean tempo.

4.3 Temporal Difference Learning

The Temporal Difference learning portion of the code is closely integrated with the fusion algorithm. Referring back to Figure 4.3, once the new measurement from the fusion step is obtained, it is checked for consistency with the current prediction and previous readings. If the new readings are consistent with previous readings (and not the current prediction), the learning algorithm is re-initiated using the new prediction. If, however, the readings are consistent with the current prediction,

it is input into the learning algorithm in Equation 4.4, taken from [25].

$$\begin{aligned}\hat{V}(s_{t+1}) &= \hat{V}(s_t) + \alpha\left(r_t + \gamma V(s_t) - \hat{V}(s_t)\right) \\ &= (1-\alpha)\hat{V}(s_t) + \alpha(r_t + V(s_t)) \\ assuming\ \gamma = 1,\ r &= \begin{cases} True\ Tempo & at\ terminal\ state \\ 0 & otherwise \end{cases}\end{aligned} \tag{4.4}$$

Here, $\hat{V}$ signifies a predicted value, and s_t signifies the state of the system at time t. Therefore, using this notation, we can see that each time this equation is used, we obtain a new prediction of tempo for the next time step $t+1$. While $\hat{V}$ represents a predicted value (one that arose from Equation 4.4), V signifies a measured value from one of the sensors. Therefore, one will not see a $V(s_{t+1})$ in the TD algorithm, as $V(s_t)$ represents the most current reading from a sensor. . The discounting parameter,γ, is set equal to 1, which is a common selection in finite-horizon cases. Setting $\gamma = 1$ ensures that the final value of each state converges to the true tempo, regardless of the number of steps one takes to get there. The reward variable, r, is set equal to the true tempo at the terminal state, as we want all states to converge to this value. There is only 1 terminal state as there can only be one correct value for tempo. The step size parameter,α, greatly affects the rate of convergence.

With each new measurement, the prediction of the tempo is updated using $V(s_t)$ as the value of the measurement and $\hat{V}(s_t)$ as the current prediction. The prediction is then calculated using equation 4.4, which can be thought of as an

exponential moving average. In the proposed approach, $\alpha = 0.2$ is used, which leads to a fairly slow rate of convergence. While quickly converging systems ($\alpha \lessapprox 1$) may allow the robot to react quickly to changes in tempo, it also creates instability in the presence of noise. On the other hand, slowly converging systems($\alpha \gtrapprox 0$) leads to a very stable tracking tempo, but slowly reacting. However, with the proposed policy switching condition, this is negated when attempting to track large changes in tempo, and thus ideal for our algorithm. Whenever the switching condition is triggered, the Temporal Difference algorithm in 4.4 is re-initiated with $\hat{V}(s_{t+1}) = V(s_t)$

4.4 Discussion and Conclusions

The first step of the beat detection algorithm is to record a short video clip of the musician and audio of him playing. For this task, a Kinect camera (which consists of single RGB camera and IR depth Sensor) and a separate microphone are used. Although it is not currently done, fusion can be applied at this point (on the signal level) to decrease noise and time delay. First, there is an inherent lag between audio and video due to the differences between speed of light and speed of sound. Generally, the distance between the musician and sensors is quite small, however this may become an issue when distances increase, perhaps in use at a concert. Distance can be measured directly through the Kinect sensor, and the lag of the audio can be adjusted accordingly.

In the future, it can also be used to decrease the noise in the audio recording

by using a microphone array as opposed to a single mono microphone. Although the Kinect has 4 built-in microphones, they are currently unable to be used due to limitations in the third party drivers. When this feature is enabled, we plan on utilizing the on-board microphone array to decrease the noise floor and implement signal-level fusion.

It is also highly recommended that fusion is more seriously considered in the feature level. Each convolution of a comb with the audio signal produces a probability distribution of where beats are located in the signal at that tempo. It has been seen by looking at these signals that there are occasions when some beats are missed, and some when extra beats are added. These are referred to as deletion and insertion errors, respectively. These types of errors are mostly caused by deviation in human tempo, complicated beats, and effects of noise. While this may not always lead to an error in detection, most incorrect measurements are caused by these effects (see 5.2.2). At the same time, a peak (or trough) in the video signal also represents the probable location of a beat. By comparing the motion of the human body (which tends to emphasize each and every major beat) with the probability distribution of beat location,it may be possible to limit the number of these types of errors. Figure 4.6 shows an example of how this might work.

In Figure 4.6 (a), we see a music signal with a complex beat. After performing the beat detection algorithm, a probability distribution of where a beat might occur is formed (b). Notice how although the program obtained the correct tempo, there is a belief that a major beat may occur at $\sim$0.45s. Although this did not lead to an error in this example, it may cause mis-detection in other cases. When the

audio beat detection is overlayed on top of this distribution, it is clear which peaks represent true beats and which are spurious. In order to align and compare the two signals, one can use a normalized cross-correlation in these domains. Once the spurious beats are determined, we can go back to the original signal, remove the peak, and re-run the beat detection algorithm. We see the final result in (d), which is much clearer and reliable than using the original signal only.

One issue with our Markov Model is that it assumes all of the songs have a 70% chance of being correctly tracked, which is not necessarily the case. From experiments (Section 5.2.2), some songs are correctly tracked at every moment in time, while others are particularly difficult to detect the correct tempo. The probability that is song is correctly tracked changes from song to song, and therefore the switching condition should as well. In order to model this behaviour, a Hidden Markov Model can be used to determine beat pattern, beat modulation, vocal pattern, etc in order to calculate difficulty of tracking. However, this is a field of research in and of itself and would increase calculation time enormously. Another approach would be to allow the robot to accept feedback from the user in the form of either positive or negative rewards. The user can confirm when the robot is in synchronization with the musician and this reward can be input into the TD learning. This would greatly increase the accuracy or the robot, but in a sense, would destroy the mentality of musicianship and autonomy.

Despite the lack of dynamics in the modelling, the switching policy derived from this model was shown to be quite accurate. Of the 4600 samples calculated in Experiment 1 for the Popular Music database (100 songs x 46 samples per song),

incorrect switching only occurred 21 times, or 0.46%. Compared to the modelled 0.26%, this value still satisfies the 99.3% switching condition, validating the model as a sufficient estimate of our system.

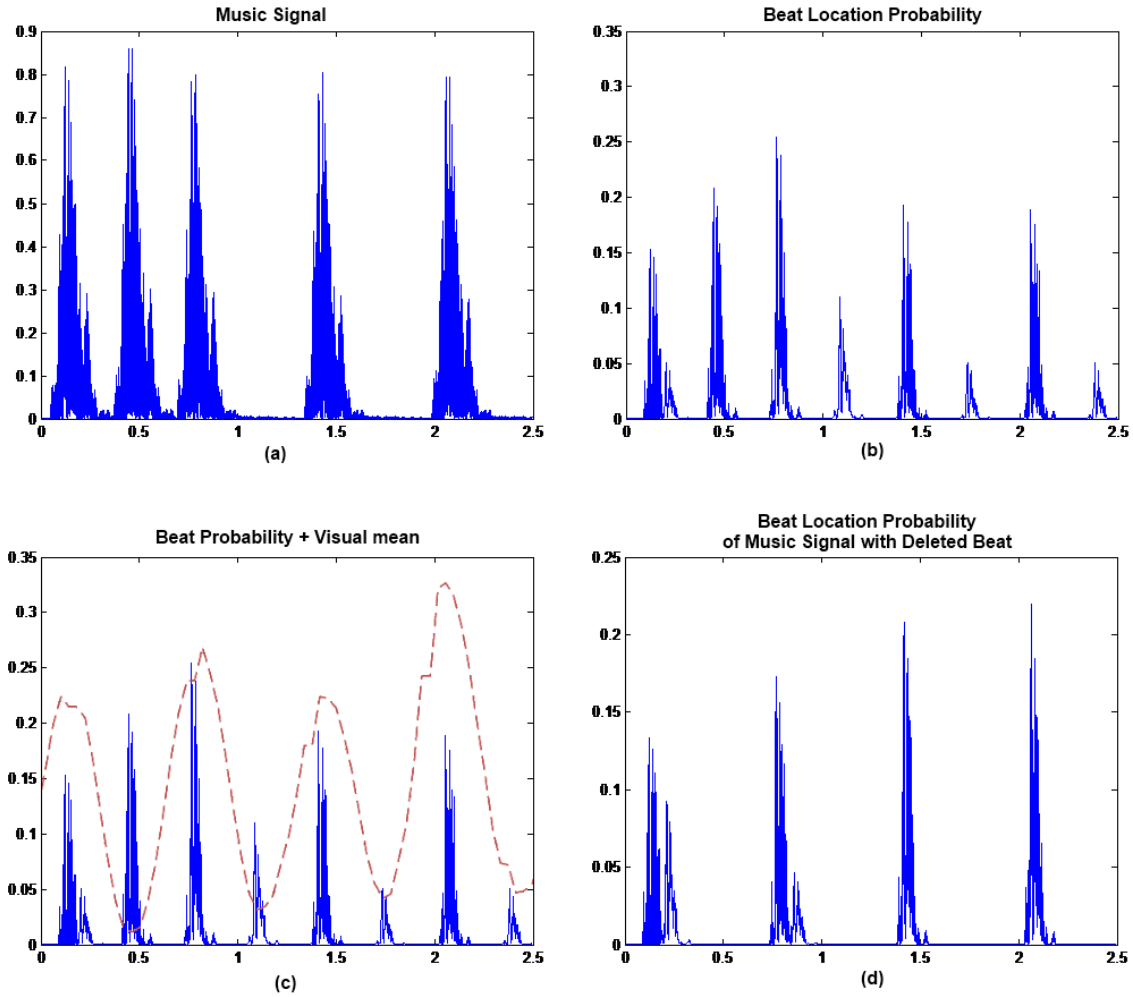


Figure 4.6: **(a)** The original music signal **(b)** The probability distribution of beat location in music signal (a) **(c)** The visual body tracking(dashed) overlayed on top of the audio probability distribution **(d)**The probability distribution of the modified music signal. With the second candidate beat detected as a false detection, we go back to the time domain and delete the peak. After the audio beat detection is performed on the new signal, the resultant distribution is seen here.

Chapter 5

Testing and Results

5.1 Overview

The field of robotic musicianship is fairly new, and being such, does not have the luxury of having well defined metrics or extensive databases which are universally tested. While Goto [30] has developed a set of metrics which have caught on [14, 5, 21], the statistics are not reliable without standardized testing procedures. Some songs/beats structures are more difficult to synchronize with than others, and unless every algorithm is tested against the same database, the quality of synchronization is subjective. The trouble in comparing algorithms is that there are multiple music databases [31, 32, 10, 11, 33, 34], and it is rare to find results against even one entire database. Furthermore, while some algorithms are tested against performance versions of the database, others are tested against the MIDI version of the database. This presents major difficulties since the MIDI versions do not have vocal tracks and have perfectly clean and well defined notes, and the tempo of the performance versions may fluctuate with respect to the performer. While one would expect that a professional musician would keep a perfectly steady beat, the inter-onset intervals have been seen to fluctuate with a standard deviation of as much as 10% [35]. In other words, human musicians can fluctuate ± 12 BPM at a tempo of 120 BPM, making it significantly harder for a robotic musician to anticipate beats.

Since the ultimate goal of robotic musicianship is to allow the live synchronization of a robotic musician with a human counterpart, with and without vocals present, it seems improper to test non-score-following tempo detection algorithm against MIDI files and/or metronomes. It was also decided to test against the entire RWC database of 100 Popular songs and 15 Royalty-Free songs (115 total), performed by human musicians. This was chosen due to the large size and availability. Since the database is only available through one channel, it can be ensured that our algorithm is being tested against the same exact version of the music as are other algorithms.

Despite the inconsistency of musical databases, any standardized database to test video algorithms is non-existent. In order to test the visual beat detection algorithm independently and compare its effects on the overall performance of the proposed algorithm, I have duplicated the tests performed by Itohara [21] with both a guitar and drum. Videos of Itohara's tests are not available, however for sake of standardization, copies of my tests are available upon request.

5.2 Testing

In the following sections, the testing procedures and results are discussed. Experiment 1 uses the RWC Popular Music and Royalty Free Database, to test only the audio portion of the algorithm. Experiment 2 uses simulated music signals to test the extent to which the proposed method works. By varying base tempo, standard deviation of tempo fluctuation, and added white Gaussian noise, the audio

algorithm is tested to find under what conditions it will fail. Experiment 3 uses video recordings of myself performing different beat patterns, ranging from easy to hard. This is used to test both the audio and video algorithms independently, as well as the integrated system. The section "Further Experimentation" analyzes the effect that the Temporal Difference learning has on the system, as well as the effect of window length with regards to speed and accuracy.

The metrics used to quantize the results are Recall Rate (R), Precision (P), F-measure (F), and Longest Continually Correctly Tracked Section (CL). The first three metrics are commonly used metrics in the field of Pattern Recognition, and proposed by Murata [14] to describe the "correctness", "completeness", and "effectiveness" of beat detection algorithms. The last metric was proposed by Klapuri [36], and is best used to describe the stability of the algorithm.

Recall Rate, by definition, is the ratio of relevant information received to the total amount of relevant information. In this case, the total amount of relevant information is related to the total amount of time music is being played. Since the actual tempo of the music being played is relatively constant throughout the entirety of the song, and data is only collected while music is playing, all information received should be relevant. Therefore, each time a section of music is recorded (2.5s intervals), marks an instant of relevant information. Only if the algorithm correctly determines the tempo of the music ($\pm$10 BPM, unless otherwise noted) during that section is it considered a correct retrieval. Therefore, Recall Rate is defined as:

$$R = C/A \tag{5.1}$$

where C is the amount of times the tempo is correctly determined, and A is the total amount of times a tempo should be detected. Recall Rate is best used to describe how often the robot should be playing in sync with the human.

On the other hand, Precision is very useful when determining the reliability of a measurement. For example, in our audio-visual integrated algorithm, it is very important to know which estimation of tempo (whether audio or video) is closer to the actual tempo. Although this is partly done by checking signal to noise ratios, all things being equal, it is best to know which approach, on average, produces a more reliable reading. Assume the case of an underwater robot which uses high resolution GPS based localization, but only comes to the surface for a reading once every few hours. While the robot would have a poor idea of where it was most of the time (poor recall rate), the few times the robot did receive a reading, it would be certain of its exact position. Therefore, Precision is the ratio of relevant information to retrieved information. In the experiments, each time a tempo is detected, it is said that a new piece of information is received, and if it is a correct estimate, then it is relevant. Therefore, Precision is defined as:

$$P = C/N \tag{5.2}$$

where C is the amount of times the tempo is correctly determined as defined above, and N is the total number of times an estimation is formed. In experiment 1 and 2, where only the audio algorithm was tested, the program was set such that an estimate would be formed at each instant in time. Therefore, N would be equal

to A, and subsequently, the Recall Rate would be equivalent to the Precision. For this reason, only the Recall Rate is reported for the first two experiments.

The F-measure is simply the harmonic mean of Recall and Precision, defined by:

$$F = 2 * \frac{P * R}{P + R} \tag{5.3}$$

The F-measure is a convention normally used in the information retrieval field to illustrate the effectiveness of a completed retrieval.

The final metric, Longest Continually Correctly Tracked Section, or Continuous Length, is also useful in describing the reliability and stability of the estimation. Suppose again the submarine example. If the GPS received a correct reading 50% of the time, it is important to the stability of the platform whether every other reading was correctly received, or if the first half of the readings were correctly received and then stopped functioning half way through. In the case of a robotic musician, this information is also very important because it indicates how long the robot can play in sync with the human before it is expected to fail. Is the algorithm sensitive to noise, or is it certain parts of music (such as bridges) where it fails? Most of these questions can be answered by looking at the CL. For standardization, CL is defined as a percentage rather than a length of time:

$$CL = L_C/A \tag{5.4}$$

where L_C is the length of the longest chain of correctly detected tempos, and

A is the total amount of times a tempo should be detected, same as above.

5.2.1 Experiment 1

The first experiment conducted was to test the how well the algorithm will perform in real-life scenarios. To do this, the code was run on all 100 songs of the RWC-P music database and all 15 songs of the MDP-R database. Along with the metrics discussed earlier, the estimated tempos for the entire song were also computed by performing some post-processing on the mode of the estimated tempos. The percentage of correctly determined songs is shown under the column SP, or Song Percentage. Below are the results for the entire databases, averaged across all the songs in the database:

Table 5.1: Overall Performance against RWC Database

	Recall	CL	SP
Popular Music	69.2391	59.8478	77
Royalty Free	78.0392	72.7451	86.6667
Weighted Avg	70.3869	61.5301	78.2609

In the graph below, one can easily see the times in which the algorithm correctly tracks the tempo, and the portions of the song where an erroneous tempo was detected.

Note that in the above example, it took a short while to lock on to the correct

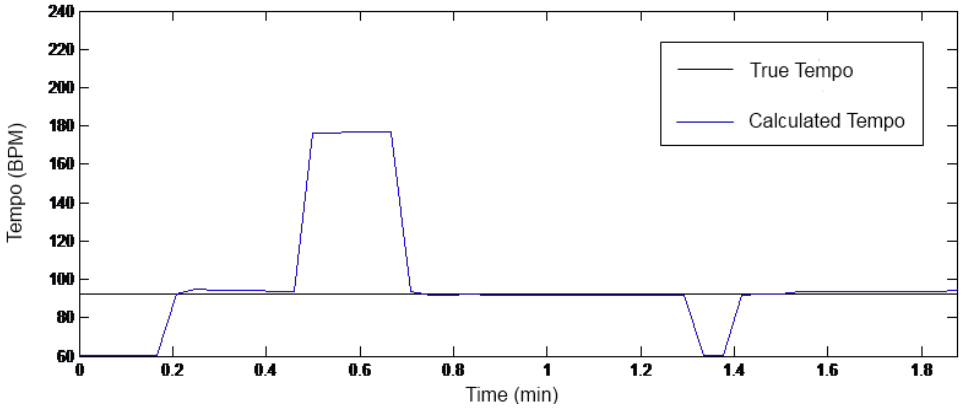


Figure 5.1: The Tempo Prediction output for the entire song ID 68, at 92 BPM

tempo, it stayed stable for a good while afterwards. Although it did get confused roughly a third of the way into the track by detecting double the actual tempo, it was able to correct itself not to long afterwards. This detection by half/double is referred to as an "allowable error" and in the analysis, is not considered to be an error. The quick switching, yet otherwise stable response is a perfect example of the Temporal Difference Learning with Policy Switching in action. It is also important to note that although the tempo detection provided an accurate tempo roughly 85% of the time, the Continuous Length, CL, is much less than that. This highlights the conservative nature of the CL metric.

To compare these results to the leading audio algorithm, the proposed algorithm was tested against the same 5 songs Murata tested his algorithm on [17]. In this paper, Murata publishes his results with respect to differences in beat times as opposed to overall tempo variation, as defined earlier. Instead, he defines a successful detection as the detection of a beat which is within $\pm.35I(t)$, where $I(t)$ is the

inter-onset interval at that moment in time, or the time between two adjacent beats. By definition, this is a much looser bound than the 10 BPM threshold, as previously reported. The following table shows that, on average our algorithm outperformed his for those 5 songs:

Table 5.2: Table showing a Comparison of Recall Rates

		Recall Rates	
ID	BPM	Murata	Proposed
4	86	81.2	71.7
11	90	72.1	100
17	97	81.6	100
18	112	83.2	100
29	103	82.2	58.7
Average		80.1	86.1

Although it is too small of a sample size to make any definite determination as to which algorithm is more accurate, the proposed algorithm performed similarly on the rest of the database (using Murata's error bounds). Against all 100 songs of the Popular Music Database, the algorithm correctly determined the tempo 85.4% of the time, with a standard deviation of 21.68%. This indicates that although the proposed audio algorithm is proficient at determining the correct tempo most of the time, there are some songs for which the prediction is highly unreliable. For

example, Murata's algorithm performed better on song ID 29 than did the proposed algorithm. To determine under what scenarios it will fail, an in depth analysis of the algorithm was performed under varying controlled conditions.

The results for Experiment 1 for each individual song (using the ±10 BPM error bound) can be found in the Appendix section.

5.2.2 Experiment 2

While testing against a database of songs is one approach to determining accuracy and robustness, it has its pitfalls. Each song is unique, with its own signature beat pattern, and an algorithm that may perform well on one song or type of song, may perform miserably on another. Therefore, it can test the variety of music on which an algorithm will perform well, but it does not determine what factors will lead to failure. In order to determine this for the proposed method, 124,440 simulations were run, systematically varying average tempo, noise, and tempo fluctuation (simulating human error). In the following results, the results have been collapsed over mean tempo. We are more interested in noise and fluctuation effects than average tempo, and having more tests over a wide range of tempos would be helpful to draw tighter conclusions from the analysis. As a note, mean tempo had almost no effect (atleast no predictable effect) on the data.

For the analysis, the algorithm was tested against simulated music signals. The simulated music signals are essentially binary click tracks, or metronomes, with clicks of 70ms. For example, a 2.5s, 60 BPM signal would have 3 segments with value of 1

(70ms in length), the onsets of which are separated by 1 second, and 0 everywhere else. The 70ms was chosen as the length of a normal musical beat by analyzing multiple musical signals in the database and looking at the average time a beat is clearly visible in its unprocessed form. To simulate human error, fluctuations in tempo were added by modulating the time between successive beats. The standard deviation of fluctuations in tempo ranged from 0-5.5% in 0.5% increments. This translates to a standard deviation of 0-5 BPM @ 90 BPM, quite consistent with even the worst musicians. While Mizumoto [35] used 10% to test his algorithm, this number is unrealistically high. Though our algorithm is tested against his at 10% standard deviation of fluctuation, to form the performance curve of the proposed method, values of this magnitude are not considered.

White Gaussian noise was also added to the signals with SNR ranging from 1E-20 to infinity. 1037 tests were performed over 61 different tempos for each combination of tempo fluctuation and noise to provide the following results. ANOVA was run on the results to find the distributions of the data with 99.3% probability.

The first part of this experiment, the algorithm was tested against fluctuations in tempo, with no added noise. Figure 5.2 shows that as tempo fluctuations increase, it becomes proportionally harder to track the tempo. While this effect is rather small at reasonable levels of fluctuation, it becomes significant at higher levels. The whiskers on the box plots represent 99.3% of the data, the boxes themselves represent 75% of the data, and the red line in the middle represents the median error.

This test also serves as a template for testing. Using an error threshold of $\pm$10BPM and assuming a perfectly clean signal, the algorithm will be correct 99.3%

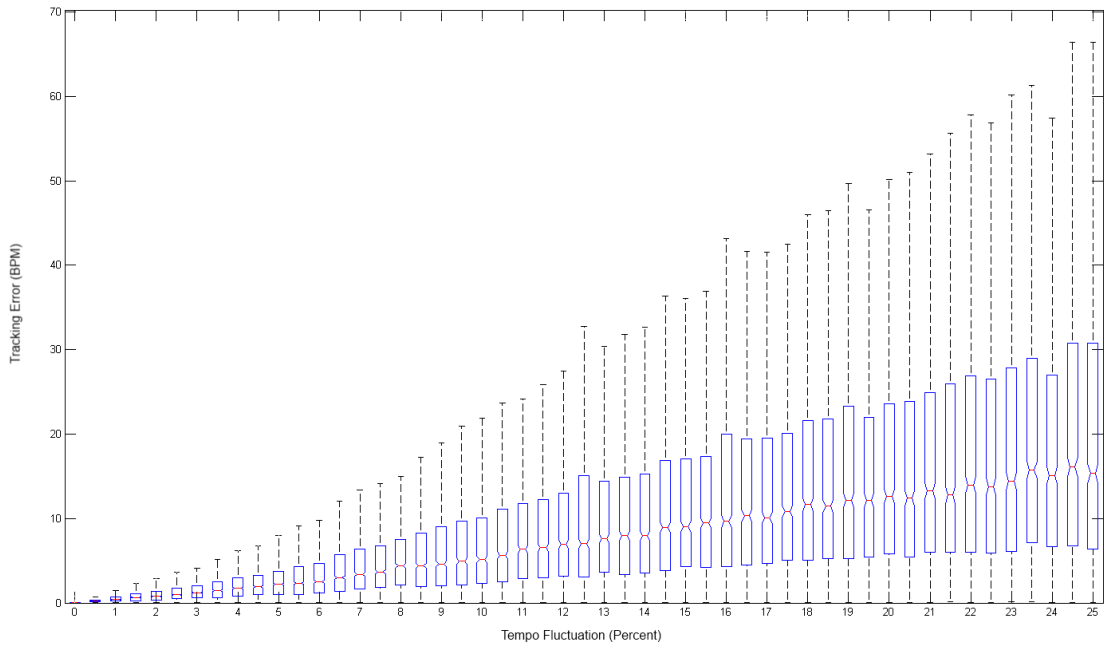


Figure 5.2: Proposed algorithm vs. Tempo Fluctuation. Outliers have been deleted for clearer viewing

of the time if fluctuations are kept below a standard deviation of 5.5%. For this reason, only fluctuations up to 5.5% were used when testing combinations of noise.

To test how the system performs under different levels of noise, the same experiment was performed, but this time, additive white Gaussian noise was introduced. Varying the signal to noise ratio, the following surfaces were calculated from performing over 100,000 tests with different combinations of noise and tempo fluctuation.

Figure 5.3 shows the mean performance over different error levels. Assuming perfect time, i.e. a metronome with no time fluctuation, the proposed algorithm can perfectly sync with the musical signal for reasonable to high noise values (greater than an SNR of 0.5). Only when noise reaches 0.375 or lower, where the signal is

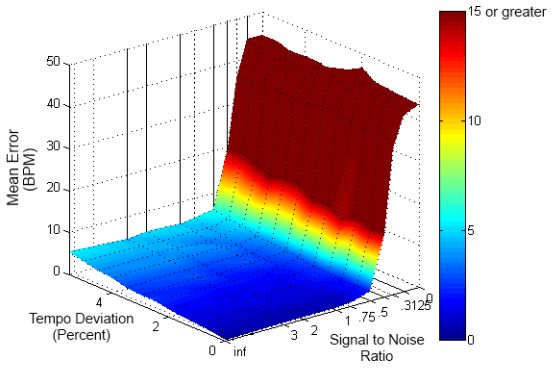


Figure 5.3: Mean Error

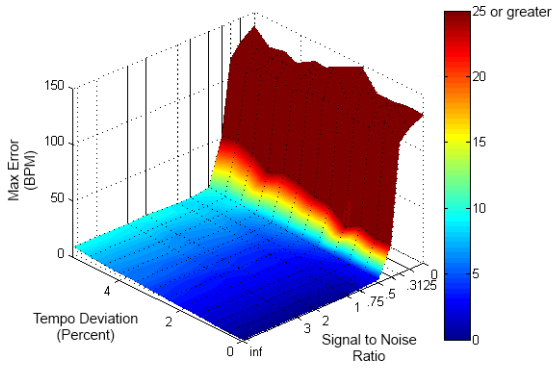


Figure 5.4: Max Error representing 99.3% of data

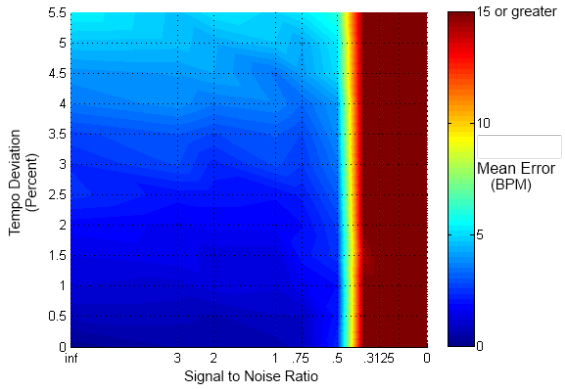


Figure 5.5: Mean Error (Top View)

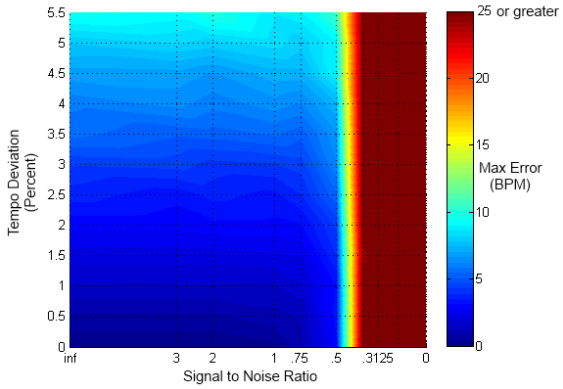


Figure 5.6: Max Error representing 99.3% of data (Top View)

exceedingly overpowered by noise, does the proposed approach begin to fail. The algorithm is slightly more effected by deviations in tempo. While there is no drastic change in performance at some critical value as seen in the noise from SNR 0.5-0.375, tracking error steadily grows with increase in tempo fluctuation. At a large deviation value of 5.5%, the mean tracking error is only ± 5 BPM, while the 99.3% upper bound is only 9.6 BPM, small enough to be reported as a correct measurement.

In other words, the algorithm can correctly determine the tempo of a metronome with 5.5% tempo fluctuation, 99.3% of the time.

Another important finding is regarding the interaction of these errors. For reasonable levels of both noise and fluctuation, the combination of both these errors add, not multiply. In other words, the total error in the system can be approximated as a linear combination of the two, as opposed to a coupled interaction which can intensify the errors exponentially. This translates to a much more robust algorithm, allowing better performance in more environments. The interaction effect was found using Matlab's surface fitting toolbox.

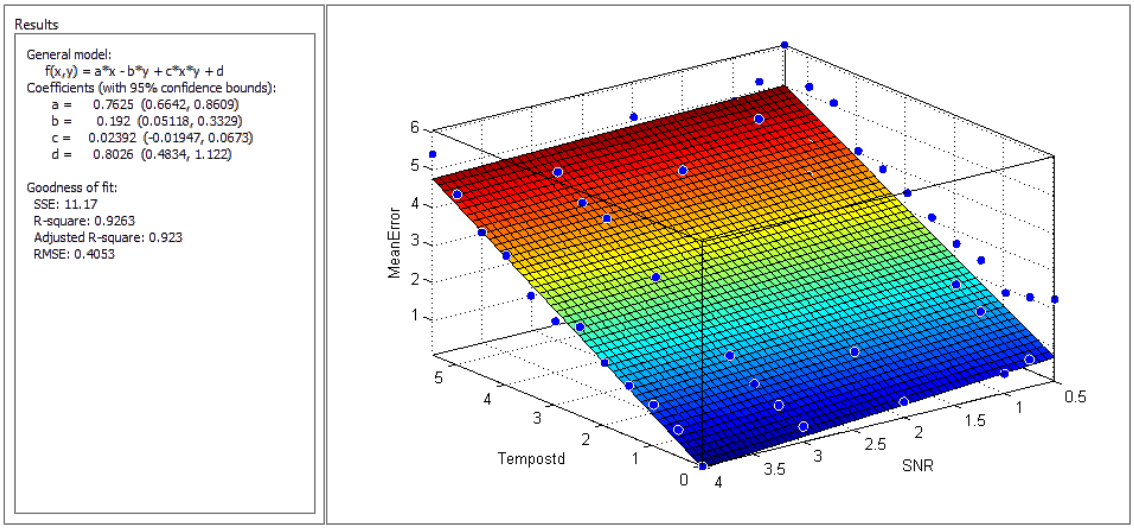


Figure 5.7: Surface fit of the cropped surface. All data represents mean error, the x-axis is fluctuations in tempo in percent, and the y-axis is the Signal to Noise Ratio.

Notice how the coupling variable is extremely small compared to the effects of each component. Furthermore, removing the coupling component altogether yielded a similar surface with the same R-squared value, hinting that the coupling effect is

negligible. The figure also shows that under normal circumstances, tempo deviation has a larger (yet still small) affect on total error than does noise.

The final part of this experiment was to compare our algorithm with Mizumoto's Coupled Oscillator (CO) and Spectro-Temporal Pattern Matching approaches [35]. First, the algorithm was tested against a metronome with perfect time and no noise, the results of which compared to those published by Mizumoto.

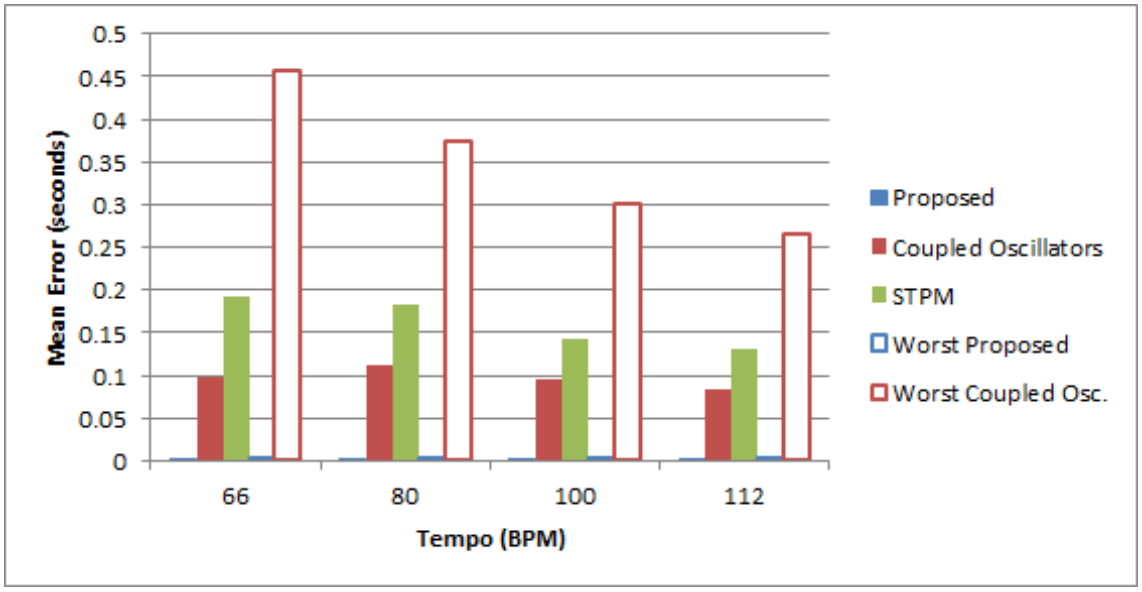


Figure 5.8: The mean errors against a perfect metronome for (left to right) the proposed algorithm, Mizumoto's coupled oscillators, and his STPM approach, along with the maximum error for the proposed algorithm, and the maximum of the coupled oscillator algorithm

Under a perfect time-keeping metronome, the proposed algorithm was in near perfect synchronization with the metronome every single time. This fairs much better than the coupled oscillator which needs time to synchronize, and even more-so than his STPM approach, whose error is most likely due to poor time resolution.

Tempo fluctuations with a standard deviation of 10%, much worse than an average musician, were then introduced. The proposed method had the smallest mean error, followed by the coupled oscillator approach, and then the STPM method. However, there was too much deviation in the data to draw any conclusions about the performance of the algorithms with such high input error. This result further proves the difficulty of the task in tracking tempos with large fluctuations.

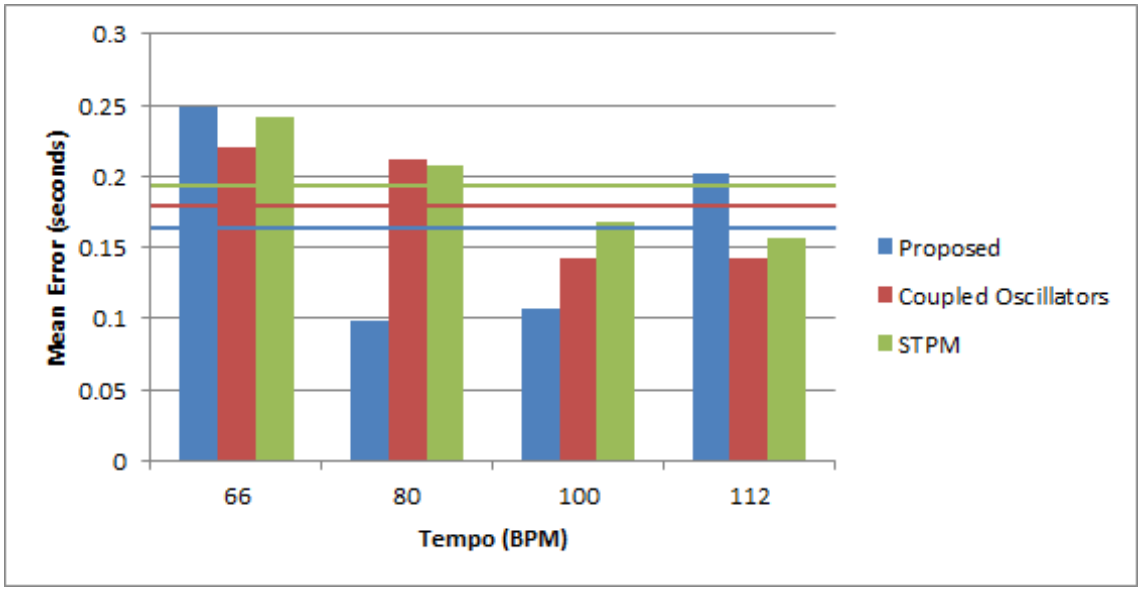


Figure 5.9: The mean errors against a highly fluctuating metronome for (left to right) the proposed algorithm, Mizumoto's coupled oscillators, and his STPM approach. The horizontal bars represent the average error across the 4 tempos

While Experiment 2 is meant to test the robustness of the algorithm in a systematic and objective way, it can only imitate the errors found in simply modeled simulated music, and does not represent all sources of error. This experiment tested the relative difficulty of synchronizing with music containing tempo fluctuation and

in the presence of noise. It does not, however, test the relative difficulty in synchronizing with complex beats, which is the most common source of error. Instead, we attempt to test this in experiment 3.

5.2.3 Experiment 3

For this experiment, extensive analysis on the performance of the integrated system is performed, as well as independent studies on the performance of the audio and visual algorithms. Since the field of Audio-Visual Beat Detection is still in its infancy stage, there are no set procedures on how to test the algorithm. In a recent publication by Itohara [21], subjects played 8 different strumming patterns on a guitar, and observed the response of their robotic musician. Since there are no available recordings of the studies, I have replicated the studies by performing the unique beat structures in front of a camera. While Itohara's visual approach only allows for the tracking of a guitarist, the full screen approach allows for the detection of practically any instrument. For this reason, we have replicated the beat structures while playing both the guitar and drums (separately, of course). Figure 5.10 shows the beat patterns used in this experiment.

To perform these experiments, I set up the microphone/camera (a Microsoft Kinect) on a tripod, and sat roughly 1.5 meters away. As per the guitar experiments, I played the pieces exactly as transcribed. To imitate the instructions for the drum, I struck a piece of hard rubber for a muted note (represented by an x), and a piece of wood for all other notes. For each run, I recorded myself for roughly 45 seconds,

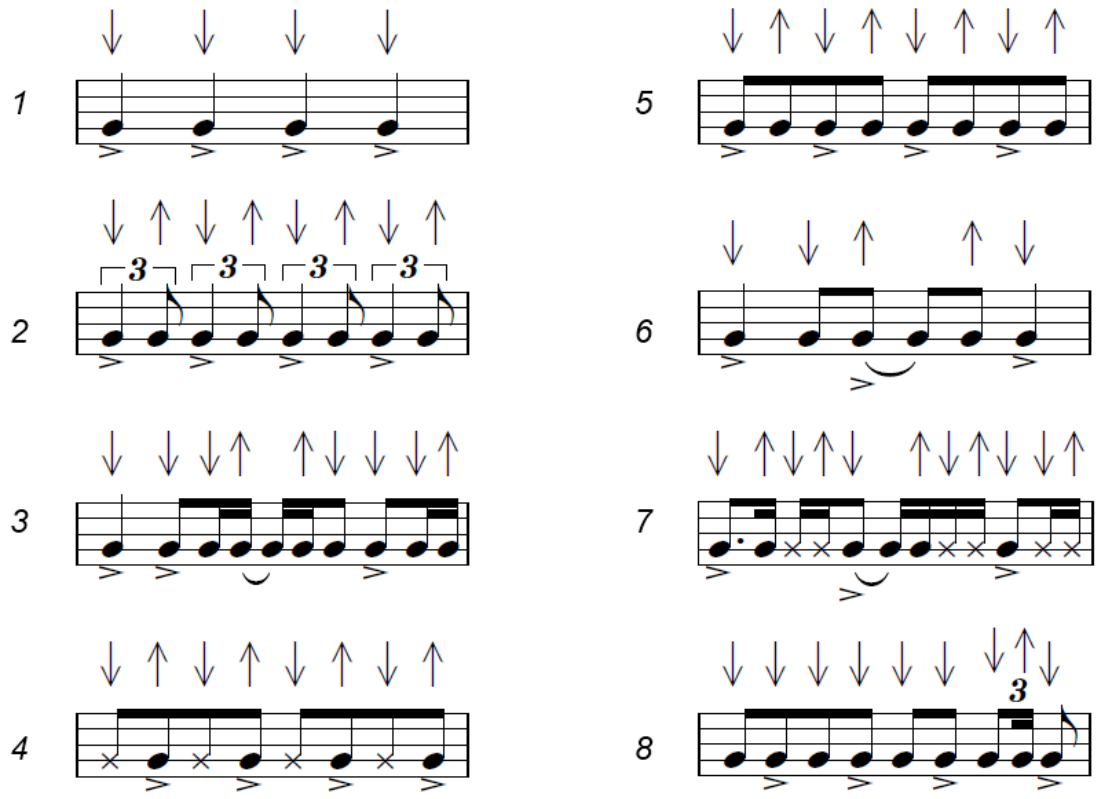


Figure 5.10: The 8 beat patterns used by Itohara to test visual beat detection algorithms

keeping the tempo as steady as possible, performing the beat detection live.

At each time step (every 2.5 seconds for audio and every 0.5 seconds for video), new predictions for tempo were made independently, and then used our sensor fusion/TD learning to produce the integrated estimate. Therefore, at each time step, 3 readings were recorded: audio, visual, and integrated. At the end of each run, each beat was manually marked in the recording to obtain the true tempo of the performances at each point in time. It is important to note that although the performance tempo normally fluctuated by less than ±10BPM, fluctuations in tempo sometimes exceeded ±20BPM depending on the difficulty of the beat pattern.

Shown in the following tables are the tabulated results for the performances, compared with that of Itohara and Murata's audio-only algorithm. Here, G represents the results from the beat patterns played on the guitar, while D represents the results from the beat patterns played on the drums.

Table 5.3: Table showing Precision Rates

		Precision								
		1	2	3	4	5	6	7	8	Average
Audio Only	G	100	100	40.0	100	18.8	63.2	100	100	88.9
	D	100	100	100	100	100	100	100	100	
Video Only	G	72.4	92.6	84.0	100	100	100	100	100	91.1
	D	97.7	100	37.7	100	100	100	73.9	100	
Integrated	G	97.2	96.9	80.0	100	95.1	100	100	100	93.8
	D	93.2	100	45.8	100	100	100	93.3	100	
Itohara		69.9	75.7	71.1	65.1	48.3	46.8	74.0	40.1	61.4
Murata		86.3	82.4	83.2	44.1	39.9	22.4	25.5	22.3	50.8

Table 5.5 shows that, on average, the proposed integrated algorithm outperformed both Itohara's and Murata's algorithms on these 8 patterns. With the exception of beat pattern 3, the proposed algorithm scored the highest F-score, or effectiveness rating, than the other previously proposed algorithm. A possible explanation as to why the third beat structure was particularly difficult to track is that the performance of that particular beat structure fluctuated the most. While the tempo fluctuated by less than ± 10 BPM in most of the cases, fluctuations in beat 3 exceeded ± 20 BPM in certain parts. It is very likely that the same beat performed by a better musician would yield a much higher accuracy. In the same vein, the difference between musicians in this testing versus Itohara's experiments may have

Table 5.4: Table showing Recall Rates for Exp. 2

		Recall								
		1	2	3	4	5	6	7	8	Average
Audio Only	G	88.2	88.9	33.3	78.3	16.7	60.0	85.7	87.8	78.0
	D	85.0	93.8	90.5	94.4	87.5	87.5	84.6	85.7	
Video Only	G	69.6	90.0	65.6	91.3	87.1	100	85.4	90.9	83.0
	D	87.8	89.3	34.1	93.8	89.6	95.2	66.7	90.9	
Integrated	G	88.6	90.0	62.5	91.3	82.9	98.0	85.4	90.9	85.6
	D	83.7	100	43.7	93.8	89.6	95.2	82.4	90.9	
Itohara		70.3	75.8	71.9	66.0	47.7	45.6	74.5	39.7	61.4
Murata		89.6	87.1	87.0	48.8	43.7	26.7	27.2	24.4	54.3

contributed to this difference in performances, as well. In our testing, the musician kept as steady a beat as possible; If the musician in Itohara's testing kept worse time, it would ultimately lead to lower accuracy estimates. Therefore, there is still some subjectivity in testing, and so one cannot make any definite conclusions about the performance of the approaches. However, the results from the experiments are better illustrated in Figure 5.11:

To test if there is a statistical difference between the performance of the algorithms, the datasets are analyzed using Matlab's built-in ANOVA, or Analysis of Variance, tool, as seen in Figure 5.12. The output shows that there is statistical difference between the proposed method and Ithohara's and the proposed method and Murata's. However, this conclusion is spurious as Matlab flagged the data from experiment 3 as an outlier, and did not take it into consideration when analyzing the variance. Had these data points not been considered outliers, there would clearly be no statistical difference between the approaches for this set of experiments. Further,

Table 5.5: Table showing F-Measures for Exp. 2

		F-Measure								
		1	2	3	4	5	6	7	8	Average
Audio Only	G	93.8	94.1	36.4	87.8	17.6	61.5	92.3	93.3	83.0
	D	91.9	96.8	95.0	97.1	93.3	93.3	91.7	92.3	
Video Only	G	71.0	91.3	73.7	95.4	93.1	100	92.1	95.2	86.8
	D	92.5	94.3	35.8	96.8	94.5	97.6	70.1	95.2	
Integrated	G	92.7	93.3	70.2	95.4	88.5	99.0	92.1	95.2	89.3
	D	88.2	100	44.7	96.8	94.5	95.2	87.5	95.2	
Itohara		70.1	75.7	71.5	65.5	48.0	46.1	74.2	39.9	61.4
Murata		87.9	84.7	85.1	46.3	41.7	24.3	26.3	23.3	52.5

as different musicians were used in the various experiments, it introduces uncertainty into the experiments. Still, the proposed method did, in fact, have a higher mean accuracy than does the other approaches as per our testing.

The results of these experiments are also great indications of the effectiveness of the sensor fusion. In Figure 5.13, it is quite easy to see the effect. While it is not calculated as such, the F-measure of the integrated approach is normally close to the maximum F-measure of the the audio-only and video-only approaches. This means that on average, the integrated approach is more effective than both the audio-only and video-only approaches. Since this also implies a lower standard deviation, it translates to a much more reliable algorithm than either of its parts. This is not only seen in the width of the error bars, but also the closeness in F-measure between trials.

Sensor Fusion can be thought of as decision making with imperfect information. Using the individual tempo predictions from the audio and visual beat detec-

Table 5.6: Table showing Continuous Lengths for Exp. 2

		Continuous Length								
		1	2	3	4	5	6	7	8	Average
Audio Only	G	88.2	88.9	33.3	78.3	16.7	40.0	85.7	87.5	76.7
	D	85.0	93.8	90.5	94.4	87.5	87.5	84.6	85.7	
Video Only	G	69.6	90.0	26.0	91.3	87.1	100	92.1	90.9	76.4
	D	37.8	89.3	11.9	93.8	89.6	95.2	66.7	90.9	
Integrated	G	74.7	72.9	17.7	91.3	42.9	98.0	85.4	90.9	72.6
	D	36.7	100	8.73	93.8	89.6	95.2	72.5	90.9	
Itohara		49.9	64.2	50.0	43.6	22.8	25.3	54.8	26.5	42.1
Murata		84.2	68.9	78.6	24.1	11.0	8.4	19.4	16.9	38.9

tion, The true tempo must be found. If one or both of the sensors always provided a correct tempo (i.e. had a Precision or Recall of 100%) this task would be simple. However, as shown previously, this is not the case. The audio algorithm is correct 78% of the time (combining the results from Experiment 1 and Experiment 3) while the video algorithm is correct roughly 83% of the time. Assuming this, one might expect that the fused sensor should correctly detect the tempo 96.26% of the time, as calculated below:

$$error = (1 - .78) * (1 - .83) = .0374, or P_i = 100 * (1 - e) = 96.26\% \qquad (5.5)$$

However, this assumes that at each point in time, one is certain as to which sensor(s) is providing the correct tempo, which is not the case. Instead, the fusion algorithm attempts to determine which is correct by analyzing signal to noise ratios and reliabilities with respect to previous readings. To determine how efficiently the

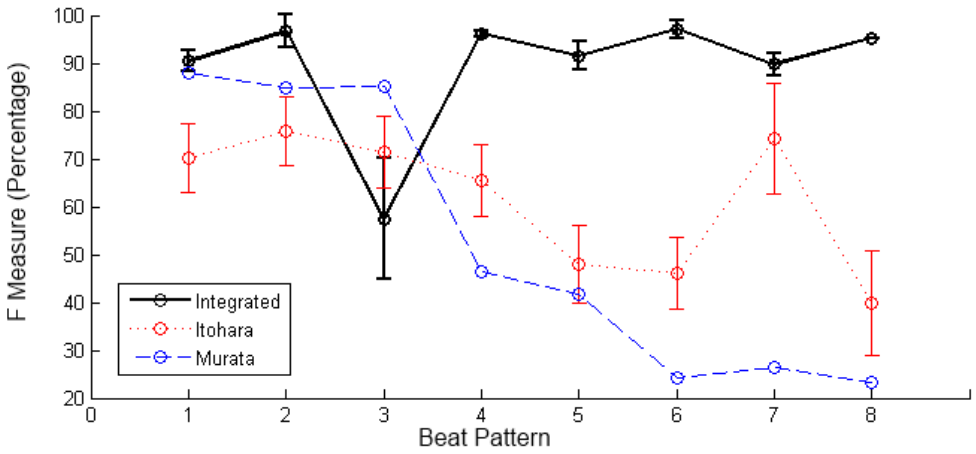


Figure 5.11: The F-measures of the Audio-only, Video-only, and Integrated algorithms.

fusion technique does this, the ratio between the experimental results and the ideal case is calculated, adjusting for worst case scenario.

$$E_{adjusted} = 100 * \frac{P_f - \bar{P}}{P_i - \bar{P}} = 100 * \frac{85.6 - \frac{1}{2}(78 + 83)}{96.26 - \frac{1}{2}(78 + 83)} = 32.36\% \qquad (5.6)$$

where: P_f = Probability of the Fused System

P_i = Probability of the Ideal System

$\bar{P}$ = Average Probability of Independent Components

Here, the $\bar{P}$ represents the probability of selecting the correct tempo by means of random selection between the two candidate tempos (audio and visual). While it is possible to score below 0% on the adjusted scale, it represents the efficiency with respect to the baseline fusion by random selection.

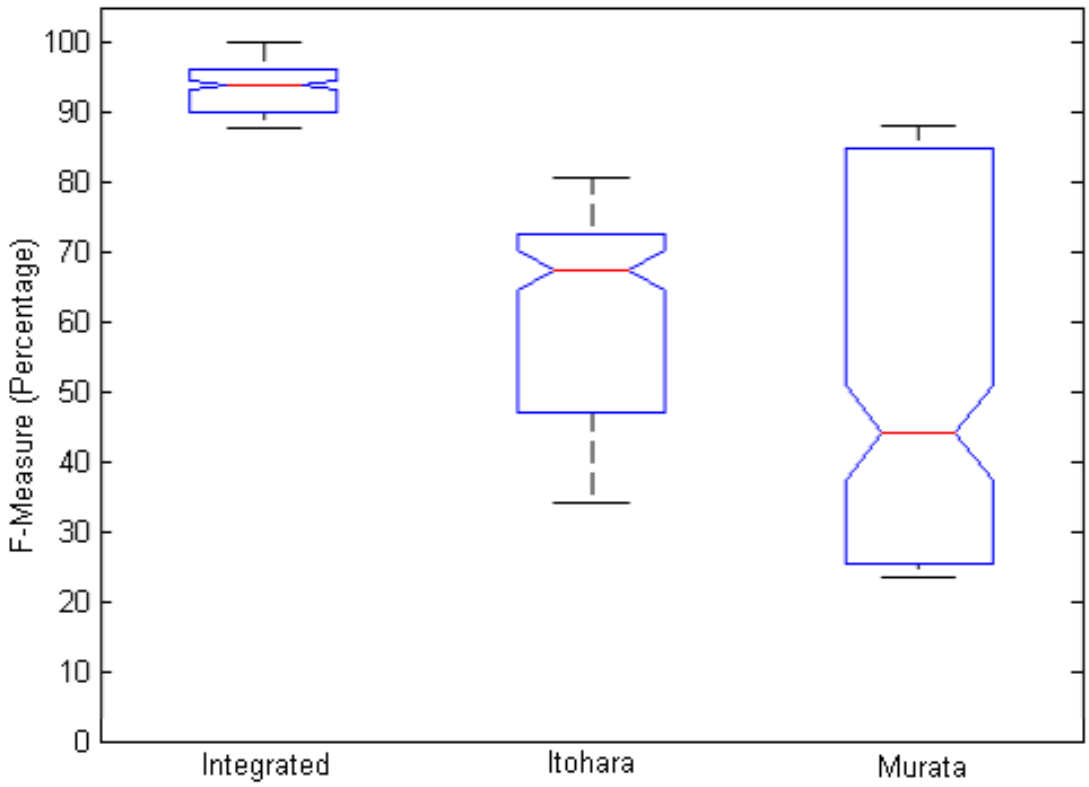


Figure 5.12: ANOVA result for Experiment 3

5.2.4 Further Experimentation

The first additional experiment that was conducted is to analyze the scalability of the approach. To test this, the audio algorithm was run on the entire RWC Popular Music Database and Royalty Free Music Database with window lengths of 2.5s (the default) and 5s. In the first case, the algorithm records for 2.5 seconds, analyzes the music, records for another 2.5 seconds, analyzes that signal, etc. In the second case, the analysis is done on every 5 seconds of data. What was found was that, as expected, the time it takes to analyze the data scales linearly with time. Instead of the algorithm taking 45ms to compute each prediction, the algorithm took 100ms, only 10ms longer than twice the baseline time. Although this would increase

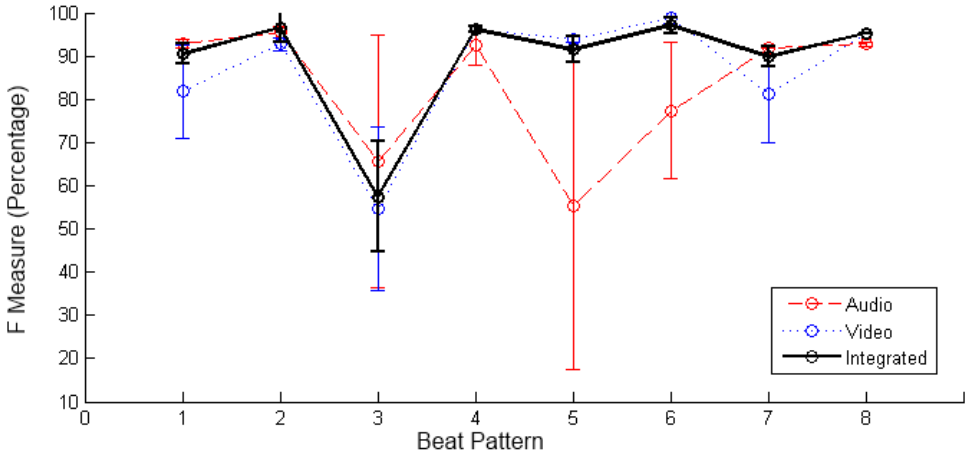


Figure 5.13: The F-measures of the Audio-only, Video-only, and Integrated algorithms.

reaction time, it would be feasible to increase the window time if it significantly increased the performance of the algorithm. As can be seen in Table 5.7, although the performance of the algorithm increased according to both Recall and Continuous Length, the effect is small.

Table 5.7: Performance against RWC Database w.r.t Window Length

	2.5s Window		5s Window	
	Recall	CL	Recall	CL
Popular Music	69.2391	59.8478	70.2174	66.7391
Royalty Free	78.0392	72.7451	74.1176	74.1176
Weighted Avg	70.3869	61.5301	70.7261	67.7015

It is also important to distinguish the difference between the effectiveness of

the raw tempo prediction algorithms, sensor fusion, and the proposed learning algorithm. The testing done in Experiment 3 showed the effect of our fusion technique, however it makes no mention on the effects of the temporal difference learning. Up until this point, all of the aforementioned data represents the system operating under our TD learning policy, and does not represent the precision or reliability of the raw tempo predictions. In other words, the Overall Amplitude Approach which has been proposed as the key to the proposed audio algorithm has not yet been analyzed without influence from the learning algorithm. To test the accuracy of the Overall Amplitude Approach for audio beat detection, the same tests as in Experiment 1 are performed, but this time, without applying reinforcement learning. Table 5.8 shows the results of these experiments.

Table 5.8: Effect of TD Learning against RWC Database

	Without TD Learning		With TD Learning	
	Recall	CL	Recall	CL
Popular Music	62.6739	25.8261	69.2391	59.8478
Royalty Free	71.9608	42.1569	78.0392	72.7451
Weighted Avg	63.8852	27.9562	70.3869	61.5301

There are two things to note about the experimental data shown in Table 5.8. First, is the moderate increase in Recall Rate between the two experiments. While the raw tempo prediction algorithm (the audio beat detection before TD learning

is applied) does perform admirably against the database, there was a noticeable increase in accuracy when the learning was introduced. This suggests that the Overall Amplitude Approach is a viable means for tempo prediction, and that TD Learning did have a positive effect on Recall.

As expected, the learning algorithm has its greatest effect on the stability of the signal. Our raw detection algorithm does provide a good estimation of tempo, but there will inevitably be errors and anomalies in the sensing. Reinforcement Learning, along with policy switching, provides a way of detecting those errors and smoothing over the noise. This is apparent in the drastic increase in the Average Longest Continually Correct Section, or CL. An example of one song is shown in Figure 5.14.

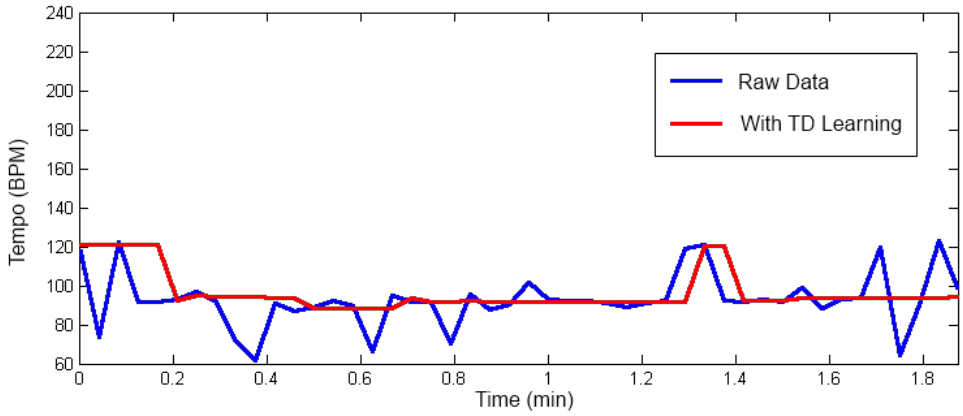


Figure 5.14: Comparing Continuous Length of Raw Prediction, and Prediction with TD Learning for song ID 68.

Notice how the system with TD learning does not erroneously switch tempos when a singular anomaly occurs. As Continuous Length can drastically change from one false reading in the middle of the song, it is important to detect anomalies and

smooth over similar readings. The resulting system will have a much more stable response and increases overall effectiveness.

5.3 Discussion

In this chapter, the experiments used to test the proposed algorithm have been discussed and the results have been thoroughly analyzed. Although there was found to be little statistical differences in performance between the proposed approach and other STPM approaches, the mean performance of the proposed algorithm is greater than that of the compared algorithms across all experiments. This holds true for all metrics (Precision, Recall, F-measure, and Continuous Length).

The proposed approach was then dissected in order to determine the effectiveness of each of its components (audio, video, fusion, and learning). The proposed single bin method for audio beat detection, or Overall Amplitude Approach, was analyzed; in its raw form, it can correctly determine the tempo music with a 63% accuracy. While in itself, is not the most accurate, it was designed to be the quickest. When Scheirer first proposed the Comb Filter Convolution method, he first broke the signal into multiple frequency sub-bands, then analyzed each band individually. His method was then expanded into the eventual Spectrogram, or Spectro-Temporal Pattern Matching, approach. This approach requires the use of STFT's to form a three dimensional Frequency-Time-Energy plot. While this step is computationally inexpensive, the image must then undergo two-dimensional convolution which is not. Furthermore, while this is being done, resolution in the time domain is being greatly

reduced. The Overall Amplitude Approach is an attempt at creating a simplified algorithm which is just as accurate as the more sophisticated approaches which could be computed in a fraction of the time. The idea is to use only one frequency band which captures the entirety of the musical content, eliminating the need to compute convolutions on multiple bands or a spectrogram. The resulting algorithm takes 45ms to compute and has a resolution of up to $\pm$.005-0.1 BPM depending on the tempo in question, meaning that it is a perfectly feasible approach for use in live beat detection applications.

While the 45ms computation time is less than a half of the $\sim$110ms it takes for the baseline algorithms, this can be decreased even more if resolution is sacrificed. Using a Matlab port, it was found that the computation time is perfectly linear with respect to the number of convolutions. Since the algorithm itself has an accuracy of $\pm$3 BPM, there is no need for such precise precision. At a uniform resolution of 1 BPM, an 18% increase in speed was seen using our pyramidal scheme. This resolution is perfectly acceptable considering that the spectrogram approach has an optimal resolution of 1.45 BPM @ 60 BPM to 20.4 BPM @ 240 BPM.

While the audio algorithm was designed for speed, there were no such concerns for the video algorithm. While beats in music correspond to impulses, i.e. the hitting of a drum), the human body is constrained by both velocities and accelerations. We can only move our limbs as quickly as our strength allows, and so this motion is inherently fluid. Therefore, are able to capture every motion using a standard 30FPS camera. At such low frequencies, the convolution of combs method would not offer an increase in resolution, and a standard Fast Fourier Transform would

suffice in converting the time-position data into the frequency domain. Since FFT's are computationally negligible, it is possible to compute the tempo of thousands of points without any concern for speed. This ability to track thousands of points simultaneously is what sets the Full Scene Motion Tracking approach apart from the hand-tracking approach of Itohara, the only other non-cue based visual beat detection algorithm. Instead of tracking the hand of a guitarist, tracking thousands of points on the human body and instrument allows for a more robust estimation of tempo and permits for the use of any instrument. Using the full scene analysis, one can not only track the guitarist's strumming hand, but also his opposite hand, the tapping of his foot, the nodding of his head, etc.

This approach is also extremely accurate, even more-so than the audio algorithm. The one downside to the video algorithm, however, is its lack of resolution. At best, the 30FPS/FFT based approach can only provide a resolution of 3.5 BPM. While the Kinect is useful to differentiate the musician from the background to decrease noise and necessary calculations, the maximum frame-rate of the camera is 30FPS. In order to get higher resolution, it is recommended that a faster camera is used and a different form of background cancellation is considered. However, since the time between frames is now decreased, the number of points being tracked should be limited to less than 1000.

Experiment 3 also shows that our sensor fusion was effective in increasing the overall performance of the system, as seen in the increase in Precision, Recall, and F-Measure. However, if there were one place for improvement in this algorithm, it would be at this step. In our approach, the majority of the fusion which affects

the accuracy of the algorithm is incorporated into the learning algorithm at the symbol level. This is all done after tempo detection from each sensor has taken place. Perhaps it would prove more effective to compare the audio-visual signal data at the feature level before the beat detection is performed.[28] A cross correlation between the raw audio and visual signals might work as feature extraction, providing insight into the location of the beats and which beats were falsely detected. This area should be studied in future research.

Our fusion technique is closely integrated with our reinforcement learning algorithm, but for means of analysis, the two have been separated for many of the experiments. While the proposed fusion approach only slightly increases the performance, there is a much larger boost when TD learning is applied. Integrating such with a policy switching condition not only provided an extremely accurate and stable prediction, but also a quickly reacting algorithm. One area for future research would be in the incorporation of different switching policies to decrease the amount of false switching and decrease the delay in tracking. Once again, this approach was chosen for speed considerations, taking only 5ms for the integrated fusion/learning step.

The code was written in C++, and run on a Dell OptiPlex 980 running an Intel i5@3.2GHz processor with 4GB RAM and Windows 7 64-bit. The audio and visual beat detection takes $\sim$45ms and $\sim$10ms respectively, with an additional $\sim$5ms to determine the integrated BPM. Therefore, the total calculation time is $\sim$50ms, less than half the time of the baseline, at $\sim$110ms.

For simpler analysis, accuracy of the audio and learning algorithms were per-

formed on a MATLAB port of the original code.

Chapter 6

Recreation of STPM and Comparison

6.1 Recreation of Code

The first step of any beat detection algorithm is to record the music which is to be analyzed. In Murata's STPM algorithm, the music is recorded for three seconds to ensure that at least three full beats were recorded during that time. For example, since the slowest tempo being calculated is 61 BPM (∼1 Hz), the signal must be at least ∼3s long to guarantee that three beats were observed, assuming perfect timing and that the first beat occurs before the 1 second mark. The signal is sampled at the normal audio rate of 44.1 kHz as to allow for the detection of every frequency that the human ear can hear (up to 20 kHz), as per the Nyquist Theorem. Although the signals of interest are within 1-4 Hz (60-240 BPM), sampling at such a high rate is necessary, since the length of the events (the beats) can be as short as 1ms. Sampling at a slower rate, on the order of ∼10 Hz, one risks missing the detection of an event.

The raw data is then converted into a spectrogram by a series of Short Time Fourier Transforms. The window used in this stage is a Hanning window of 4096 points, and a shift length of 512 points. This is the equivalent of saying that the signal is analyzed in chunks of ∼93 ms, and that each analysis only contains ∼11.6 ms of new information with respect to the previous window. This is obtained by

the following calculations:

$$\text{Lwindow} = \frac{4096}{44100\text{Hz}} \text{ x } 1000 \text{ ms/s} = \sim 92.88\text{ms}$$
$$\text{Lshift} = \frac{512}{44100\text{Hz}} \text{ x } 1000 \text{ ms/s} = \sim 11.6\text{ms}$$

This provides a very good representation of the signal in the spectro-temporal domain. The 93ms window is an appropriate choice as it provides sufficient resolution in the frequency axis while being short enough to capture transience in the time domain. The choice is also appropriate for efficiency considerations, as 4096 is a power of 2, aiding in quicker Fourier transforms. The 11.6 ms shift may seem overkill, as it is an overlap of 87.5%, however this provides extra resolution in the time axis which will be crucial to the resolution of the beat detection in the later stages.

In Murata's implementation, the next step is echo cancellation. Since the algorithm was designed for a physical robot, the program should not take into account the noise made by the robot itself; only the external music should be analyzed for tempo information. Since my implementation would be used solely to analyze clean, pre-recorded audio, this step has been skipped. Therefore, all the results presented are for the case with no echo cancellation.

The spectrogram is then converted into the Mel-scale. By the following formula:

$$m = 2595 log_{10}\left(1 + \frac{f}{700}\right) \tag{6.1}$$

The Mel scale is a perceptual scale of pitches based on psychoacoustic studies performed in the 1930's. It represents a human's ability to distinguish different pitches at lower frequencies, but our trouble in differentiating pitches at higher frequencies. The idea behind this transformation is to program the robot to interpret music as a human does. If a human ear is insensitive to pitch change at higher frequencies, and we are perfectly able to deduce tempo and beat structure, it would be a good idea to program the robot to hear as we do. Furthermore, since humans write the music for the entertainment of other humans, it is rare that complicated beat patterns would appear at the higher edge of our capacity to hear. In doing this transformation, very little information is lost at high frequencies, but better recreate the human perception of the music being played. In the process of this conversion, the 2049 frequency bins which result from the spectrogram are binned into 64 mel-scale bins for more efficient analysis.

The spectrogram is then analyzed for onset detection of notes being played. Murata utilized the Sobel filter for visual edge detection of the spectrogram image, however, in my implementation, I varied this slightly. In my implementation I still used a Sobel kernel, however, I convolved the signal with only the x-axis Sobel kernel, detecting only the onsets in the time axis. This seems logical as we only want to emphasize the onset time of the notes, and not the clarity of the frequency in the spectral axis. It is for this reason that only the positive values are kept, half wave rectifying the resulting image. This is to keep only those areas which represent a sharp increase in power (the onset), and disregard those which represent a decrease (the ending of a note). Before continuing to the pattern matching portion of the

algorithm, this resulting image is then normalized, and point-wise taken to the 1/2 power. This is to normalize the amplitudes such that the system is not overpowered by a pair of sharp rises. This pre-pattern-matching step is the vaguest portion in Murata's explanation of his algorithm, so most of the optimization has been done here.

The final step in the algorithm is the Spectro-Temporal Pattern Matching, which is nothing more than a normalized one-dimensional autocorrelation of a two-dimensional image. This process measures the similarity between the original data (spectrogram) and the spectrogram delayed by a certain length of time. Although this is a 2d series, it is only calculated for delays in time, not shifts in frequency. This is because we are interested in sinusoidal patterns over time, not similarities between frequency bins. The similarity between the two spectrograms is calculated in the following manner:

$$R(t,i) = \frac{\sum_{j=1}^{62} \sum_{k=0}^{P_{width}-1} d(t-k,j)d(t-i-k,j)}{\sqrt{\sum_{j=1}^{62} \sum_{k=0}^{P_{width}-1} d(t-k,j)^2 * \sum_{j=1}^{62} \sum_{k=0}^{P_{width}-1} d(t-i-k,j)^2}} \tag{6.2}$$

where P_{width} is the length of the window, characterized by the tempos being searched (60-240 BPM) and 62 represents the number of frequency bins being compared. In this formulation, at each time delay, the difference between the two images are calculated over all frequency bins as opposed to analyzing each bin independently. Furthermore, this calculation is only carried out for delays between .25 seconds and 1 second, corresponding to tempos of 240 and 60, respectively. Although Murata's algorithm only searches between 61-120 BPM, the test database

ranges in tempo from 60-240 BPM, so this was an obvious change.

6.2 Comparison with Published Results

As previously mentioned, the issue in comparing the Overall Amplitude algorithm to Murata's algorithm was the lack of published results on available test databases. There were, however, results for five select songs from the Real World Computing Popular Music Database [31]. Aside from the sparsity of results, the error margins used (on the order of 25%) are far too large to conclude anything about the accuracy of the algorithm. This can be seen by the conflicting results in other papers [21]. Below are the results provided by Murata, compared to the results from my implementation using Murata's error margins.

In the above results, one can see that our implementation performed just as well as Murata's algorithm against the selected songs, when no noise was present. While the results were surprisingly different, our implementation outperformed the baseline implementation on four of the six songs, leading to an average error extremely close to that of the original code. Therefore, with sufficient reliability, his method can be tested against the entire database of 115 songs using our implementation.

It is important to note that when noise was introduced, our implementation did not fair as well, leading us to believe that there is a noise cancellation stage in their algorithm which they have omitted from their publication. As a result, it would be unfair to compare the proposed algorithm with any noise-related tests (such as

Experiment 2 in Chapter 5), which were not obtained using their original code. The following experimentation is done only with perfectly clean, lossless versions of the songs in the Popular and Royalty Free Music Databases.

6.3 Extended Experimentation

When comparing different approaches or algorithms, in this case audio tempo detection algorithms, it's important to test each identically and objectively. That includes using a large database and the same metrics for each. In Experiment 1, the proposed algorithm was compared to Murata's using the 5 select songs for which results on accuracy were published. Then, in the previous section, Murata's original code was compared to our implementation of his code. Using the same database and the acceptable error conditions published by Murata, it was shown that the proposed method, and our implementation of his algorithm produce extremely similar results.

However, using such high allowable error artificially inflates the accuracy of the algorithm and does not bode well for comparison. As noted by Gouyon et al. [11], as allowable error increases, the difference in accuracy between algorithms decreases. Therefore, we tested both the proposed algorithm and our implementation of Murata's algorithm against the entire database of 115 songs with the ± 10 BPM accuracy defined earlier.

As one can see, the difference in accuracy (and stability) is amplified when a smaller error bar is used. It can be seen that the proposed Overall Amplitude Approach provided a much better estimate of tempo than did the Spectro-Temporal

Approach. Of course, this is not to say that Murata's original algorithm might not fair better than our implementation, however it should be noted that accuracy strongly depends on what is considered to be a "correct estimation". During the testing, a huge increase in accuracy and robustness was seen when performing beat detection on the original signal as opposed to expanding about the frequency axis as does STPM.

Table 6.1: Table comparing Murata's STPM vs. Our Implementation of STPM

	Murata's Algorithm			Our Implementation
	ASIMO Power Off		Power On	Power Off
Playing	off	on	on	off
ID 64	95%	68%	64%	100%

	Percent Correct			
ID	No Noise	with Noise	No Noise	with Noise
4	81.2%	84.9%	71.1%	100%
11	72.1%	67.3%	47.4%	50.0%
17	81.6%	83.1%	89.5%	18.4%
18	83.2%	82.8%	100%	86.8%
29	82.2%	81.5%	100%	68.4%
Average	80.1%	79.9%	81.6%	64.7%

Table 6.2: Table comparing Murata's STPM, Our Implementation of STPM, and the Proposed Algorithm against selected songs from the RWC-P Database

	Percent Correct (±0.35IOI error)		
ID	Murata	Our STPM	Proposed Algorithm
4	81.2%	71.1%	71.7%
11	72.1%	47.4%	100%
17	81.6%	89.5%	100%
18	83.2%	100%	100%
29	82.2%	100%	58.7%
Average	80.1%	81.6%	86.1%

Table 6.3: Table comparing Recall Rates using the Proposed Approach and the STPM Approach

	Percent Correct (±10BPM error)					
	Proposed Algorithm			STPM Approach		
	Recall	CL	SP	Recall	CL	SP
Popular Music	69.2%	59.8%	77%	28.9%	27.5%	33%
Royalty Free	78.0%	72.7%	86.7%	50.7%	48.6%	60%
Weighted Avg	70.4%	61.5%	78.3%	31.8%	30.3%	36.5%

Chapter 7

Discussion and Conclusion

In this thesis, a novel audio-visual beat detection algorithm, AVISARME, as well as a few new concepts, such as the Overall Amplitude Approach and higher order comb filters were proposed. By combining old approaches with new concepts, a new method of beat detection was created which is not only more robust than previously published methods, but is also more efficient.

Through extensive testing, the proposed audio algorithm was shown to be, on average, more accurate than the previously proposed approaches which were used in the comparison. Since the testing was mainly done on a single genre database and the visual algorithms were tested using different musicians, one cannot draw any conclusions with absolute certainty. However, what stands out in our audio algorithm is the extremely high resolution and the computational efficiency associated with it. Combining the audio algorithm with our novel gesture-less, multi-instrument visual beat detection algorithm, resulted in the development of an more reliable and robust beat detection algorithm. Introducing our TD learning with optimal policy switching, the resulting algorithm produces very stable, quickly reacting synchronization, all while taking only 50ms to compute on a 2.5s recording. It would be interesting to see how well the proposed video/fusion/learning algorithms would affect the performance of other popular audio algorithms.

It is important to note the robustness of both the proposed audio and video approaches. Since the audio is not split into multiple frequency bands, notes of every frequency are captured and treated equally. This allows the algorithm to detect even the most intricate beat patterns. Furthermore, there are currently no published algorithms for visual beat detection that can be applied to more than one instrument; the proposed approach can be applied to theoretically any instrument, without any instrument-specific alterations. We believe the future of visual beat detection is not in the locating and tracking of one specific point, but on the entire frame, hence Full Scene Motion Tracking. Given enough resolution and processing time, one would be able to capture every periodic movement of the musician and instrument which aids in determining the tempo at which the musician is playing. Recommendations for future research in this area have been supplied in Chapter 3.

The integrated fusion/learning algorithm also proved to be both efficient and effective, increasing the effectiveness by 10%, and the Continuous Length, or stability metric, by as much as 120% (though at times a slight decrease was seen). However, much of this boost in performance can be attributed to the learning algorithm. TD learning was chosen for its negligible computational cost, and implementing a novel switching condition based on statistical modelling provided us with a extremely stable, yet quickly reacting response. One area of future research is in the fusion aspect. As fusion is only applied on two levels, the proposed method is only operating with an adjusted efficiency of 32%. In the Fusion Section of Chapter 4, some recommendations have been supplied on how to apply fusion at the lower levels of abstraction.

It should also be noted that the comb filter technique for time-frequency conversion can be applied to more applications than merely beat detection. Although Fourier transforms are the most often used technique for this process, it lacked the resolution needed for our application. The Fourier Transform's resolution is based on both the sampling frequency and the length of the signal being converted, whereas the resolution of the comb filter convolution (CFC) technique is based solely on the sampling frequency, but far less dependant. Although there are Fourier transforms, such as the short-time Fourier transform, or STFT, which are designed for converting short signals, the proposed technique was found to be far superior to other methods. If the sampling frequency is sufficiently high, and if resolution is the primary concern, comb filter convolution should be seriously considered when performing spectral analysis on short signals.

Furthermore, thanks to the pyramidal optimization scheme, a greater resolution prediction is obtained while computing less convolutions during CFC. This means that our technique can be used to not only provide a more accurate representation of the tempo, but also allow the robotic musician to react quicker to changes in the music. Through our implementation, we have increased the resolution of our tempo detection algorithm by eight times, while decreasing the amount of convolutions by 30%. While the MATLAB port showed a speed increase of $\sim$17%, our C++ implementation showed no statistical increase in speed.

The cause of this discrepancy is the filtering process. While the amount of combs decreases, there is a low pass filter that is introduced to the system at each level of the pyramid. Although this is a relatively cheap process compared to the

time it takes to complete the convolutions for a full level, the number of levels in the pyramid should be kept to a minimum to limit the number of calculations.

Our 1D pyramidal optimization technique, like CFC, is not simply limited to beat detection. Like other direct-search optimization techniques, PO coarsely searches the entire domain of the function, and then populates the area surrounding local maximums to find the true global maximum. This is similar to genetic algorithms (GA) and/or simulated annealing (SA), in that it populates the area of the suspected global maxima, however it is deterministic in nature. While GA and SA are heuristic, relying on probability to span the entire search domain, PO relies on prior knowledge to find the global maxima. In this case, PO was told to search in predetermined locations (60 BPM to 240 BPM at 2 BPM intervals) for the suspected tempo, and to populate the area in search of the predominant beat. It was then told to repeat the process, terminating at a specific resolution. Therefore, PO can be used for any set with either a decent prior estimate of where the global maximum is, or with a small domain of which to search. PO would also be useful when multiple iterations are run and time/resolution between the iterations must be consistent. Otherwise, a heuristic search should be used to determine the search domain.

While implementing an STPM-based beat detection algorithm, there were many aspects of the approach worth mentioning. In the first stage, the choice of variables for the window size, shift, etc, all seem optimal. Though many different values for each were tried, all had negative effects on accuracy. The 93ms window for the spectrogram is much larger than the 30ms window which is quite standard

across the speech community, but it acts as a smoothing agent. Without it, another type of smoothing would be needed. A Gaussian window, as opposed to the Hanning window used in Murata's implementation, did seem to provide slightly better results. However the overall effect on accuracy was barely noticeable, while it significantly increased the time needed to calculate the spectrogram.

Secondly, the use of the Sobel filter to identify onset times in music, though uncommon, is controversial. When looking at the time domain of a music signal, it is easy to see that percussion and string instruments act as an impulse excitation, the response having an amplitude decaying over time, so differentiation is pointless for determining the onset of a beat.

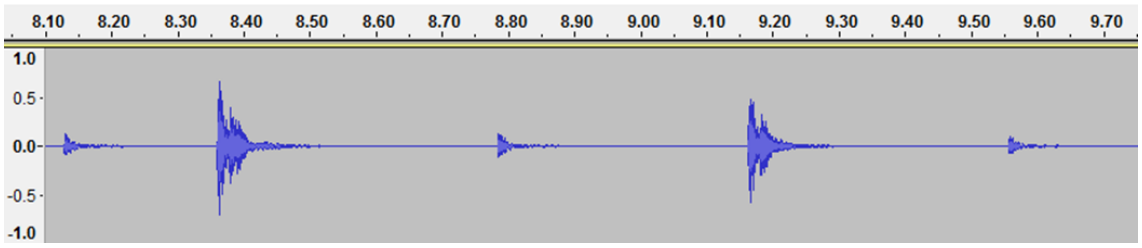


Figure 7.1: Amplitude-Time Response to a simple Drum Pattern

From Figure 7.1, it is obvious that the onset of the beat coincides with the maximum value of the amplitude. Differentiation would only emphasize the half-beats as well, introducing noise into the system. Though the differentiation did increase accuracy, it was quite minimal. It is possible that with further optimization, the direct method may prove to be the more accurate route. The non-differentiating approach more accurately represents the auditory signal, and therefore, it is our belief that a better estimation of tempo can ultimately be achieved from this method.

The approach might also be optimized by using a modified Normalized Cross Correlation algorithm. In the proposed algorithm, it was shown that a better approximation of tempo can be obtained by weighting higher amplitude matches greater than lower amplitude matches. For example, raising the numerator in the Normalized Cross Correlation algorithm to a higher order power.

The proposed approach to tempo detection also provides a more accurate representation of the tempo than does our implementation of the Spectro-Temporal Pattern Matching approach, yielding a correct tempo 72.9% of the time as compared to 31.8%. However, the main advantage that the CFC method has over STPM is in the resolution; the resolution of the time domain approach is also much better, having a cap of 0.02 BPM, while the resolution of the spectrogram approach is dependent on the window/overlap variables. Using the published variables, the resolution in the tempo domain ranges from 1.45 BPM @ 60 BPM to a horrendous 20.4 BPM @ 240 BPM. Furthermore, this advantage will never go away until processing power becomes so advanced that an STFT can be performed every 0.02ms. On the other hand, the time-domain approach also has its limitations. For example, it is specialized only to detect tempo and onset times, and cannot be used to deduce the underlying beat structure of the music.

For that, one must also determine the pitch of the music at each moment in time. One approach to this is to create a spectrogram. While the speed of this approach lends itself well to live analysis, it suffers greatly from a lack of resolution in the time domain. Another direction to look into is the wavelet transformation. Like the spectrogram, the wavelet transform converts the time-domain signal into

the spectro-temporal domain, though it does so with variable resolution, determined by the frequency bin. Since this is more similar to how humans perceive music, it might allow the robot to achieve a better estimate of the signal in the spectro-temporal domain, and therefore a better estimate of the feel intended by the artist. Although this would have some trade-off with respect to computational efficiency, this is a suggestion that should be considered very seriously when developing the next version of the Spectro-Temporal approach to beat detection.

A small adjustment that could be made to the proposed algorithm which has not been mentioned in previous sections is the use of rolling windows. The proposed algorithm performs beat detection on the audio and visual signals every 2.5s and 0.5s, respectively. While it only takes 45ms and 10ms respectively to compute the predictions, these restrictions were placed in the code as to not distort the signals when starting/stopping the recording process. By utilizing the multi-core nature of our processors more efficiently, it is possible to get around this distortion and obtain a new prediction every 50ms. This would allow for smoother transitions and quicker reaction time to changing tempo. While this is a small change, it would have a huge impact on the performance of the system.

Another area of future research would be to incorporate song identification into the tempo detection process. Most song identification algorithms compare the recorded signal to a large database of pre-recorded music, containing artist information, track name, lyrics, beat structure, and even average tempo. Using song identification, alone, for beat detection is unwise since human fluctuations are inevitable and it is common for a musician to change tempo to evoke certain

responses from the audience. Instead, it might serve well to periodically search the song database for the average tempo, and then perform beat detection over a small span of tempos around this mean. This would allow for more accurate, efficient, and reliable beat detection, but would require significantly more storage and/or internet connection. It would also further allow the robotic musician to play song-specific chord or beat patterns as opposed to simply tracking the tempo. The proposed algorithm is intended only for tempo detection, and without the incorporation of song identification, is limited in its application. If more information other than tempo is required, such as beat structure, instruments, and even lyrics, using a binned-frequency approach like STPM would be necessary. The STPM approach could therefore be used for any song, as opposed to the CFC method with Song ID, which can only be used with high accuracy and speed, but only for the songs in the database. Perhaps combining all three algorithms might prove to be the most useful and robust approach to date.

Appendix A

Experiment 1 Complete Results

ID	Recall	CL
P1	63.0435	58.6957
P2	100	100
P3	21.7391	10.8696
P4	21.7391	13.0435
P5	89.1304	58.6957
P6	84.7826	47.8261
P7	26.0870	26.0870
P8	82.6087	30.4348
P9	91.3043	91.3043
P10	82.6087	63.0435
P11	82.6087	65.2174
P12	78.2609	71.7391
P13	95.6522	73.9130
P14	100	100
P15	67.3913	21.7391
P16	95.6522	84.7826
P17	100	100
P18	100	100
P19	100	100
P20	95.6522	95.6522

ID	Recall	CL
P21	91.3043	65.2174
P22	100	100
P23	71.7391	47.8261
P24	100	100
P25	65.2174	65.2174
P26	100	100
P27	4.3478	4.3478
P28	13.0435	13.0435
P29	6.5217	6.5217
P30	86.9565	58.6957
P31	0	0
P32	84.7826	84.7826
P33	43.4783	30.4348
P34	56.5217	34.7826
P35	43.4783	39.1304
P36	6.5217	6.5217
P37	34.7826	19.5652
P38	54.3478	36.9565
P39	63.0435	63.0435
P40	100	100

ID	Recall	CL
P41	100	100
P42	100	100
P43	65.2174	28.2609
P44	84.7826	84.7826
P45	95.6522	89.1304
P46	100	100
P47	76.0870	36.9565
P48	86.9565	67.3913
P49	95.6522	80.4348
P50	63.0435	52.1739
P51	91.3043	47.8261
P52	95.6522	80.4348
P53	86.9565	86.9565
P54	84.7826	47.8261
P55	100	100
P56	95.6522	86.9565
P57	0	0
P58	21.7391	17.3913
P59	8.6957	8.6957
P60	100	100

ID	Recall	CL
P61	100	100
P62	100	100
P63	71.7391	50
P64	0	0
P65	84.7826	84.7826
P66	52.1739	26.0870
P67	100	100
P68	84.7826	58.6957
P69	19.5652	15.2174
P70	30.4348	15.2174
P71	100	100
P72	4.3478	4.3478
P73	73.9130	73.9130
P74	30.4348	30.4348
P75	71.7391	71.7391
P76	15.2174	10.8696
P77	6.5217	6.5217
P78	100	100
P79	65.2174	34.7826
P80	100	100

ID	Recall	CL
P81	0	0
P82	52.1739	32.6087
P83	100	100
P84	67.3913	36.9565
P85	100	100
P86	80.4348	80.4348
P87	60.8696	34.7826
P88	86.9565	86.9565
P89	100	100
P90	100	100
P91	100	100
P92	93.4783	93.4783
P93	17.3913	17.3913
P94	93.4783	93.4783
P95	21.7391	13.0435
P96	100	100
P97	78.2609	43.4783
P98	17.3913	10.8696
P99	93.4783	93.4783
P100	93.4783	60.8696

ID	Recall	CL
R1	70.5882	50
R2	85.2941	52.9412
R3	76.4706	50
R4	64.7059	64.7059
R5	91.1765	91.1765
R6	100	100
R7	32.3529	32.3529
R8	100	100
R9	70.5882	70.5882
R10	85.2941	85.2941
R11	100	100
R12	100	100
R13	11.7647	11.7647
R14	100	100
R15	82.3529	82.3529

Bibliography

[1] Hideki Kozima, Marek Michalowski, and Selma Sabanovic. A dancing robot for rhythmic social interaction. *Proceedings of the 16th IEEE International Conference on Robot & Human Interactive Communication*, 2007.

[2] Gil Weinberg, Scott Driscoll, and Mitchell Parry. Musical interactions with a perceptual robotic percussionist. *Proceedings of the 2005 IEEE International Workshop on Robots and Human Interactive Communication (ROMAN 2005)*, 2005.

[3] Takeshi Mizumoto. Integration of flutist gesture recognition and beat tracking for human-robot ensemble. *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[4] Takuma Otsuka, Kazumasa Murata, Kazuhiro Nakadai, Toru Takahashi, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G. Okuno. Incremental polyphonic audio to score alignment using beat tracking for singer robots. *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[5] Tatsuhiko Itohara, Takuma Otsuka, Takeshi Mizumoto, Tetsuya Ogata, and Hiroshi Okuno. Particle-filter based audio-visual beat-tracking for music robot ensemble with human guitarist. *Proceedings of the IEEE / RSJ International Conference on Intelligent Robots and Systems (IROS-2011)*, Sep. 2011.

[6] Angelica Lim, Takeshi Mizumoto, Louis-Kenzo Cahier, and Takuma Otsuka. Robot musical accompaniment: Integrating audio and visual cues for real-time synchronization with a human flutist. *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[7] Dirk-Jan Povel and Peter Essens. Perception of temporal patterns. *Music Perception*, 1985.

[8] Masataka Goto, Yoichi Muraoka, David Rosenthal, and Hiroshi Okuno. Music understanding at the beat level: Real-time beat tracking for audio signals. *IJCAI-95 Workshop on Computational Auditory Scene Analysis*, 1995.

[9] Eric D. Scheirer, Machine Listening Group, and MIT Media Laboratory. Tempo and beat analysis of acoustic musical signals. *Acoustical Society of America*, 1998.

[10] A.P. Klapuri, A.J. Eronen, and J.T. Astola. Analysis of the meter of acoustic musical signals. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(1):342 – 355, jan. 2006.

[11] F. Gouyon, Anssi Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Transactions on Speech and Audio Processing*, 14, 2006.

[12] Guy Hoffman and Gil Weinberg. Gesture-based human-robot jazz improvisation. *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, 2010.

[13] Peter Desain and Henkjan Honing. Computational models of beat induction: The rule-based approach. *Journal of New Music Research*, 1999.

[14] Kazumasa Murata, Kazumasa Nakadai, Ryu Takeda, Hiroshi G. Okuno, Toyotaka Torii, Yuji Hasegawa, and Hiroshi Tsujino. A beat-tracking robot for human-robot interaction and its evaluation. In *Humanoids*, pages 79–84, 2008.

[15] Aisha Hitz. Synchronization of movements of a real humanoid robot with music. 2008.

[16] Jean-Yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker. 1999.

[17] Kazumasa Murata, Kazuhiro Nakadai, Kazuyoshi Yoshii, Ryu Takeda, Toyotaka Torii, Hiroshi G. Okuno, Yuji Hasegawa, and Hiroshi Tsujino. A robot singer with music recognition based on real-time beat tracking. In Juan Pablo Bello, Elaine Chew, and Douglas Turnbull, editors, *ISMIR*, pages 199–204, 2008.

[18] Eric W. Wasstein. Parseval's theorem.

[19] OpenCV, 2011.

[20] Ashish Gourav. Exploring the hedonism of life: Continuity and discontinuity, October 2011.

[21] Tatsuhiko Itohara, Takuma Otsuka, Takeshi Mizumoto, Angelica Lim, Tetsuya Ogata, and Hiroshi G. Okuno. A multimodal tempo and beat-tracking system based on audiovisual information from live guitar performances. *EURASIP J. Audio, Speech and Music Processing*, 2012:6, 2012.

[22] OpenNI, 2011.

[23] Haruto Takeda, Takuya Nishimoto, and Shigeki Sagayama. Rhythm and tempo recognition of music performance from a probabilistic approach. In *in Proc. of the 5th Ann. Int. Symp. on Music Info. Retrieval*, 2004.

[24] Takuma Otsuka, Kazuhiro Nakadai, Toru Takahashi, Tetsuya Ogata, and Hiroshi G. Okuno. Real-time audio-to-score alignment using particle filter for coplayer music robots. *EURASIP J. Adv. Signal Process*, 2011:2:1–2:13, January 2011.

[25] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2009.

[26] Gheorghe Comanici and Doina Precup. Optimal policy switching algorithms for reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1*, AAMAS '10, pages 709–714, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems.

[27] R. C. Luo and M. G. Kay. A tutorial on multisensor integration and fusion . *Proc. 16th Annu. Conf. IEEE Ind. Electron*, 1:707722, 1990.

[28] R.C. Luo, Chih-Chen Yih, and Kuo Lan Su. Multisensor fusion and integration: approaches, applications, and future research directions. *Sensors Journal, IEEE*, 2(2):107 –119, apr 2002.

[29] T. Nicosevici, R. Garcia, M. Carreras, and M. Villanueva. A review of sensor fusion techniques for underwater vehicle navigation. In *OCEANS '04. MTTS/IEEE TECHNO-OCEAN '04*, volume 3, pages 1600 – 1605 Vol.3, nov. 2004.

[30] Masataka Goto and Yoichi Muraoka. Issues in evaluating beat tracking systems, 1997.

[31] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. Rwc music database: Popular, classical, and jazz music databases. In *In Proc. 3rd International Conference on Music Information Retrieval*, pages 287–288, 2002.

[32] Masataka Goto and Takuichi Nishimura. Rwc music database: Music genre database and musical instrument sound database. In *in ISMIR*, pages 229–230, 2003.

[33] Klaus Seyerlehner, Gerhard Widmer, and Tim Pohle. Fusing block-level features for music similarity estimation. *Proc. of the 13th Int. Conference on Digital Audio Effects*, September 2010.

[34] Helge Homburg, Ingo Mierswa, Blent Mller, Katharina Morik, and Michael Wurst. A benchmark dataset for audio classification and clustering, 2005.

[35] Takeshi Mizumoto, Takuma Otsuka, Kazuhiro Nakadai, Toru Takahashi, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G. Okuno. Human-robot ensemble between robot thereminist and human percussionist using coupled oscillator model. *The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[36] Anssi P. Klapuri. Musical meter estimation and music transcription. In *In Proc. Cambridge Music Processing Colloquium*, pages 40–45, 2003.