Robotically Reproducing Turntable Techniques

by

Andrew Wong

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Bachelor of Science in Computer Science and Engineering

and

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2002
[June 2002]

Author ....
Department of Electrical Engineering and Computer Science
May 20, 2002

Certified by ....
Chris Csikszentmihalyi
Associate Professor, MIT Media Lab
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Robotically Reproducing Turntable Techniques

by

Andrew Wong

Submitted to the Department of Electrical Engineering and Computer Science
on May 20, 2002, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science in Computer Science and Engineering
and
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The goal of this work is to simulate the techniques of a scratch DJ (turntablist) by designing and building a mechanism to manipulate a record and by creating software to intelligently command the mechanism. The long term goal of the project is to create a completely autonomous system that will be able to compose and perform scratch routines comparable to those of the best human DJ's. The metric of success for the work performed for this thesis will be its ability to simulate human attributes.

This thesis will discuss the construction of a mechanical system that will manipulate records in ways similar to those of a human scratch DJ, the integration of hardware and software to accurately control the mechanical system, and the design and implementation of software to record the motions of a human scratching a record, to play a tune on a record of a constant note by varying the playback speed, to pick beats out of a piece of previously recorded music, and to integrate these into a performance interface.

Many scratch DJ's use volume controls to enhance their musical compositions. This technique is not in the scope of this project. Additionally, DJ's switch records during the course of a performance, but this project will not include such functionality. Furthermore, the mechanical component of the project is built around the industry standard Technics SL-1200 M3D model turntable. Other turntable models are not supported by this project.

Thesis Supervisor: Chris Csikszentmihalyi
Title: Associate Professor, MIT Media Lab

Contents

List of Figures

Chapter 1

Introduction

The ITF's (International Turntablist Federation) definition of turntablism is as follows:
"Turntablist: One who uses the phonograph turntable as a component to make music as well as an instrument to literally play music." Turntablism is a fundamentally different art than DJing in that turntablists create new music with the turntables by scratching records, whereas DJ's play other people's music. Unfortunately, the terms are often interchanged and so the meaning of the term DJ is often ambiguous. The Disco Mix Club annually holds regional, national and world level competitions in which turntablists and DJ's perform six minute routines showing off their technical skills and their creativity. The competitions are judged by DMC officials and the winners from each level of competition proceed to the next level.

The final goal of this project is to build a robotic turntablist, DJ I, Robot, that is capable of competing on the national level against humans. The concept is much like that of Deep Blue, the chess playing computer program, in that it will be technically more precise and more thorough than any human could ever be. However, unlike chess, the art of turntablism is more subjective and there is no well defined metric for ability. Thus, the robotic turntablist will have to be able to adapt to the changing art. Additionally, the technical and creative aspects of turntablism are often very distinct and can be separated. In other words, it would be quite plausible to have DJ I, Robot perform scratches that are selected and arranged (a skill known as programming) by a human.

The scope of this thesis is the construction and technical programming aspect of DJ I, Robot. It includes the building of the mechanical system to sense and control the position of the record. It also includes the design and implementation of software to imitate, save, and load scratches, to pick the beats out of a piece of music, and to be able to scratch out tunes on a record. The thesis does not address the generation of scratches, or any other potentially creative processes.

Chapter 2

Background

2.1 DJing Basics

Currently, DJing is composed of four different skills. The most basic skill is beat matching. Beat matching involves making the tempos of two different records match by adjusting their speeds appropriately. Beat matching also involves aligning the beats so that they fall together. The second skill involved in DJing, called mixing, allows for the DJ to incorporate some creativity. Mixing is when the DJ switches between the two records and may use effects, or filters to make the two songs complement and interact with each other. The third skill, programming, is the most sophisticated function of a DJ. Programming is simply selecting the next record to play. However, when programming, a DJ must take into account the mood of the crowd and of the music and how, where, and when the incoming record will mix with the outgoing. The final skill called scratching is very different from the other three in that it uses the record player as an instrument, in and of itself, rather than as a medium for playing back previously recorded music. When a DJ scratches, he spins the record backward and forward at different speeds creating a very unique squeaking sound which can be precisely controlled by the speed with which the DJ moves the record and by the content of the record being scratched. [2]

2.2 History of Phonograph

The first sound recording device was designed and developed by Leon Scott de Martinville in 1855. He called it the "phonautograph." The phonautograph used a mouthpiece horn and membrane attached to a stylus that recorded sound waves onto a rotating cylinder wrapped with smoke-blackened paper. Unfortunately, there was no way to play the sounds back at the time.

In 1877, Thomas Edison invented the "tinfoil phonograph," which was the first device to ever play back a recording. Edison connected a mouthpiece attached to a diaphragm to a stylus that etched the recording onto a rotating piece of tinfoil. For playback, the mouthpiece apparatus was exchanged with a more sensitive diaphragm. His first recording was "Mary Had a Little Lamb." Throughout 1878, Edison continued to refine the tin-foil phonograph and continued to demonstrate it for audiences. However, his work on the electric light took much of his attention and he stopped work on the phonograph.

In 1887, Alexander Graham Bell revealed the "graphophone" which exhibited some key improvements over Edison's model. The cylinders were made of wax instead of cardboard and tinfoil, which improved recording clarity and length. It also used an electric motor instead of Edison's hand crank to avoid pitch fluctuation and it used a loosely mounted "floating" stylus for clearer conversion into sound.

The phonograph quickly became popular, spawning phonograph parlors where patrons could listen to a few minutes of music for a few coins. Phonographs also became a medium for advertising. Companies would mount phonographs that could be activated at the touch of a button in conspicuous places and play songs or oration occasionally interrupted by someone reciting their slogan.

In 1893, Emile Berliner released his own version of the phonograph which he called the "gramophone." Berliner's model used discs pressed in hard rubber which were much cheaper to produce than the wax cylinders in Bell's model. Additionally, the "gramophone" enabled copying from a zinc master disc to the rubber discs.

The commercial competition between Edison's, Bell's and Berliner's models led to many innovations. During the late 1890's, the Berliner model used discs stamped in Duranoid, a shellac-based plastic material that proved far superior to rubber. Edison's model became able to play two cylinders with one winding of the spring drive. Other models sprung up that were driven by compressed air, or that could play up to 12 minutes on one disc, or that used records made of chocolate that could be eaten when they wore out.

During World War I, the Decca, the first truly portable phonograph, was released. The Decca used a pleated and varnished paper diaphragm speaker, more durable cylinders and discs (which also allowed longer play and better quality) and a way of installing a tone arm mount for the stylus to reduce its size such that it could be shipped to soldiers overseas. Over 100,000 Deccas saw active service in World War I. Following the war, the American Society of Composers, Authors and Publishers (ASCAP) was formed to ensure that its members were paid for use of their work. [15]

2.3 History of Turntablism

In the late sixties in New York, DJ Kool Herc (a.k.a. Clive Campbell) was pioneering DJ techniques which would later be known as hip hop. He used disco or funk beats which MC's would rap over instead of the reggae beats that were popular. Because the percussive breaks that Kool Herc was using were relatively short, he learned to artificially extend them indefinitely by using an audio mixer and two identical records and cutting back and forth between the records. The term "hip-hop" comes from the DJ "hip-hoppin" back and forth between the two turntables.

In 1970, Technics released the SL-1200 model turntable, which is still the industry standard turntable.

In 1975, New York DJ, Grand Wizard Theodore, invented the idea of scratching a record by moving it back and forth to produce unique sounds. Grand Master Flash had first used the idea of "rubbing" a record for segues into other tracks, but Theodore was the first to use scratching as a medium on it own. The technique was invented by Theodore when he would listen to records in his home. His mother would come in to tell him to turn the music down, and he would hold the record in place with his hand while talking to his mother. He noticed that moving the record back and forth would produce unique and interesting sounds. He spent a few months practicing and developing the technique before he used it in any of his sets, but when he debuted his new sound, the crowd loved it and scratching was born. [17] [11]

House music, the predecessor of many forms of electronic dance music, is named for the Warehouse, a club in Chicago. The Warehouse was opened in 1977 and "was beginning to develop a new style that was deeper, rawer and more designed to make people dance." Additionally, the Warehouse along with the Paradise Garage in New York, were among the very first clubs to break race and sexual preference. Before these clubs existed, Blacks, Whites, Hispanics, Straights and Gays would segregate themselves in clubs, but at the Warehouse and the Paradise Garage, the emphasis was on the music. By the early 80's disco music had already been aiming its music specifically at DJs, with extended 12" versions featuring long percussion breaks for mixing, and it was not long before these records were being played every night at the Warehouse and the Paradise Garage. [10]

GrandMixer DXT, formerly know as GrandMixer DST, was the driving influence for many of the current top DJ's in the world. GrandMixer performed the solo scratches on Herbie Hancock's hit "Rockit" which was performed during the 1984 Grammy Awards and on Saturday Night Live. GrandMixer DXT's performance was seen by and inspired many greats including Mix Master Mike, Q-Bert, Cut Chemist, Babu and others. [17]

In 1987 the DMC (Disco Mix Club) held its first annual DJ competition. Since then the DMC has hosted prestigious DJ battles and is the accepted proving grounds for any up and coming DJ.

In 1995, DJ Babu first coined the word "turntablist." In a 1996 interview, he said, "My definition of a Turntablist is a person who uses the turntables not to play music, but to manipulate sound and create music." [5]

By the mid-eighties San Franciso became the haven for scratch DJ's and is still home of the best turntablists in the world. [1] DJ Q-bert, widely considered the most technically skilled DJ and two time DMC world champion, and a founding member of the Invisbl Skratch Piklz, resides in San Franciso.

2.4 DJing on the Rise

The recent popularization of the rave scene has greatly contributed to interest in DJing. DJ's all over the world are combining the tried and true turntable techniques pioneered in the 70's and mixing them with the new sounds of electronic dance music. In the UK, turntables are outselling guitars by a ratio of 2 to 1. [6]

Turntablists and DJ's still use turntables and records, even with the extensive availability of digital audio sources because of the nature of the medium. While CD's and MP3's are more compact and allow random access, the appeal of the physical medium offered by vinyl is still the deciding factor. The dynamics of a song recorded on vinyl can be seen simply by looking at the record. More importantly, however, the analog nature of vinyl allows analog cueing and analog pitch shifting, which is vital for the precision most DJ's require. Furthermore, it provides the DJ with tactile feedback as she manipulates the record.

2.5 Deep Blue

Chess has traditionally been a major battlefield in the war of man vs. machine. Other games such as checkers or draughts are too simple, while the Chinese game, Go, is too complex.

In 1950, Alan Turing, the renowned mathematician wrote the first program to play chess. It took eight years before the program was able to beat a human (albeit one who had learned the game a few minutes before the match). In 1967 a program called Mac Hack started to compete successfully in human chess tournaments. Computer chess tournaments were organized throughout the 70's and continue today in the form of the Microcomputer Chess Championship held every October. In 1983, a program called Belle was the first to attain the United States Chess Federation level of "expert." [14]

In 1985, three doctoral students (Hsu, Campbell and Anantharaman) at Carnegie Mellon developed a chess playing program called Chiptest. Chiptest was developed and renamed Deep Thought. It tied for first place with Grand Master Tony Miles in the 1988 U.S. Open championship and defeated the 16-year old Grand Master Judit Polgar in 1993 in a 30-minute game.

Another successful program of the time was called Fritz. It was especially good at games of speed chess. Fritz 2 defeated Kasparov in a 5 minute game (a game in which each player has a total of 5 minutes to make his moves) in Cologne in 1992. Fritz 3 also bested Kasparov in 1993 in Munich in a blitz tournament. Fritz 3 defeated grand masters Vladimir Kramnik, Viswanathan Anand, Boris Gelfand and Niel Short. Grand Master Robert Heubner refused to play it. Kasparov did defeat Fritz 4 in London in November 1995.

In the early nineties, Deep Thought was bought by the computer company, IBM, resulting in its name change to Deep Blue. It was extensively developed and in May of 1997 stunned the world by defeating Gary Kasparov 3.5 to 2.5, in a six game series.

It is important to note that Deep Blue calculates 200 million moves per second, while, according to IBM's statistics, Kasparov calculates three. This implies that, while the computer uses brute force to decide it's moves, humans use creativity and innovation. This is further demonstrated by the fact that Kasparov performed better against Deep Blue when he made unconventional moves and took risks than when he played conservatively. However, since chess is typically thought of as a game that requires intelligence, the fall of the best human player to a computer made headlines.

2.6 Eliza Project

Another attempt to simulate a human intelligence with a computer came in 1966 in the form of the Eliza Project. Developed at MIT by Joseph Weizenbaum, Eliza was designed to emulate a Rogerian Psychotherapist. Rogerian therapy is based on attempts to help a patient clarify his or her feelings by reflecting those feelings back to the client in an accepting, nonjudgmental fashion. When it first debuted, many mistook it for human. However, it has almost no intelligence whatsoever. It was originally only 240 lines of code. [8] It is based on tricks like string substitutions and canned responses based on keywords. [7]

2.7 Other Computer Assisted DJ Projects

A system called scratchrobot built by Stijn Slabbinck of the Spess company in Belgium uses a pneumatically controlled arm to drive a record. Scratchrobot only address the scratching aspect of DJing. The system takes input from e-mails sent to it, and translates them into commands which it uses to scratch the record. However, the scratchrobot can only utilize approximately 120 degrees of one rotation of a record, and is bolted onto the table around the turntable. Additionally it is remarkably unsophisticate, scratching randomly based on a translation of text. [18]

Dave Cliff of Hewlett Packard's laboratories in Bristol, CT has also created a system which addresses the programming aspect of DJing. It uses biofeedback sensors which monitor bodily functions of members of the audience to predict which songs will be hits and which will be duds. However, it only addresses programming and does not address the other skills involved in DJing. [3]

A recently released package called FinalScratch is similar to the robotic DJ system being proposed. It allows DJ's to use their turntables as mechanical inputs so that they can "scratch" digital sound files stored on a computer. Seeking, pitchshifting, cueing and other functionalities are also supported by final scratch. However, FinalScratch is only accurate when the DJ adjusts the record slowly. If the DJ scratches at a rate anywhere near the rate at which he normally performs, FinalScratch will fail. Additionally, FinalScratch has no intelligence and relies entirely on the DJ to do all the work. [9]

The robotic DJ system being proposed is based on DJ, I Robot Mk 1.0, the work of the supervising professor, Chris Csikszentmihalyi. I, Robot MK 1.0 is built around the motors and sensors rather than an existing turntable, making it extremely difficult, if not impossible, for a DJ to perform on it. However, in it's current incarnation, I, Robot can play a record, record a scratch or loop a previously recorded scratch on each of its three turntables independently. [4]

There has been extensive study into measuring the tempo of an audio input. In 1994, Large and Kolen described a beat-tracking model based on nonlinear oscillators [12], and in 1998, Eric D. Scheirer developed an algorithm using a small number of band pass filters and banks of parallel comb filters which is as accurate as human detection. [16] This algorithm performed well on most types of music, save jazz and classical. However, due to the nature of electronic dance music, such a complex algorithm is not necessary. A software package called Luminescence is designed to provide visualizations for audio output, and can accurately detect and predict beats in electronic dance music using a filter and simple frequency analysis [13].

Chapter 3

Problem Outline

Because of the continuing appeal of vinyl and turntables, the robotic DJ system will output audio directly from the turntable, instead of digitally prqocessing music. The system will not manipulate the digital information, but the records themselves, just as a human DJ would.

The problem is broken into three distinct sub-problems. The mechanical part of the problem deals with the actual physical construction of a system that will be able determine the position of a record on a turntable, as well as drive the record back and forth rapidly. The interfacing aspect of the problem entails creating and connecting electronic hardware that will allow a user to get positional information from a sensor and control a mechanical actuator with software. Finally, the software portion of the project encompasses the software that controls the mechanical system to mimic the techniques of human DJ's.

3.1 Mechanical

The mechanical requirements for the robotic DJ system are as follows. The sensor system must be able to detect the position of a record on a Technics 1200 M3D turntable (not the platter) within a 1000th of a rotation (0.002 seconds). Motion along or around any other axis must not affect the reading, so long as the record remains on the turntable. The sensor mechanism must also have minimal effect on a

DJ's use of the turntable. To this effect, the sensor mechanism must exert minimal torque on the record, and must not block the usable portion of the record. The sensor must also be versatile enough to work on warped or bent records. Furthermore, the sensor mechanism must not damage the record in any way. Finally, the sensor must not interfere with the audio output of the record player. Within these constraints, minimal weight and cost is desirable.

The requirement for the drive system is very similar. The drive system must also attach to a Technics 1200 M3D. The drive mechanism must be accurate to within a 1000th of a rotation, and it must be easily disengaged from the turntable so that the user can quickly switch between the using the system and using the turntable normally. The drive mechanism must also not damage the record in any way and it must not interfere with the audio output of the turntable.

3.2 Interfacing

The interfacing segment only has a single requirement beyond the obvious functionality it must provide for basic operation. The interface must allow the system to support up to thirty two sensor/motor pairs independently.

3.3 Software

The software sub-problem is further broken into a few distinct parts.

3.3.1 Style Stealing

The first and most fundamental function that the system needs to perform is the recording of scratches made by human DJ's. The system must be able to record at least 5 seconds of motion and duplicate the same motion using its own mechanical system. It must also be able to store the motions of the scratches in files and recall the scratches from files.

3.3.2 Tune Mimicking

The second software sub-problem is known as tune mimicking. Tune mimicking requires implementing a tune playing algorithm which allows the system to accept audio input and, by varying the speed at which a record with a constant tone is played, mimic the tune. The tune has to be composed of single frequency notes. The tune mimicking software must also be able to store the motions in files and recall the motions from files. Very skilled human DJ's are capable of doing this kind of tune mimicking, but only with a lot of time and practice, and never accurately on the fly.

3.3.3 Beat Matching/Tracking

Another proposed software sub-problem was beat matching. The original intent was to enable the system to beat match a record to a previously recorded audio sample. In other words, the system would need to match the tempo of the record it is playing to the tempo of a previously recorded sample. Additionally, the system would need to align the beats so that both the record and the audio sample play their beats at the same time. The system would only need to beat match records that have simple steady beats, and would only be able to match them if their beats per minute count are within 90 percent of each other. It was discovered that this could not be accomplished with the software sound library being used. However, the following similar possible problem was formulated. The system will, instead, need to record an audio sample and detect the beats in the sample. The sample must have a simple steady beat. The system will need to detect the beat of music that plays at 160 BPM (beats per minute) and slower. The beat detection must be accurate to with 25ms. per beat.

3.3.4 Performance Interface

Finally, the system needs to combine the above functionalities in a performance interface which will allow users to compose and edit scratch compositions. The system should record an audio sample of up to 45 seconds and, using the beat detector, pick the beats out of the sample. It should then allow the user to select any scratch previously recorded with the style stealer, the tune mime, or scratcher by ear, and load the scratch into any second measure (every eighth beat). For each scratch that is loaded, the user should also be able to dictate whether the system will reset the record after the scratch; that is, whether the record should spin back to the position it was at before the scratch started, or whether it should begin the next scratch at the place where the previous scratch ended. Furthermore, for each scratch, the system should allow the user to determine whether to stretch the scratch; that is, whether the scratch should play at normal speed or whether it should be expanded or compressed to play throughout the entire two measure period. If a scratch is longer than two measures and the stretch option is not used to compress the scratch into two measures, or if the system cannot reset the scratch before the end of the two measure period, the scratch should simply abort and the next scratch should commence from that point on the record.

Chapter 4

Design

The following section describes the design for the robotic DJ system. It includes the mechanical, interfacing, and software specifications.

4.1 Mechanical Design

The mechanical design for the system was mostly done on a trial and error basis. The parts were all machined out of acrylic for ease of construction, light weight, and low cost. The parts were designed in CorelDraw v9.0 and cut from large sheets of acrylic using a Universal Systems laser cutter. The power, speed, and dpi settings for the laser cutter were all determined on a trial and error basis. All joints were made using a methylene chloride, tricholorethylene, and methyl metacrylate monomer solvent. To join two pieces of acrylic, the solvent was applied heavily to both parts, the parts were pressed together in their desired configuration, and pressure was applied with a C-clamp for a few hours. Joints made in this fashion proved to be strong enough for all required purposes. The acrylic color selection for all the parts was completely arbitrary, with the exception of the motor's drive wheel.

4.1.1 Encoder Mount Design

When designing the encoder mount, the first concern was designing an interface with the record that wouldn't significantly affect its moment of inertia. This would give a human DJ the closest experience to DJing on a normal turntable while still allowing the system to track the record.

The design for the encoder/record interface includes a single piece of 1/4" acrylic. The piece is cut into a three pronged shape.

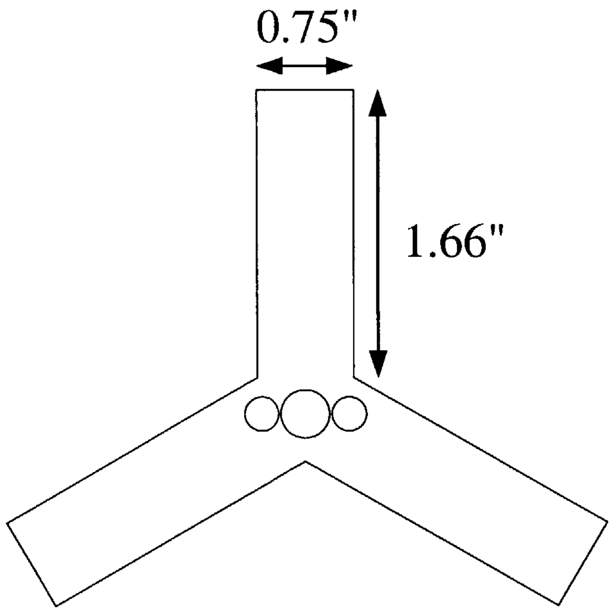


Figure 4-1: Three Pronged Record/Encoder Interface Piece

The piece is slightly smaller than the size of a record's label, so it sits in the center of the record and holds the encoder's shaft directly above the turntable's axle. A 0.82" hole was drilled in the center of the piece 1/8" deep so that the axle of the turntable could enter the hole to ensure that the piece is centered. On the opposite side, two 0.25" holes were drilled 0.53" apart. Three holes were drilled all the way through the piece at the ends of the prongs. They were tapped so that wood screws could be mounted in the holes to support the apparatus. The screws come from the side with two holes and pointed toward the single holed side. One of the encoder's pronged coupling pieces was attached to the end of the encoder's shaft and using a glue gun, the two prongs were glued into the two 0.53" holes.

Before the actual acrylic piece was cut, a mock up was made by cutting the shape out of three layers of corrugated cardboard and taping them together. Screws were mounted in the cardboard and the piece was tested to make sure the interface did not slip.

One downfall of this design is that the three pronged piece is too large. After a record is done playing, when the needle spirals into the center of the record, the acrylic piece hits the side of the cartridge and knocks it toward the outside of the record. This is not a major problem because the groove in the center of the record does not contain an audio signal and is only there to stabilize the needle when the record is done playing.

Failed Encoder Mount Design

The initial design for the sensor housing involved a counter balanced arm mechanism.

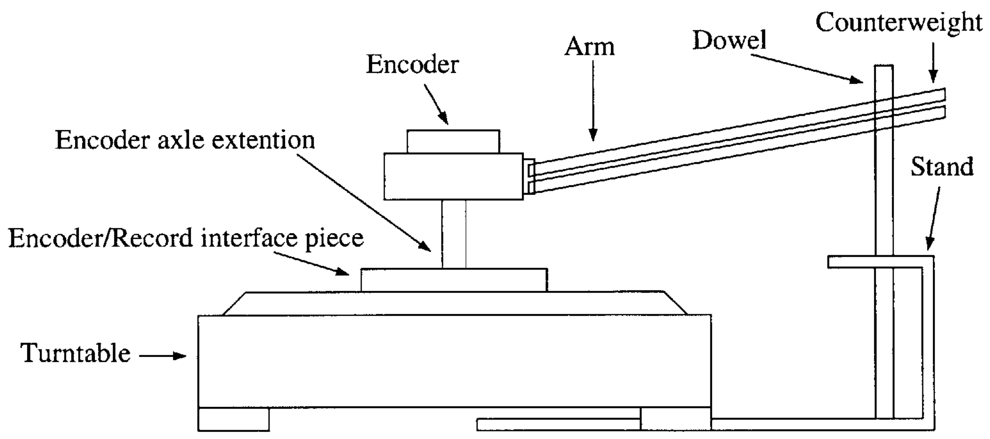


Figure 4-2: Failed Encoder Mount Design

The housing would hold the sensor vertically at all times, and well above the record's axle. The shaft of the encoder would be extended and would attach to the three pronged piece. A counter weight would be attached to the non-encoder side of the arm to allow the user to control the downward force of the encoder on the record.

The arm would be supported by a dowel which would swivel in a stand.

However, after preliminary construction it was found that the encoder could be supported without a counter balancing arm simply by placing it on the acrylic attachment and placing the attachment on the record. Thus, the final design only involves the encoder being supported by the single three pronged piece.

Because it sits in the center of the record and has a wire connection, the current position of the encoder prevents human turntablists from turning the record a complete rotation without removing their hand, a technique that some human DJs use. However, if the encoder mount were redesigned, it could allow the DJ this freedom. If another mount, similar to the one used for the motor, were created, but for the encoder instead of the motor and the same five layer drive wheel and rubber membrane system were used as the interface for the encoder, the DJ would have full freedom on the turntable.

However, this design does have its drawbacks. First of all, the encoder would no longer have a 1:1 coupling with the record. This problem could be solved by determining the actual ratio and recalculating the ratio between the motor's rotation and the record's rotation to reflect the change. Secondly, coupling the drive wheel to the record adds more moment of inertia to the record making it less like the experience of a regular turntable. Additionally, the motivation for using a separate encoder over the motor's built in encoder was to ensure accuracy in spite of slippage between the motor and the record. If it is assumed that there is a problem of slippage then we must also assume there is slippage between the new encoder design and the record, because it uses the same interface as the motor. However, the center mounted system can still claim to be slip free.

If the current encoder mount design is kept, the counter balanced arm could still be implemented to give the user control over the downward force of the encoder on the record. This could be useful in maintaining a slip free interface between the encoder and the record and in minimizing the system's impact on the regular operation of the turntable.

Alternatively, a rigid stand and clamp apparatus could also be designed to increase the accuracy of the encoder. Currently, the only thing keeping the main body of the encoder from turning with the shaft is the tension from the signal wire. While this setup is sufficiently accurate, ideally the body of the encoder would be held in place by something more robust. A goose neck pipe with an alligator clip on one end and a base on the other would serve this purpose well.

4.1.2 Motor Mount Design

Initial Motor Mount Design

The first issue encountered in designing the mount for the motor was the need for a stable base that could support the motor and could apply significant pressure to the record. Because the drive wheel would be applying pressure downward onto the record, the base needed to wrap around under the bottom of the turntable. Additionally, the mount would have to be fixed in the horizontal plane. Based on the shape of the Technics 1200, it was decided that the mount should be designed around the extra cartridge holder. The extra cartridge holder is a hole drilled into the case of the 1200 M3D which is designed to hold spare cartridges. However, it also proves to be a convenient place to mount a device and restrict its movement in all horizontal directions.

The initial design for the motor housing had the motor driving an acrylic wheel which would be clamped onto the face of the record.

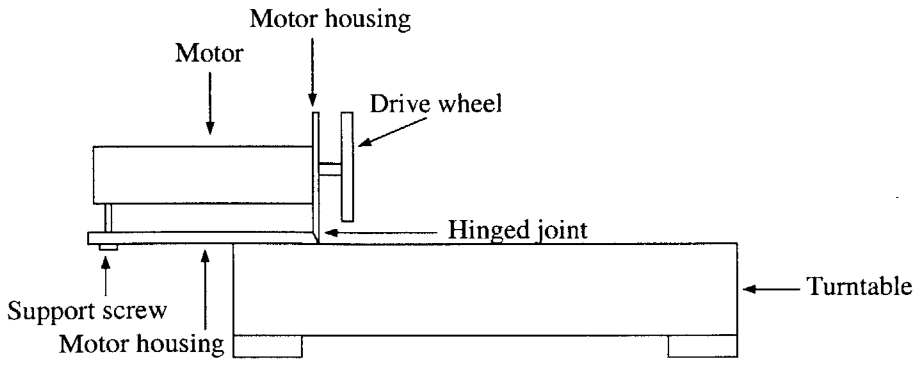


Figure 4-3: Initial Motor Mount Design (Platter omitted for clairty)

A model of this design was built and a rubber membrane was attached to the drive wheel. It included a mechanism to anchor it to the extra cartridge hole in the body of the turntable. It was attached to the Technics 1200, but testing revealed some fundamental flaws with the design. First, nothing was keeping the rubber membrane on the drive wheel and it would consistently slip off. Another problem with this design was that the drive wheel actually touched the surface of the record. When operating normally, the rubber membrane left traces of rubber on the face of the record, degrading the sound quality. When the rubber band slipped off the drive wheel, as it occasionally did, the bare acrylic would spin across the face of the record and often completely ruin it. Also, because the drive wheel was mounted vertically, the housing had to apply pressure downward on the drive wheel in order for it to get a non-slip grip on the record. However, regardless of the slip mats used, this also increased the friction between the record and the platter to the point that the record did not slip relative to the platter at all. Thus, the motor was driving the platter and the record as a unit. This additional moment of inertia significantly impaired the ability of the motor to accelerate and decelerate the record. A final concern with this design was the fact that it was not easily removed from the turntable. The support screw could be loosened to allow the record to play normally, but in order to completely remove the apparatus from the turntable the drive wheel had to be taken off the motor, then the motor had to be detached from the mount, then the anchor screw had to be taken out, then the mount had to be slid off the turntable, then the anchor screw would have to be reinserted on the anchor to pull the anchor out of the extra cartridge hole. Because of all these concerns, a new motor mount design was conceived.

Horizontal Motor Mount Design

The new design addressed the problems of the first by using a horizontally mounted drive wheel.

The drive wheel was composed of five different 1/16" layers of acrylic joined with the solvent. The outer most layers had the largest radii. Thus, the drive wheel had

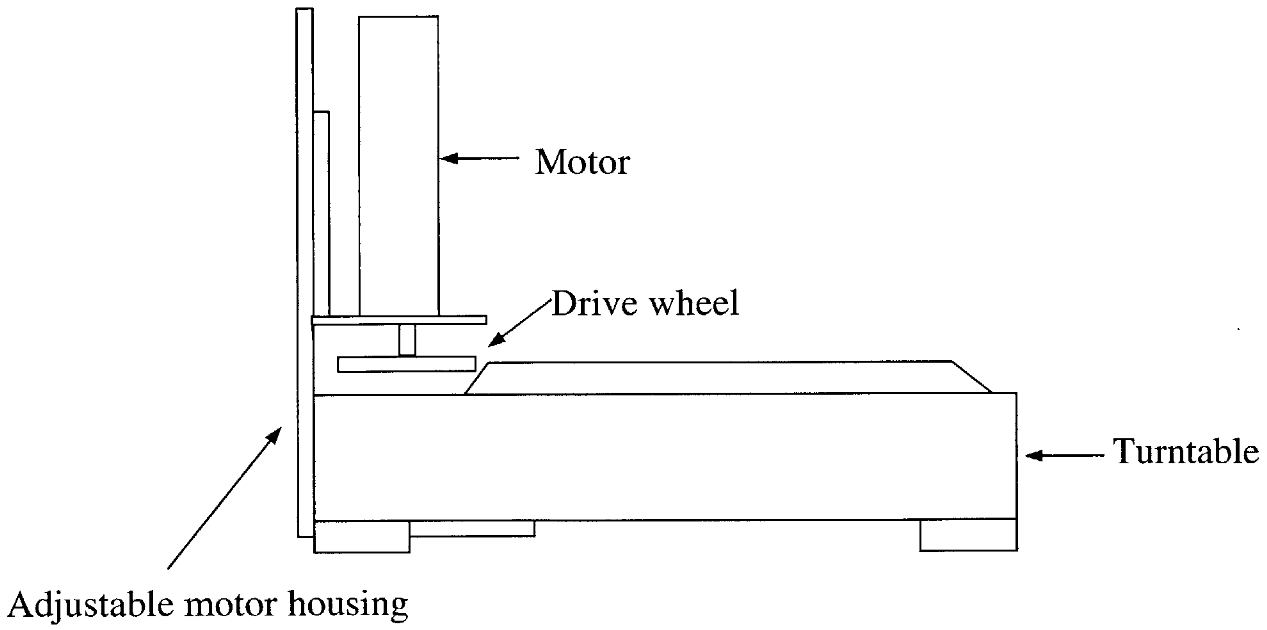


Figure 4-4: Initial Horizontal Motor Mount Design

a channel into which the record could fit. The rubber membrane fits between the outermost layers and sits around the second and fourth layers.

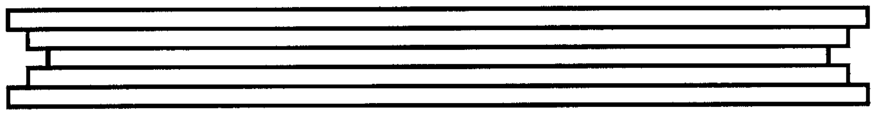


Figure 4-5: Profile of the Drive Wheel

The motor's five layered drive wheel was purposely constructed out of clear acrylic. This allows the user to see the set screw through the drive wheel for ease in attaching and detaching said drive wheel. It also affords a view of the rubber membrane, so the user can make sure it is in the proper position.

The horizontally mounted drive wheel does not touch the face of the record, so the playability of the record is preserved. Additionally, the force of the drive wheel pushes the record against the spindle, as opposed to onto the platter. Thus, with a good slip mat, the record and platter can move almost independently. The edges of the slip mat did have to be trimmed so that the bottom edge of the drive wheel did not make contact with the mat. Also, with a horizontal design, the apparatus can be removed or engaged by taking it from or placing it on the turntable in the appropriate place. No assembly or disassembly is required.

The problem with this design was that the size of the drive wheel was limited by the main support piece. This space was not large enough to accommodate a drive wheel that would be able to drive the record at the required speed. Two solutions to the space constraint problem were considered. The first was to redesign the main support piece so that it would have a slit through which the drive wheel could pass. It would also mount the motor on the side away from the record to allow the most flexibility in the drive wheel's size. The other possibility was to flip the motor upside down so that the shaft pointed upwards. This would eliminate the conflict between the main support piece and the drive wheel. The drawback to this design is that the motor is longer than the turntable is high. Thus, if the body of the motor were pointing downward, the entire turntable would have to be elevated.

Final Motor Mount Design

It was discovered that the dust cover for the Technics 1200 M3D was perfect for setting the turntable upon, as it is the same size as the turntable. It is also tall enough to give enough clearance for the motor to hang down the side of the turntable. Thus, the latter design of the two above mentioned designs was implemented.

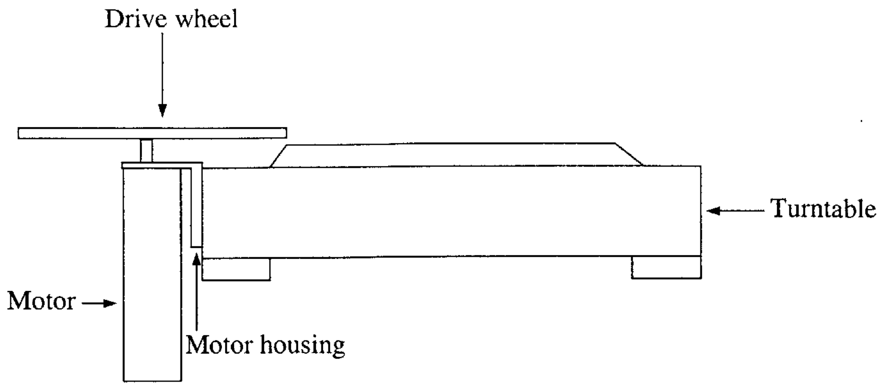


Figure 4-6: Side View of Final Motor Mount Design

The parts used to implement this design are as follows.

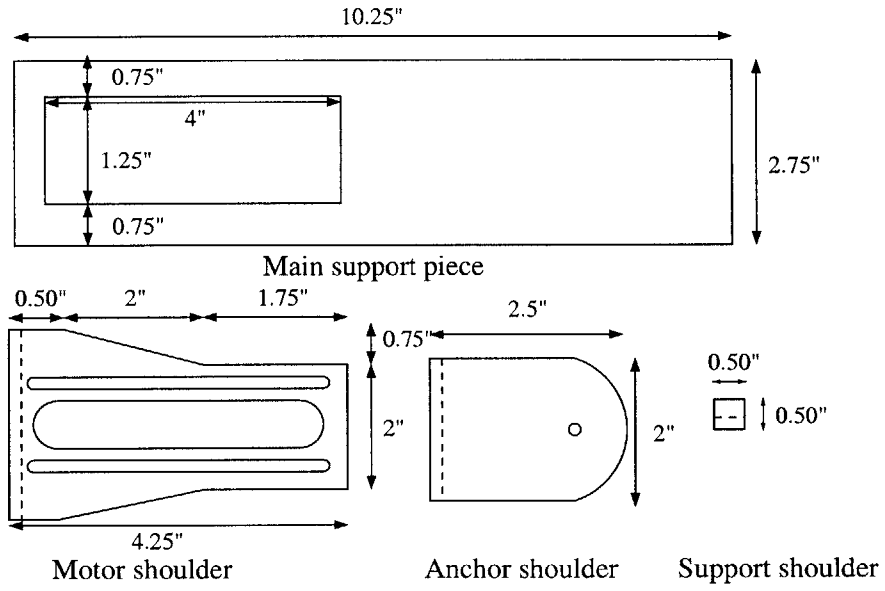


Figure 4-7: Parts used for Final Motor Mount Design

The dotted lines represent where the shoulders are attached to the main support piece. The main support piece holds all the other pieces together. It has a rectangular hole because the side of the Technics 1200 M3D is not flat. The MK2 model had mounts to which the dust cover would attach, but these were taken out in the M3D. However, the M3D still has protrusions in the spots where these mounts used to be. The support shoulder was attached to the side of the main support piece without the hole and provides stability. The anchor for the motor mount was created out of a plastic 1/4" dowel rod. A 3/4" piece was cut off and filed conically so that it fit snugly into the extra cartridge hole in the body of the turntable. The larger end of this piece was drilled and tapped so that it could be screwed onto the anchor shoulder. The anchor shoulder was joined with solvent to the main support piece opposite the support shoulder. It was joined on the same side. A hole was drilled and tapped to the same dimensions as that of the anchor. The anchor was securely fixed to this hole with a screw. The motor shoulder was similarly joined to the main support piece. However, it sits on the opposite side as the anchor shoulder and support shoulder. The slots and hole were cut in it to allow the motor to be attached with screws, but to allow its position be adjusted. These pieces were all cut from 1/4" acrylic.

A similar but larger drive wheel was built for this design. It is still composed of five layers of 1/16" acrylic, but the respective radii for the pieces are 5", 4.88", 4.63", 4.88", and 5". A similar drive wheel anchor was also cut out of 1/4" acrylic. It was drilled and tapped to accommodate a set screw and then joined to the drive wheel. The motor was then screwed onto the motor shoulder. The drive wheel was placed on the end of the shaft of motor and the set screw secured the drive wheel to the shaft. Finally the rubber membrane was placed between the first and fifth layers of the drive wheel so that it rested on the second and fourth layers. The apparatus was placed onto the turntable, and the motor's position was adjusted until a 12" record fit snugly into the channel between the second and fourth layers, and made good contact with the rubber membrane.

This design meets all of the requirements and was used for the implementation of the software requirements. It can drive the record faster than a human can and is as accurate as the encoder used to measure the position of the record. It is also very quickly and easily attached and disengaged from the turntable so the user can switch between the system's different modes.

4.2 Interfacing Design

Many of the solutions to the interfacing sub-problem were already available commercially.

4.2.1 Drive Motor

The motor used was the Pittman GM-9236C532-R2, which is in the Pittman GM-9236 series. It is a DC brush motor geared at 5.9:1. It's maximum load is 175 oz-in and is capable of 1189 RPM with no load. It has a built in encoder that generates 1024 pulses per revolution. The motor's other specifications including it's physical dimensions are available at:
"http://www.pittmannet.com/pdf/lcg_bulletin.pdf"

4.2.2 Encoder

A Lucas Ledex S-9974 optical encoder was used for the independent feedback. It is bi-directional, incremental, and generates 1024 pulses per revolution.

4.2.3 Other Interfacing

The J.R. Kerr Pic-Servo motor control board provides extensive functionality for polling the Lucas Ledex S-9974-1024 optical encoder sensor and commanding the Pittman GM-9236C534-R2's motor servo motor. It is commercially available and uses the RS485 protocal which allows the daisy chaining of up to thirty two Pic-Servo boards from a single serial port. It uses the encoder's input as feedback while driving the motor, so it can be as accurate as the encoder. Additioanlly, it interprets the encoder's signal making it accurate to 1/4096th of a rotation.

The J.R. Kerr Z232-485 board provides the translation between the RS232 protocal used by the serial port and the RS485 protocal used by the Pic-Servo motor control board. It is also commercially available from J.R. Kerr.

The logic power for the Z232-485 board was provided by a 12V AC/DC adapter.

The existing lead was cut off and a jumper was soldered on. A bit of glue was attached to the side of the jumper to physically prevent the jumper from being connected in the wrong direction, as such a connection could potentially damage the board. The motor power was provided to Pittman motor through the Pic-Servo Board by an existing 25V power supply. The Z232-485 was connected to the JP2 port of the Pic-Servo board with a 10-wire ribbon cable, and jumpers JP3, JP4, and JP5 were installed because the system was in a single controller configuration. The DB15 output from the Pic-Servo board was split into two separate connectors. Pins 5, 6, 13, and 14 were separated for the connection to the encoder. Pins 1 and 9 were separated for connection to the motor. This made it much easier to switch the connections for the encoder without affecting the connections to the motor. Details are available at "www.jrkerr.com".

4.3 Software Design

The main focus of this research is the software sub-problems. All code was written in Microsoft Visual C++ for Windows v6.0 on Windows 2000 Professional.

4.3.1 FMOD Sound Library

A large portion of the software is based around the FMOD audio processing API. It is a freely available software library (www.fmod.org) written by Brett Paterson. FMOD stores sounds as samples. Samples can be any length within the limits of memory, and can be recorded in 16 or 32 bits. They can be stereo or mono, can be recorded at frequencies up to and including 44100 Hz, and can be set to loop for recording and/or for playback.

FMOD supports multiple input and output channels. Input channels record into samples which can be played back on output channels. Samples can also be loaded from .wav, .mp3, or other files. A sample can be played back on an output channel at any speed by changing the playback frequency. The output channels have individual volume controls in addition to a master volume control that affects all output channels.

Additionally, FMOD provides functionality for digital sound processing during playback. Whenever a sample is playing, FMOD will call the DSP chain every 25 ms. as a callback. The DSP chain contains DSP units which analyze, process, and potentially alter the output. The units are executed in sequential order and pass the original and the modified sound to the next unit. Thus, the last unit in the chain outputs the modified sound to the output channel. The DSP chain has room for up to 1000 DSP units. Among the DSP units provided with FMOD are an echo unit, a reverb unit, a flange unit, and an FFT unit. The FMOD programmer can also create custom DSP units and insert them into the DSP chain.

Thus, to get the Fourier Transform of an output sample, a DSP unit was created. The DSP unit makes use of FMOD's GetSpectrum() procedure, which returns a buffer of 512 floats. GetSpectrum() requires that FMOD's FFT unit is active and returns the Fourier transform of the sound sample in the frequencies of 1 Hz to half the output frequency, 22050 Hz in this case. The frequency range is broken up into 512 bands, each of which is represented as an element in the buffer. The created DSP unit queried this buffer and populated a large array with the data. This DSP unit also writes the data to a file in an ascii graphic format. Each data point was represented by one character. The rows of the file were 512 characters long and each row represented one of the 25 ms calls to the DSP chain. The relative size of the point was reflected in the character that was chosen to represent it. For instance, a weak point was represented by '.' while a stronger point was represented with '@'. Thus, a quick glance at the file revealed many of the distinguishing characteristics of the sound sample.

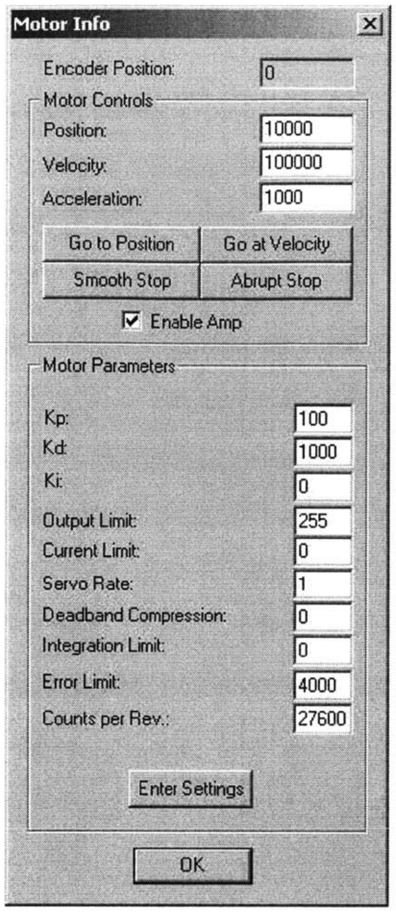
4.3.2 Motor Information Dialog Box

The first piece of software to be implemented was the motor information dialog box. The motor information dialog box serves as a way for the user to change the motor's characteristics and to test the motor. It is also the only way for the user to disable the motor's amp so that the record can be freely cued. The motor dialog

```
Time:      351 11.33#++^#*++^,+,-.+-++,--,-++---+^+--,+^^^+++^^****#*#*##@@@@@@@@@@@*###**##**^^^*^*^^^^*^
Time:      371 15.39#*^^*++^+*^^^*^^^**^*^*^*^*^*********#*###########@#@@@@@@@@@@@@@@@@##########**#******
Time:      401 13.93*+++*--^^+,,,.^^^^++^^^^^^^^^^^^^^*^*^^^^*^*****##*##########@@@@@@@^@@@@@################**#***
Time:      421  9.66**^*+^^-^^--+,-++-.----,,.,,.-,,.+-,,-++++++^^,*^^##@@@@@@@@@@@#@@#**+^+^,++--+++,,+--,
Time:      451 13.14#^*+*^^+^-,,-++,-+,-,+-^-^,^+^+^^^+*^^*^*^#*#*#######@@@@@@@@@@@@@@#*###*#*#*#*********^
Time:      471 12.75##^*^*#*,^*^^^*^****^***^***********#############@#@@@@@@@@@@@@@@@@#########******^*^^^
Time:      501 15.79#*^*#,-^,.,,--+-++++^**├--^^*^*^***^*^***^*#*##*###@#@#@@@@@@@@@@##^##,##+##*######*##*#^
Time:      521 14.07*,*+,^-,,^+,++---,.^+++^+,-,+^+^^-++^++^^^+^*^#*#^####@@@@@@@@@@@#@@#########**#*^****^^
Time:      551  8.44^#*+#+^^^^,-++--+++++-++++++++++++++^^^^^^+*^+^*****#*@##@@@@@@@@*-***^^*^^+*^^^+++^^^+
Time:      571  9.45*^#*+****^---,+-+--+,+---,+---++,+--++^++^++^,-,+*#####@@@*@@@@@@@###**^*+^.**^+,+-,^+
Time:      601  9.65^^++^^++++^+++---+-^++++^++++++++++^^++^^^^^^*^^*****#**##@@@@@@@@@###*#**^**^^^^^^^^^^+
Time:      621 10.05##**,*^-+^^^^++,++^,^++^+^^^^^^^^^^^^**^*^^*****^^*^**####@@@@@@@@@@@@########**#**^^^^+^^
Time:      651 10.71*^+^*--^*+++++-+-+-,,-----++++--+-++^^++^^^^^^**^*#*####@@#@@@@@@@@#@@###*#*********^^^
Time:      671  9.50*+***,*+-+--++,++-,+--^++^++^^+^*^^^^^*#**#*#@@@@@@@@@@@#####*#**^*^^^^-^^^^-+-^^+++,+
Time:      701  8.43*^*^**^^,+-+,+,,-,,--+,+--+++,^-^++^+*^^###@@@@@@@@@@@@@@#*****^^,+-*+-^+^-^,,^-++++-+.
Time:      721  9.80****-^^+^+^^+^-++-++^^^^^^^^^+^^*****^*^##*##@@@@@@@@@@@@@#@#*************^^^^+^*-+++++++-++
Time:      751 11.16-^**^^+-+++++^++-++^+^+^++*^^^^****^*#^#@*@@*@@@@@@@@@####***^****^*^^++-^^-^+^+^^-^-^
Time:      771 11.79##***#^+*-^^^+^^^*^^**********##*########@@@#@@@@@@@@@*######*###****^*^^*^+^^^^^^^^^^^
Time:      802  7.55^^^##*^-,-+,,-,,,-,+,----+,+++++-^+^^^^^*####@@@@@@@@@#,#*^*+*+-,^++--++--,-.+,-,-,-,-
Time:      822 10.30##*+*^*^^^^^^^^-++-+^+^*^^^^^^^^*******^*#**#@*@#@@@@@@@@@####*#*####*^+++****^^^+^++^^+
Time:      852  7.90+#^^*#^^^,+-,,,,+,.,,-,--,-,,-+-.,--+++^***#@@@@@@@@@@###*+--+,-,--.,+,+---.-..+-.-.,,
Time:      872 10.05#^-*#^^^^^^+^^^^+^^^*^^************#######@@#@@@@@@@@@@#@##*#*##**^**^*^*^^^^+^^++^++++
Time:      902  7.25#**#**-^^,-,-.-,.,..,..,,,--,-,,---,-++^##@#@@@#@@@@##*^^*-+,,-+,-,-,,,,-++--,.----,.
Time:      922  9.61#^###*^^+---,-++++^***###@@@@@@@@@@@##*^^^^+^^+^^+^+^+-+-+-+-+,++,--,,,,,+++^+^+^**#*
Time:      952 10.76^*#*^^^+-+^+^+^^^^^^^^**+#@#@@@@#@@@@@@@@@####^**^^^**^^+^++++-+--++,--,-+,--+--+,^^^^**
Time:      972  9.34^^*****++^+^-^+*^^^****###@@@@@@@@@@@@@@^-**^^-,-,^^^-+,^+++++,+,-+,------+-++++^++^^,**
Time:     1002 10.41#*-**#**^+^+**^*****#*#######@@@@@@@@@@@##*#*#**^^^*^*^-^^+^-^^+^--+-++++-+,,-,,-,-+-^+
Time:     1022 11.21**+,++^^,+,-^^^^^^^*^^#######@@@@@@@@@@@#######***#****^^**^^^^^^+^^^+^^^-^^-+^++++-+++
Time:     1052 12.26^***^*^++-,+****#**####@@@@@@@@@@@@#@@###############**#********^**^**^^^^^******##****
Time:     1072 12.23+#*^^+^+-*++******######*@#@@@@@@@@@@@@#########*********^^^^*^*^^^^^^+^++^++^+^++-+^-
Time:     1102 11.96#*#^#*^^^+,--^,^^*^*#****#^@@@@@#@@@@@@@#########*#**^*^**^^^*^**^^^^+^^^^^^-+^-+^*^^**
Time:     1122 12.05#^*##*^#********#*##########@@@@@@@@@@@#######****^**^*^^^++^^*^^+++^+++^^+-+++-,+^*+-*
```

Figure 4-8: An Example of the Ascii Graphic Format for a Three Note Sample

box is called modally and takes a pointer to the motor module. It displays the motor's parameters in the lower section labeled "Motor Parameters". The user can enter parameters in the fields and press the set button to set the motor parameters, at which point the parameters will be uploaded to the motor. It also displays the encoder position in the "Encoder Position" field. Using a Windows timer, the system queries the encoder thirty times a second. In the "Motor Contols" section the user may also enter a destination in the "Position" field. Pressing the "Go to Position" will command the motor to drive the record and consequently the encoder to the destination. The encoder will be driven at the velocity specified in the "Velocity" field with the acceleration specified in the "Acceleration" field. Alternately, the user can enter press the "Go at Velocity" button and the motor will drive the record at the velocity specified with the acceleration specified. If the specified velocity is negative, the motor's REVERSE flag will be set and the absolute value of the velocity parameter will be used. The "Smooth Stop" button causes the system to issue a "stop motor" command with the STOPSMOOTH flag. This will cause the motor to decelerate at the specified acceleration to a speed of zero and can be used to stop during both


Motor Info

Encoder Position: 0

Motor Controls

Position: 10000

Velocity: 100000

Acceleration: 1000

[✓] Enable Amp

Motor Parameters

Kp: 100

Kd: 1000

Ki: 0

Output Limit: 255

Current Limit: 0

Servo Rate: 1

Deadband Compression: 0

Integration Limit: 0

Error Limit: 4000

Counts per Rev.: 27600

Figure 4-9: Motor Information Dialog Box

a "Go to Position" and a "Go at Speed" command. Similarly, the "Abrupt Stop" button issues a "stop motor" command with the STOPABRUPT flag. This causes the motor to decelerate as fast as possible. Abrupt stops can also be used during both "Go to Position" and "Go at Speed" commands. The "Amp Enable" check box corresponds to the amplifier for the motor. When the amplifier is off, the motor's shaft can be freely turned and thus, the record can be easily cued to any point. When the amplifier is on, the motor attempts to keep the encoder at the position specified. Thus, any attempt to move the record will be met with great resistance, and any movement will be undone. When the motor information dialog is closed, the amplifier is automatically turned on.

4.3.3 Main Dialog Box

The main dialog box is where the user accesses all the other functionality. It is

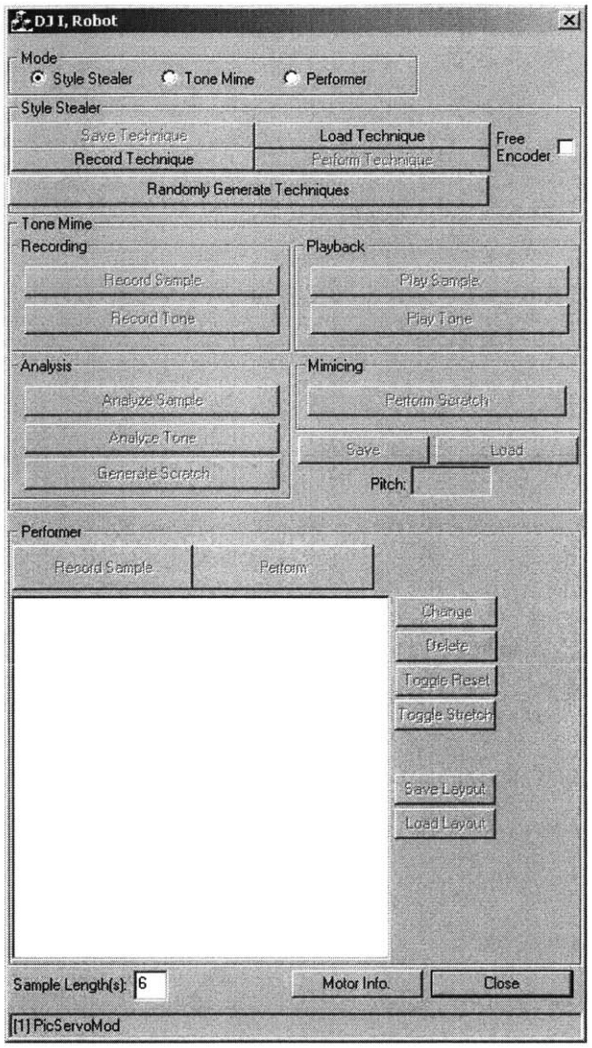


Figure 4-10: Main Dialog Box

called when the application is started. It handshakes with the motor module, sets the motor's parameters to appropriate values and initializes the FMOD API. It uses FMOD to initialize an input and an output channel and creates a DSP unit for Fourier transforms and another for the performance interface. It also initializes two FMOD sound samples: a main sound sample and a tone sample.

When the main dialog box is closed, it turns the motor off, as the motor may still be in operation and exiting the application will not automatically stop it.

4.3.4 Converting between RPM and Vel

Throughout the development of the software, it was very convenient to be able to convert the arbitrary angular velocity units used when inputting a speed to the motor (MIU's) to rotations per minute of the record (RPM's) and vice versa. In order to determine the conversion, the "Go at Velocity" feature of the motor information dialog was used. The Technics 1200 M3D has a built in strobe light which flashes over the platter, illuminating the speed dots on the platter. The speed dots are placed such that if the platter is rotating at 33 1/3 RPM, the dots will appear to be stationary. In order to determine the input for 33 1/3 RPM, the strobe light was turned on, the turntable's drive was turned off, and the motor information dialog's "Go at Velocity" feature was set to 50,000 MIU's. The speed dots on the platter were inspected and appeared to be progressing counter clockwise. Thus, the input, 50,000 MIU's, was driving the turntable at less than 33 1/3 RPM. The process was repeated in binary search fashion until it was discovered that when 76,450 MIU's was input into the motor speed field, the dots appeared to be stationary over a period of two miniutes. Thus, it was determined that an input speed of 76,450 resulted in the motor driving the record are 33 1/3 RPM. Thus the conversion between RPM and motor input units is as follows.

$MIU = RPM * 76450/(100/3)$

It was also convenient to be able to convert between pulses per second (PPS) and RPM's. Because the Lucas Ledex encoder is measuring the rotation of the record and the Lucas Ledex encoder generates 4096 pulses per revolution the conversion is as follows.

$RPM = PPS * 60/4096$

4.3.5 Style Stealer

The topmost section of the dialog box contains the controls for the style stealer. The style stealer allows the user to record and playback a scratch technique. It also allows the user to save and load the techniques to and from files.

Recorded techniques are a maximum of 5 seconds long, but can be indefinitely short. Once the "Record Technique" button is pressed, the system polls the encoder every 25 ms. for its position. The positions are recorded in an array as integers. This cycle continues until the technique's length has been reached.

Originally the system was going to allow the user to select the frequency at which the position was recorded. However, because the techniques recorded with the style stealer are used in the performance interface which relies on the FMOD DSP chain, a 25 ms. period was hard coded.

After the sample is recorded, the system trims the sample. Because techniques do not take integral numbers of seconds and it may take the human user a second or two to start the technique after the button is pressed, the system trims the unused header and trailer of the technique. The system starts at the beginning of the array and compares the recorded positions of the elements with that of the first position. It will omit all the positions that are within five encoder pulses of the original position. It will also do the same for the end of the technique. If all the positions recorded are within five pulses of the starting position, the technique will be thrown out and the user will be alerted that no technique was recorded.

The free encoder check box allows the user to select whether to use the Lucas Ledex encoder to record techniques or to use the motor's built in encoder. The system assumes that the Lucas Ledex encoder will be used for playback, so it multiplies techniques recorded with the motor's encoder by the ratio of the pulses per revolution.

Once a technique is recorded, the user can press the "Perform Technique" button to have the system repeat it. The system first assign its starting position the value 0. Then, for the length of the technique, the system polls the position array every 25 ms. and uses the motor's position mode to drive the record to the difference of that position and the starting position. Therefore, if the technique starts on position 10,000 and continues to position 10,500, the record will not be driven to position 10,000 and then 10,500, but 0 and 500 instead. This effect could also have been accomplished by subtracting the starting position from all the entries in the array.

However, if the system drives the record at a constant speed for every interval, the result will not be the same as the recorded technique. If the interval is large relative to the speed, the record will not reach its destination in 25 ms. If the interval is small relative to the speed, the record will reach its destination quickly and then stop. The latter results in the motor itself chattering and creating a noticeable unevenness in the playback. The solution was to set the drive speed directly proportional to the distance that had to be traveled during the interval.

A recorded technique can also be saved to a file. When the "Save Technique" button is pushed, the application will pop up a standard Windows file dialog so the user can select the filename. The application writes the technique out as simple text. The header consists of two lines. The first is the string "PosMode" which indicates that the file represents the positions of the record over time. The second is the number of elements in the array which are written in the file. Following the header, each line contains one element from the position array in sequential order.

Similarly a recorded technique can be loaded from a file. When the "Load Technique" button is pushed, the application pops up a standard Windows file dialog so the user can select the file to load. The application checks to make sure the file exists and that its first line is the string "PosMode". If this is the case, it will read the next line, the length of the array, and allocate an appropriate amount of space for the array of integers. It then fills the array with the contents of the file in sequential order.

When the user presses the "Generate Techniques" button the first thing the system does is call the rand() function a number of times based on the system clock. This is because the rand() function will return the same values over different instances of the application. In other words, the first value returned by rand() is always 53. By calling rand() a number of times based on the system clock, rand() will henceforth return numbers that are not the same across different instances of the application.

4.3.6 Tune Mimicking

The second software sub-problem is that of tune mimicking. The tune mime allows the user to record the singing or whistling of a tune and then have the robotic DJ system play the tune back by scratching a record with a single tone. The length of the tune is set by the user in the length field at the bottom of the main dialog box.

When the user presses the “Record Sample” button, the system activates the FMOD recording routines and stops accepting other input for the length of the sample, which is specified in the lower “length” field in the main dialog box. After the sample has been recorded, the “Play Sample” and “Analyze Sample” buttons become active. Pressing the “Play Sample” results in a call to the FMOD routine to play the sample back. All input is discarded during the playback.

When the user presses the “Analyze Sample” button, the system calls the FMOD playback routine, but it also activates the fourier transform DSP unit. Thus, for every 25 ms. interval of the sample, a 512 point spectrum ranging from 1 Hz to 22050 Hz is recorded into an array. For each 25 ms. interval, the frequency band with the largest response is also recorded. However, it was observed that, for an as yet unknown reason, the very lowest frequency band (1Hz - 43 Hz) carries a lot of noise and is consequently discarded. Additionally, if the highest frequency response does not exceed a threshold determined by listening to a sample at a barely audible volume, the interval is assumed to be all noise and is also discarded.

When the user presses the “Record Tone” button he is prompted to cue at least three seconds of the single tone record. Once the record is set and the user clicks the okay button, the motor drives the record at 50,000 MIU (approximately 22 RPM). Recording does not begin immediately, but 50 ms. after the motor begins driving the record. This is to compensate for the delay due to the motor’s acceleration. After the delay, the tone is recorded for three seconds and then the motor stops. After the tone has been recorded, the “Play Tone” and “Analyze Sample” buttons become active. Pressing the “Play Tone” button calls the FMOD routines to play the tone back. All input is discarded during the playback.

When the user presses the "Analyze Tone" button, the system calls the FMOD playback routines for the tone. It also activates the fourier transform DSP unit and stores the spectrum data in a separate array. As with the sample analysis, the frequency bands with the largest responses are all also recorded. The lowest frequency band is discarded and any intervals that do not exceed the threshold are also discarded. Once the maximum frequency bands for each interval are established, the frequencies that they represent are determined. FMOD's Fourier transform DSP unit returns an array of 512 elements each of which represents an equal frequency band from 1 Hz to 22050 Hz. Thus, the lowest element of the array represents the frequency band 1 Hz - 43 Hz and so is said to represent the band centered around 22 Hz. Because the tone is supposed to be single frequency, the maximum frequencies from each interval should all be the same. However, the system only assumes this to be true, and does not actively check it. The system does determine the pitch of the tone by taking the average of all the maximum frequencies. It then displays the pitch in the read-only "Pitch" field of the main dialog box. When both the sample and the tone have been successfully analyzed, the "Generate Scratch" button becomes active.

When the user presses the "Generate Scratch" button, the system calculates the motion necessary to recreate the sample by varying the playback speed of the tone. It first creates an array of floats with a number of elements equal to the number of 25 ms. intervals. For each 25 ms. interval, the system compares the highest frequency response from the sample with the pitch of the tone. It populates the float array with these ratios.

If the interval was discarded as noise in the sample, the system assumes that the noise was the highest response because there was no input and assumes that there was no input because the record was not moving. Thus, it assigns the playback speed a value of zero. This array now represents the speeds in MIU at which the record needs to be driven in order to recreate the tune from the sample. Sucessfully generating a scratch activates the "Perform Scratch" button.

When the user presses the "Perform Scratch" button, the system recreates the tune of the sample by playing it back on the record. It takes the ratios stored in the array and inputs the ratio multiplied by the original play speed of the tone to the motor.

$$PlaybackSpeed = OriginalToneSpeed*FrequencyOfSample/FrequencyOfTone$$

Every 25 ms. the system uses the motor's velocity mode to command it to drive the record at the speed dictated by the corresponding element of the ratio array. However, it skips the first 5 intervals of the sample. This is because FMOD's Fourier transform unit does not accurately record the Fourier transforms of the first tenth of a second of a sample. This was determined by visual inspection of the ascii graphic file output. Once the sample is finished, the motor is turned off. Otherwise, it will indefinitely continue at the last input speed.

The tune mime also has the ability to save and load its scratches to and from files. When the user presses the "Save Scratch" button, the system opens a standard Windows dialog box to prompt the user for the file in which to save. The data is written out as text. The first line of the file is the string "VelMode" indicating that the file is a list of ratios for velocities. The second line is an integer indicating the velocity in MIU's at which the tone was played. The third line is the number of 25 ms. intervals that were recorded, and consequently the number of elements stored in the rest of the file. The rest of the file contains one line for each interval. Each of these lines contains a single float which is the ratio of the playback speed for the sample at the corresponding interval to the tone's speed written in the second line. This data was already stored in an array when the scratch was generated. Note that the ratio data and the original tone speed are all that are needed to recreate the tune sample.

When the "Load Scratch" button is pressed, the user is prompted for a file to load. The system makes sure the file begins with the string "VelMode." It then reads the tone's velocity from the second line and the length of the rest of the file from the third. It allocates appropriate amounts of space to store the ratio array and loads the data into the ratio array.

The file format stores the motion of the record needed to recreate the tune. However, it would also be possible to store the tune itself by saving the tone's pitch instead of the speed at which it was played. By doing so, each 25 ms. interval's pitch is stored rather than its playback speed. Thus, if a new tone record were used, a simple comparison of the original tone's pitch and the new tone's pitch would allow a reconstruction of the original tune. The former file format was used in this implementation so that it the scratches created with the tune mime could be used in the performance interface.

The tune mime is accurate with the exception of lags due to acceleration and deceleration. Especially in the case of whistling input, when the user stops to breathe or change notes, there is no input and the system sets the drive speed to zero. Thus, the record decelerates to a stop and then accelerates again when the next note begins. This results in a time when the output of the turntable is in an intermediate state. The output is neither zero nor the actual note.

This could be solved if the independent volume control discussed in §Future Work were implemented. Instead of setting the speed to zero for the intervals with no input, the system would set the volume to zero and continue driving the record at the same speed. Then, when the input returned, the speed would be adjusted and the volume would be set back to normal. This would allow the system to only have to accelerate the difference between the speeds of the two notes rather than down to zero and back up again.

Also, when miming a tune the system always drives the tune record forward. However, this means that the tone has to be as long as or longer than the sample in order to be able to mime the entire thing, or be in a circular groove rather than the usual spiral groove on the record. However, if the system were able to drive the record backwards in addition to forwards, this problem would be eliminated. This could be accomplished by having the system detect the position at which it started and at each interval, calculate whether it had enough tone left to continue in its current direction. If it did, it could continue and if not, it could toggle the motor's REVERSE flag (the motor only accepts positive velocities and uses a binary flag to tell it which direction the velocity is indicating). The tone's length is known to be at least three seconds because of the requirement placed on the user to supply such a tone. Thus, the tone is at least 1/20th of a minute which, when played at 33 1/3 RPM, is at least 1 2/3 rotations. Thus, it is at least 6,827 encoder pulses long.

The disadvantage to moving the record back and forth is that, at the points of direction change the mimicking will become distorted. As when input gaps occur in the normal sample, the motor needs to decelerate and accelerate the record to the appropriate speeds. During this time the record is being driven at an intermediate speed and not the speed needed to mimic the tune.

The single direction approach was chosen in the implementation of the system because the tone record used had a circular groove which could be indefinitely played in the same direction.

4.3.7 Beat Detection

The user starts the process of creating a scratch composition by pressing the "Record Sample" button in the "Performer" section. This will begin recording an audio sample of the length specified in the "length" field in the bottom of the main dialog. It will also play back the sample with a small delay and take the fourier transform of the data. Because the DSP chain, and hence the Fourier transform DSP unit is called every 25 ms. the number of intervals is the length in seconds mutiplied by forty. Thus, the system will populate a 512 by forty times length array with the fourier transform data.

Once the sample is recorded and the data is stored, the system analyzes the data and detects the beats in the sample.

Algorithm Concept

It was decided that the algorithm would be separated into two distinct parts. The first would try to find the beats by analyzing the data from the Fourier transform. It would know characteristics that distinguish beats from other information and identify those characterisitcs in the data. It would allow both false negatives and false positives, but err on the side false negatives. Then the second part of the algorithm would analyze the beats that were labeled by the first part of the algorithm and add any false negatives and remove false positives based on the idea that a beat is a regular periodic occurrence. This type of algorithm was chosen because it makes good use of the knowledge that beats come periodically, whereas a single algorithm trying to identify and recognize the local pattern of a beat would have a harder time making use of this information.

A boolean array of length equal to the number of 25 ms. intervals in the sample would be constructed. Each beat would be represented by setting a single element to true at the corresponding interval.

Beat Detection Metric Evaluations

The first thing the system does in order to locate the beats in the sample is to analyze the Fourier transform for each 25 ms. interval and assign it a single number. These numbers should be such that their pattern clearly indicates the locations of the beats.

The first metric considered was the sum of the responses in all 511 frequency bands (the lowest was thrown out because of its propensity for noise). This did not reveal any useful characteristics, as only the music occupies the first 99 bands or so and noise in the upper bands obscured any pattern in the rest of the data. Thus, the second metric considered was the sum of the responses in the first 99 frequency bands. This also did not reveal any pattern because all the non-beat data in the audio sample obscured the characteristics of the beat. After visual inspection of a file output by the Fourier Transform DSP unit in the ascii graphic format, it was observed that a beat only affected the response in the lowest five bands. Thus, the third metric considered was the sum of the lowest five frequency bands.

When using this metric, beats appeared as large spikes, trailing off into a plateau. While it would have been possible to use this metric for beat detection, other musical characteristics occasionally caused spikes of similar magnitude. In order to attempt to separate these many other similar metrics were considered. Summing the squares of the responses instead of the responses themselves eliminated much of the other noise, but still did not significantly distinguish the spikes from noise or other music with the spikes from beats. Weighting each of the five bands in different ways also did not distinguish the two kinds of spikes either. Thus, the metric was changed to it's current form, the sum of the first and second frequency bands. This separated the noise and music spikes from the beat spikes more clearly. Visual inspection could quickly distinguish the actual beats from the false spikes. However, other means were needed to allow the system to accurately detect the beats.

Safety Threshold

The beat detection algorithm also establishes a safety threshold. In order for any sample to be considered a beat, its value must exceed the safety threshold. The motivation for having a safety threshold is distinguishing between actual beats and noise or other music whose metric evaluation has the same form as a beat, but at much weaker volumes. It makes sure that the a potential beat not only has the correct pattern, but also a significant volume. The challenge is to set the threshold between the height of the beat spikes and the rest of the music. If set correctly, an actual beat will exceed the safety threshold while noise and normal music will fall below it. However, the safety threshold must be dependent on the sample and cannot be hard coded. This is because the input volume may be different for different audio samples so the beats in a quiet sample may be at the same level as the music in a loud sample.

The first attempt to establish a safety threshold involved simply averaging all the metric evaluations of the intervals. However, this failed because many of the intervals' evaluations exceed this number, so an insignificant number of potential beats were ruled out by this threshold. Fixed linear scaling of the average didn't prove to be useful either. No single value could be found that would separate the actual beats and false positives for multiple samples of music. The idea of using the average of all the intervals was aborted because many seemingly harmless characterisitcs of the sample could drastically affect its outcome. For instance, if there is a dead space in the sample before the music begins playing, it will pull the average down sharply, but clearly doesn't affect the threshold between actual beats and other spikes.

In order to compensate for this, a more in depth method for determining the safety threshold was considered. Reflection revealed that the safety threshold should be just below the lowest beat spike. Thus, the next potential safety threshold considered was the average of all intervals whose metric evaluations were local maximums. This was easily checked by scanning the array for elements that were larger than both the element before and after them. Unfortunately, this did not generate the desired result because a great number of intervals were local maxima. Both in dead space and music, almost one quarter of all the samples were local maxima. Thus, the same idea was extended to only include intervals that were greater than the two intervals before it and the two intervals after it. This eliminated a large number of intervals representing music, but retained all the intervals representing beats. This threshold also accurately reflected changes in the difference between values for beats and values for music music for many different samples of music. However, the particular value occasionally exceeded the metric evaluation of weak beats, so the threshold was scaled by 3/4. This value was the safety threshold used for the rest of the algorithm.

Because of the success of the previous method, dynamic linear scaling of an the overall average, based on the characteristics of the particular sample, was not explored. However, it is unlikely to produce better results because of its dependence on such things as the dead space in the beginning of the sample.

The final concept was also extended to only include intervals that were greater than intervals three ahead and three behind. The results were very similar to those of the final method, and no particular advantage or disadvantage was seen in either method.

Dynamic Threshold

The next stage in detecting the beats was discovering and identifying the local pattern that represented a beat in the music. After visually inspecting the metric evaluation data of many music samples, it was determined that beats were almost always represented by a large spike followed by a plateau. The spike lasted one, two, or at most three intervals, while the plateau lasted for five or six intervals. The plateau was also approximately 3/4 the level of the spike. The rest of the intervals before the next beat were at a much lower level. Almost all spikes due to noise or other elements of the music were the same level as the beat's spike, but only lasted a single interval. In order to identify the pattern of the beats the system steps through the data and compares the level of each interval with the average of the levels of the five previous intervals. If the interval in question exceeds the average of the five previous intervals, henceforth the dynamic threshold, and exceeds the earlier established safety threshold, the system checks if the subsequent interval exceeds 5/8 of the safety threshold. This eliminates most false positive spikes because they only last one interval, but retains all beats because of the width of the spike and the trailing plateau. If an interval does meet all the criteria, it is labeled as a beat. The algorithm continues to step through the array and will not label another interval as a beat until an interval first falls below 1/4 the height of the spike. This is to ensure that a single beat with a wide spike is not accidentally labeled as more than one beat. Once an interval falls below 1/4 the height of the spike, the system sets the dynamic threshold to 1/4 of the height of the spike until five intervals have passed without a beat. When five intervals have passed, the dynamic threshold will return to the average of the levels of the five previous intervals.

The system averages the previous five intervals' metric evaluations when looking for a beat. The length of the window was determined empirically by testing different window sizes on different pieces of music many times. If the window is too large, the threshold does not reset quickly enough after the end of the plateau to catch the next beat. If the window is too small, the effect of a single interval affects the threshold too much and noise becomes a large problem. A window of five intervals was found to be the best compromise.

It would have been possible to find the local maximums throughout the sample and label them as beats, or intervals that were larger than some number of intervals ahead and some number of intervals behind them. However, in the cases where the spike was three intervals long, the local maximum may be the third interval after the beat actually occurred, which is 75 ms. later. As beats occur approximately every 400 ms. this lag in not acceptable. Thus, the algorithm must identify the first interval of the spike.

Interpolation

As discussed in §Algorithm Concept, the first half of the algorithm tries to recognize the local characteristics of the beats, allowing for both false positives and false negatives. The second half of the algorithm uses the knowledge that beats occur in steady regular intervals to remove the false positives and add in the false negatives.

The algorithm counts the distance between each pair of consecutive beats and finds the distance which occurs the most. However, because of the granularity of the intervals and slight inconsistency of the music, the comparison is not direct. When counting the distances of length n, the distances of length n-2, n-1, n+1, and n+2 are also included. This is to ensure that if distances of n-1 occur five times, distances of n occur five times, and distances of n+1 occur five times, and the distance m occurs six times, then n will be considered the most frequent distance. The width of the window was determined by visually inspecting results from many trials of many different songs and observing that the actual beats are always within two intervals of the most frequently occurring distance. Additionally, if more than one window shares the same total, the window centered on the distance that occurred the most is chosen. For example, if the distance n occurs eight times and the distance n+2 occurs 1 times, then the window centered on n has the same total as the window centered on n+2. However, clearly the most frequent distance should be considered n. If two windows have the same total and are centered on distances that occur the same number of times, the lower one is arbitrarily chosen as the most frequent distance.

Once the most frequent distance is discovered, the distances between each consecutive pair of beats are checked to see if they are of this distance. If the distance is smaller than the most frequent distance minus two, it is considered a false positive and is removed.

Similarly, if the distance is between twice the most frequent distance minus two and the most frequent distance plus two, the beat is removed. Also, beats between twice the most frequent distance plus three and three times the most frequent distance minus four, between three times the most frequent distance plus four and four times the most frequent distance minus five are removed. These beats fall into spaces that are not twice, three times, or four times the length of the most frequent distance. If the beats fall within a small margin of these distances (twice, three times, or four times the most frequent distance), it is assumed that there were one or more false negatives in between the previous beat and this one, rather than that it's a false positive. However, if the beats do not fall into these windows, it is assumed that they are false positives. The reason the windows increase in size as the multiple of the most frequent distance increases is because of the greater chance that the above mentioned variations in distance are compounded. However, because the variations can occur in both directions, the margin does not increase linearly with the distance's multiplier. The system assumes that it no longer has conclusive data when any beats are further apart than four times the most frequent distance. It assumes the beat after the large distance is correct and it ignores the distance and continues.

For the beats that are two, three, or four times the length of the most frequent distance apart, the beats that are missing are assumed to be false negatives and are filled in. The first interval to surpass the safety threshold that is also at least the most frequent distance minus two intervals, but not greater than the most frequent distance plus two intervals away is labeled a beat. If none of the intervals surpass the safety threshold, the greatest one is labeled a beat. If the gap is three or four times the length of the most frequent distance apart, the same procedure is reiterated to fill in the next missing beat, until the entire gap is filled in.

For simplicity, the current implementation of the system assumes that the first beat is not a false positive. The first beat could be tested to determine whether it is a true or false positive by comparing the distances between it and the next few beats to the most frequent distance. If the beat is a false positive, these should all be the most frequent distance plus some constant factor whereas, if the it were a true positive, they should all be the most frequent distance.

4.3.8 Performance Interface

The performance interface section of the software integrates all the other parts into an environment that allows users to compose their own scratch compositions and have the system perform those compositions. It records a piece of music and detects the beats in the sample using the beat tracking algorithm. It then allows the user to select scratches recorded with the style stealer or tune mime and play them back on every 8th beat. Once the beats have been identified, a boolean array with an element for each interval of the sample is declared, and the elements of the array corresponding to the intervals labeled as beats are set to true. All others are set to false. The beats are then counted, and three arrays with lengths equal to the number of scratches are allocated. The number of scratches is determined by adding seven to the number of beats and then dividing by eight and dropping the remainder. Thus, for every eight beats or part thereof, one scratch will be played. Two of the arrays are boolean and determine whether or not to reset or stretch each sample. The third is an array of strings and contains the names of the scratches to be played on the 8th beats. The user can set the contents of these arrays by selecting the corresponding line of the layout list box and pressing the "Toggle Stretch", "Toggle Reset", and "Change" buttons. Double clicking the line has the same effect as selecting the line and clicking the change button. The user can also remove a scratch from the layout by selecting it and clicking the "Delete" button.

If the user changes a scratch by selecting it and then pressing the "Change" button or by double clicking it, a file dialog box pops up. The user can select a scratch file to load into the spot selected. If the file's first line is not "VelMode" or "PosMode" it will not appear, because it is not a recognized scratch file format. If it is, the file will be checked to make sure it's formatting is correct (as described in §Tone Mime and §Style Stealer). However, the file will not be loaded. Changing the scratch causes the corresponding elements of the reset and stretch array to false.

After the user has recorded and analyzed a sample and has entered the desired scratches, he may also choose to reset or stretch each scratch. A scratch that is reset will cue the record to the same place the scratch was started when it finishes. A scratch that is stretched will be lengthened or shortened to fit into the time between the first seven beats of the eight beat period. The time during the last beat is reserved for resetting the record if the user so chooses.

Once the user presses the "Perform" button, the system will perform the composition. It starts by activating the performance interface DSP unit and then playing back the sample. As the sample plays, the performance interface DSP unit is called as part of the DSP chain every 25 ms. The performance interface DSP unit looks at the corresponding element in the boolean array representing the beats. If the interval being played back is a beat, the system stops the motor, ending the previous scratch if it is still executing. It then reads the file for the next scratch. If the file is missing or is not in the "VelMode" or "PosMode" formats, no scratch will be performed. If the scratch is set to be stretched, a coefficient of stretching is calculated. This is the new length of the scratch relative to its original length and is determined by dividing the length in time of seven beats by the length in time of the sample.

Because both file formats were recorded and saved in 25 ms. intervals, playing them back with the DSP unit is very convenient. If the scratch is in "VelMode", every 25 ms. the system drives the motor at the next velocity specified in the file. If the system is in "PosMode", every 25 ms. the system drives the motor to the next position specified. Once again, the speed for a "PosMode" formatted scratch is not constant, but is linearly dependent on the distance to be travelled. If the scratch is set to be stretched and the scratch is "VelMode" all the velocities are divided by the coefficient of stretching. If the scratch is in "PosMode" the positions are divided by the coefficient of stretching. When the scratch is finished, if the scratch was set to be reset, the system will return the record to its starting position. If it was not set to be reset, the record will stay where it is for the rest of the eight beat period.

If a scratch is not stretched and is longer than an eight beat period, the following scratch will interrupt it and begin right away. Whether or not the scratch was set to be reset, the following scratch will continue from the point at which the long scratch was. If a scratch is set to be reset and is only slightly shorter than an eight beat period, its reset may also be interrupted by the following scratch. A similar result to the above will occur.

This cycle continues for the duration of the sample with one scratch, if specified, being played every eight beats.

The stretching could have also have been implemented to be variable depending on whether the scratch was actually to be reset. If it were to be reset, the stretch would occupy the first seven beats, reserving the last for the reset as it is now. However, if the scratch were not to be reset it could occupy all eight beats. This functionality was not included for ease of implementation.

The system is also capable of saving and loading scratch layouts to files. The files are saved as simple text with the string "Layout" as the first line. The second line is the number of scratches in the layout that are to be saved. The following lines each represent one scratch in the layout. The first character of the line is '1' or '0' depending on whether the corresponding reset toggle was set or not. This is followed by a space, followed by a '1' or '0' to indicate whether the stretch toggle was set. This is followed by a space followed by the full pathname of the file in which the scratch is stored. If no scratch has been set in the layout, the string "NULL" appears.

When loading a layout, the system checks to make sure "Layout" is the first line of the file. It then loads the stretch and reset arrays, and the filenames of the scratches in the layout. However, it does not load filenames that read "NULL" but instead leaves the corresponding elements in the string array as empty strings. Note that when loading a layout, the system does not check to make sure that the scratch files exist. Thus, if the files are no longer in the same locations as when the layout was first created, an attempt to perform the loaded layout will result in the missing scratches not being performed. The other caveat of loading a layout is the fact that the sample may not have the same number of scratches as the saved layout. If the saved layout has fewer scratches than the current sample, the extra scratches are simply initialized to no reset, no stretch, and the empty string. If the current sample has fewer scratches than the layout, the extra lines in the layout file are discarded.

4.3.9 Changing Sample Length

Because all the information regarding the sample in the performance interface and in the tune mime is stored as an array, the arrays must be reallocated whenever the user changes the length of the sample. Thus, every time the sample length field is changed to a non-negative number, the system destroys the existing arrays and samples and reallocates new ones of appropriate size. All existing data regarding the original sample, and the sample itself are lost.

Chapter 5

Evaluation

5.1 Mechanical

The mechanical section of the system meets all the requirements of the problem. Neither the interface between the record and the drive wheel nor the interface between the record and the encoder slip at any normally used speed. The drive wheel and motor housing is easily removed from the turntable, is light weight and is cheap.

5.2 Interfacing

The interfacing section of the system also meets all the requirements set forth. It affords quick, accurate control over the position of the record and is extendible to allow up to thirty two versions of itself to be daisy chained.

5.3 Software

5.3.1 Style Stealer

The style stealer addresses the issues brought up in the §Problem Outline. However, occasionally when a technique is recorded that involves very fast scratching, the audio output generated by repeating the technique does not match generated when recording the technique, even though visual inspection reveals that the record traveled the same path. It is suspected that this is due to the dynamic calculation of the record's speed for each interval, and further testing and trial with this calculation will solve the problem.

5.3.2 Tune Mimicking

The implementation of the tune mime fulfils all the requierements of the problem. It can accurately repeat any whistle or song given as input or the scratching of a single tone sample. However, because the record being used to mimic the tune has a slight waver in its tone, the audio results did not always match exactly. The application of tune mimicking to scratching by ear (described in §Potential Improvements) is still unclear.

5.3.3 Peformance Interface

The beat tracking algorithm was tested on the beat loops from the John Johnson Mix of Bedrock's "Heaven Scent", the E-Werk Club Mix of Paul Van Dyk's, "For An Angel", the Original Mix of ATB's "9PM", and Art of Trance's "Madagascar". All were input from a 12" record and were cued to approximately half a second before the beginning of the song. All four samples have simple quarter note beats. The algorithm was able to detect and count the beats properly for all the test samples at multiple volumes.

When tested on any sample that didn't have simple quarter note beats, the algorithm failed.

When tested on silence, the algorithm occasionally failed. Because of the dynamic nature of both the dynamic and safety thresholds, a few false positives were detected. Thus, the algorithm thought it had established a pattern of beats when it in fact had not. It filled in the rest of the pattern. This problem could be corrected by requiring that the most frequent distance occur more than a certain number of times.

The actual performance of the scratch compositions worked well. The scratches selected started right on the beats and the stretching and resetting functionalities operated as proposed. However, the same problem when playing back stolen techniques mentioned above was present.

Chapter 6

Potential Improvements

This work could be extended in many ways to further increase its functionality. It is not at the level of DMC competition, but with the addition of some or all of the following, it could best most human DJ's.

6.1 Remachining Parts

Further mechanical work could be done in the form of remachining some parts of the motor mount out of metal. The drive wheel is not mounted perfectly horizontally on the shaft so the record tends to flutter up and down when being driven rapidly. This flutter leads to the more frequent skipping of the needle. Lathing the part, especially out of metal, should make it able to be mounted horizontally. However, making the drive wheel out of metal may increase its moment of inertia to the point that the motor can't drive it fast enough to meet the requirements.

Other parts of the motor mount could be made out of metal for added stability. However, the three pronged encoder mount should be left as acrylic. 1/4" acrylic is sufficiently strong and durable to perform the requirements whereas metal would be heavier and would affect a human DJ's performance more.

6.2 Volume Control

Unlike a human DJ using a mixer, the system currently has no control over the output volume of the turntable. This control could be introduced in two different ways.

A new mechanical system could be built that physically controls a cross fader on a mixer as a human would. The advantages of this method are similar to those of the mechanical system built for the turntable. The mechanical design allows a user to control the volume while the system controls the record, control the record while the system controls the volume, or any other combination. It also would allow the system to read the position of the cross fader while the user was DJing with it, further stealing his style. One of the major foreseeable drawbacks of this method is that, unlike turntables, there is no single standard mixer. Many different brands and models of mixers with different cross faders are used by DJ's. Thus, the mechanical design would have to either be flexible enough to fit many different mixers or would have to be restricted to use with a certain type of mixer.

The alternative design would involve a software volume control. Currently, the software only receives audio from the turntable when it's looking for input. Instead, all the audio coming from the turntable could first be passed through the software system. The system would know what was sent as audio for processing and analysis and what audio was sent to be played. It could then adjust the playback volume appropriately for the audio being played. The advantage of this method is that software control is not limited by a physical system. It would have faster and more accurate control over the volume than a cross fader. A software control would also allow independent control over two or more inputs, whereas a physical control would only allow cross fading between two inputs.

6.3 Beat Detection

Currently the beat detection algorithm can only detect beats in music with a simple quarter note count beat. Unfortunately, most beats that DJ's scratch over are break beats and do not have the simple quarter note beat. Thus, in order to be competitive, the system will have to be able to detect the beats in break beat loops. Prof. Edward Large of the Florida Atlantic University has developed some such algorithms using non-linear oscillators.

6.4 Beat Matching

This was originally a goal for the DJ system and parts of it were implemented, but as mentioned previously, it was discovered that FMOD could not supply the necessary functionality required for beat matching.

6.4.1 Beat Matching Problem Outline

The original specifications were as follows. The robotic DJ system needs to first determine the BPM's (beats per minute) of both the recorded sample, and the record which it is playing. It should first record a 45 second sample from any sound source, and determine the beats' locations in the audio sample. Then, given a 5-10 second sample from a record, it should be able to dynamically match the beats of the record to those of the recorded sample. That is, while playing back the pre-recorded sample, it should drive the record such that the beats from the record play at the same time as those in the pre-recorded sample. It should use the audio input from the record player to first match the beats from the record player to those of the recorded sample. Then, for the duration of the pre-recorded sample, the system should dynamically check that the record's beats are matched and correct the record's playback if they are not. Both the record and the sound sample for pre-recording should have simple steady beats, with four beats per measure. The pre-recorded sample can have portions when the beat drops out or is not well defined. However, the majority of the sample and the entirety of the record must have simple steady beats.

6.4.2 Beat Matching Solutions

The original plan was to use a beat detection algorithm (the same one that is currently used in the performance interface section), and pick the beats out of the 45 second pre-recorded sample. As with the performance interface, these beats would be stored in an array of booleans with each entry representing a 25 ms. interval. Then the system would use the same algorithm on the 5-10 second sample of the record that it had recorded. By comparing the relative speeds of the beats for the pre-recorded sample and the record's sample the initial speed at which the record should be played could be determined.

The system also needs to use this 5-10 second sample to determine the profile of a beat. After using using the beat tracking algorithm to determine where the beats are, the program would use another pattern matching routine to determine the distinguishing characteristics in the 50-200Hz range that identify the beats in on the record.

Once the initial speed and beat profile have been determined, the system would need to cue the record up to a beat. This would be done by using the beat profile and spinning the record at normal play speed until the output in the low 50-200Hz range from the record player matched that of the beat profile.

Once the beat had been cued, the system could start playing the pre-recorded sample at normal speed and, as soon as a beat was encountered in the pre-recorded sample, start to play the record at the already determined start speed. If the speeds of the music were properly determined, the beats of the two audio sources should be together and, at worst, slowly slip apart from each other. To compensate for this, a dynamic speed feedback loop could be implemented. Using the beat profile, the system could identify the beats in the audio input from the record player. If the beats coming off the record player were behind the beats being played back in the pre-recorded sample, the system could drive the record a bit faster and vice versa. Thus, because the beats should be relatively close in the first place, the adjustments will be very minor and the slight discrepancy between the beats necessary for correction, should not be audible.

Furthermore, the system could also incorporate a non-linear speed change for correcting the record's speed if the beats are not falling together. Because the detection of the failing in the beat matching requires the beats to be slightly off, perfectly matching the speed of the beats at that time would result in the beats being permanently the same amount apart. Thus, the perfect correction must involve both perfectly matching the speeds and a one time correction to match the beats. However, the one time correction must be spread out over a period of time or the record will jump quickly forward or backward and cause a scratch like sound. After using a linear correction to reach a point at which the beats are within the same 20ms (the sample period of FMOD) period, the system could use the following algorithm. Every eight beats the system should check the difference between the beats from the two audio samples. As the beat detection is comtimes inconsistent, the check is only made every 8 beats. A longer check time increases the signal to noise ratio, but a shorter allows for faster correction of beats that are off. Because FMOD has a sample period of 25ms, the difference will only be accurate to within 25ms. The difference could be used to calculate the necessary speed change to obtain the matching speed for the record player:

$NewSpeed = OldSpeed * (1 + BeatDifference/(8 * BeatPeriod))$

where BeatDifference is positive if the pre-recorded sample is ahead and BeatDifference is in the same units as BeatPeriod. This works because the difference in the current speed and the matching speed created a gap of BeatDifference over a period of 8 * BeatPeriod. Thus, multiplying the OldSpeed by that factor should result in the matching speed.

The one time correction, which would be applied for one second, could be calculated as follows:

$TempSpeed = NewSpeed * (1 + BeatDifference)$

where BeatDifference is in seconds. This works because the BeatDifference is amortized over one second, so NewSpeed must be increased by an amount that will make up BeatDifference over the one second. TempSpeed is the speed at which to drive the record for the first second of each eight beat period, and NewSpeed is the speed at which to drive the record for the rest of period. The process should repeat at the beginning of each 8 beat period.

6.4.3 Failed Attempt at Beat Matching

The attempt to achieve this goal using the FMOD sound library failed because FMOD does not have the ability to analyze audio input in real time. FMOD cannot take the Fourier transform of an input signal. The typical way to avoid the problem suggested by the FMOD author is to play back the sample as it's being recorded and take the Fourier transform of the sample as it's being played back. However, in FMOD, the playback and record pointers don't always move at consistent speeds. Thus, when playing a sample back that is being recorded, if the playback pointer gets too close to the record pointer it can skip ahead of the record pointer which will cause the playback to fail. In FMOD, the way to get around this, is to constantly monitor the proximity of the playback pointer to the record pointer and slightly alter the playback frequency to keep the playback pointer right behind the record pointer. However, it was also discovered that this technique requires a slight delay, about 250ms, in the beginning of the sample to allow the recording pointer to move ahead of the playback pointer. It was also determined that the playback pointer had to stay 100ms behind the record pointer in order to consistently avoid overrunning. Thus, the DJ system would consistently set the record 350ms ahead, because it would play the beat from the pre-recorded sample and match it up with its analysis of the input from the record, which would be 350ms behind what the record was actually playing. As beats occur every 400ms in a typical piece of electronic dance music, these delays are unacceptable.

Additionally, if the pre-recorded sample is being played with the FMOD library while the system uses FMOD to adjust the playing of the record, the two output channels will both undergo the analysis together as one channel. FMOD does not provide functionality to individually analyze the channels.

The author of FMOD, Brett Paterson, was contacted about adding the necessary funtcionality to FMOD, but no response was received. Other sound libraries were not investigated because of the reformulation of the problem as part of the performance interface.

6.5 Phrase Matching

Another important function that DJ's perform is called phrase matching. Phrase matching involves aligning the first beats of each measure and phrase, in addition to the beats themselves. In other words, because electronic dance music typically has measures of four beats, and phrases of four or eight measures, DJ's don't only match the beats of two songs, but they match the measures and phrases so that the down beat of each measure is played at the same time for each audio sample, and the first beat of each phrase is also played at the same time. This would require the robotic DJ to be able to distinguish a down beat from any other beat. However, if such were possible, the system would only have to cue the record up to the first beat of a phrase. Then, as with beat matching, it would begin to play the pre-recorded sample and start the record as soon as the first beat of a phrase was encountered in the sample.

6.6 Scratch Selection

Currently, the DJ system relies on the user to select which scratches to play when. However, another extension might include audience input similar to Dave Cliff's record selector. The robotic DJ could sense the crowd's excitement it was generating through it's scratches or other techniques and use a genetic algorithm to select music and scratch techniques that would keep the crowd happy. The robotic DJ could also incorporate many of the other techniques that human DJ's use such as volume controls and the use of different records into this selection process. This is not as useful for a competition type environment, because there isn't enough time to "work the crowd", but it would certainly be useful for a longer performance or for creating a routine for a competition.

6.7 Scratching By Ear

Scratching by ear was in the original scope of the thesis, but for simplicity it was never implemented. The robotic DJ system would receive audio input of a scratch as well as the sample that was scratched. Using the original unscratched sample as a guide, the robotic DJ would recreate the motion used for the scratch. The methodology is very much like that of tune mimicking. Analysis of the Fourier Transform of the original sample versus that of the scratch can quickly tell what the relative speed of the record is at any given point in the scratch.

Unfortunately, scratching by ear is not this simple. The DJ also changes the direction of the record frequently. Fortunately, at each direction change, the record stops briefly, generating no output. The robotic DJ system could find each of these places and analyze the output following the break to determine whether the DJ simply stopped the record and continued in the same direction, or reversed the direction of the record. This can be done by comparing the frequency analysis of the next fractions of a second of the scratch with those of the two possible directions from the sample. Only one of the two possible directions will be a shifted version of the original sample. The robotic DJ system can then continue, confident of its direction.

Additionally, DJ's use a cross fader while scratching. The fader affects the volume of the sample being scratched and is often set to zero so that the DJ can cut back and forth to a specific place on the record without generating any output at all. The use of the fader greatly complicates the problem. For one, a quick change of the fader from the normal position to zero and back to normal could easily be misinterpreted as a change in direction.

Because of the ambiguity introduced by the cross fader, it may not be possible to solve this problem by a simple linear analysis. A tree of possible resolutions branching at any point where the volume is zero (which could either be a result of a change in direction or the cross fader) might be a useful tool. At each such point, the system could create a branch for each possible resolution of the point. It could choose one of the possible resolutions and continue down the tree until it reached a point where no possible resolutions could be made or the entire scratch was resolved.

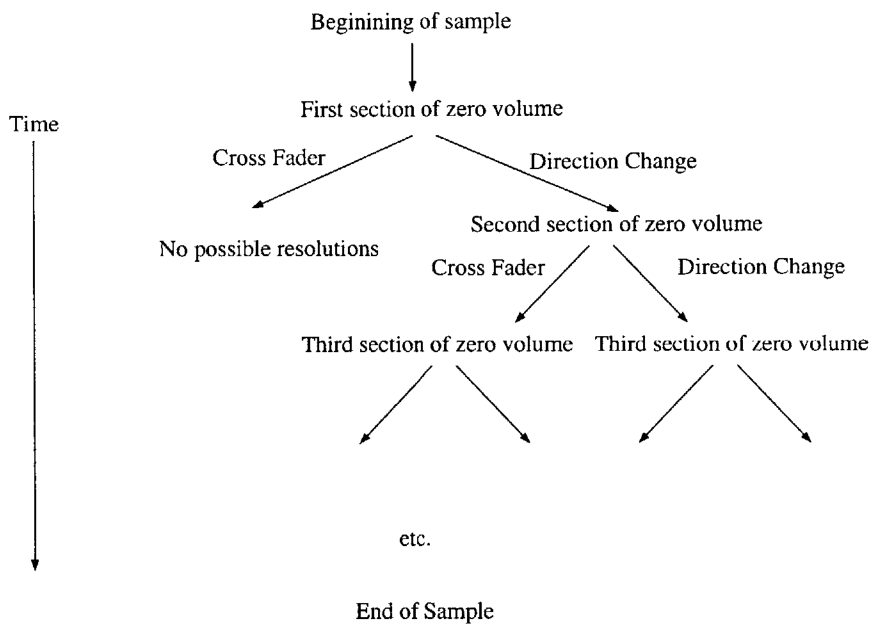


Figure 6-1: An Example Scratch Resolution Tree

There do exist some scratches whose original motions could never exactly be recreated. For instance, if the DJ scratches a constant sine, square, or triangle tone, there's no way to tell from the audio input whether the record is traveling forwards or backwards and so there's no way to tell whether the DJ originally moved the record backwards or forwards. If the signal were more complex, it may be possible to determine its direction.

6.8 Beat Juggling

Beat juggling is an important competition skill practiced by many human DJ's, but not addressed by this thesis. Beat juggling is when the DJ takes two copies of the same record and cues them up to the same point. He then plays a sample from the first (typically a bass kick or other beat sample) and then switches to the second turntable and plays the same sample from the second. While the sample is playing on the second turntable, he resets the first turntable and plays the sample again as soon as the second turntable is finished playing. By playing the same sample, typically a beat section, over and over on alternating turntables, the DJ can create his own beats and loops.

Using a second instance of the system and separate volume controls for each of the two systems, it would be fairly straight forward to include functionality for beat juggling. As mentioned previously, FMOD is not able to provide real time playback of an audio input, so it would not be able to provide the individual volume controls, but with another sound library or a physical mixer control, the task would be feasible. The user could pick any scratch made with style stealer, tone mime, or scratch by ear, and the system could alternate playing the scratch on each turntable and then reseting that turntable while the scratch was playing on the other. The scratches could simply be a few seconds of 33 1/3 RPM play, which is what beat juggling would typically use, but it could also use any other type of scratch.

Chapter 7

Future

While the robotic DJ system does include a lot of functionality, there are many aspects of it that need extensive work before it will be ready to perform. It has no way to generate any scratches on its own and the functionality presented to the user is not flexible enough to allow the user to create a scratch composition in the way a human could. However, based on the success of the mechanical system and the progress made in analysis of audio input and translation of such input into meaningful scratch techniques, it is believed that it will be possible for the system to be able to perform at the national competition level.

Chapter 8

Citations

[1] "All That Scratching is Making Me Biaaatch," http://www.bombhiphop.com/biatch.htm

[2] "The Basics of DJing," http://www.djbasics.com

[3] "Computer DJ Uses Biofeedback to Pick Tracks,"
http://www.newscientist.com/news/news.jsp?id=ns99991563

[4] "DJ, I Robot," http://www.dj-i-robot.com

[5] "DJ Turntablist," http://grove.ufl.edu/ hhc/dj

[6] "Dr. Bop's Home of DJing," http://www.drbop.co.uk/Djing_bigBreak.htm

[7] "Eliza, Computer Therapist," http://www.manifestation.com/neurotoys/eliza.php3

[8] "'Eliza' is Online but it's Better to Download,"
http://www.cumberlink.com/ONTHENET/1999/otn.02.14.html

[9] "FinalScratch - Play Digital the Analogue Way," http://www.finalscratch.com

[10] "The History of House," http://music.hyperreal.org/library/history_of_house.html

[11] "The History of Turntablism," http://www.scratchdj.com/history.shtml

[12] Large, E., and Kolen, J.F. (1994). "Resonance and the perception of musical meter," Connection Science 6, 177-208.

[13] "Luminescence," http://users.sourceforge.net/projects/luminescence

[14] "Man Versus Machine" http://whyfiles.org/040chess/

[15] "The Phonograph," http://www.digitalcentury.com

[16] Scheirer, Eric D. (1998). "Tempo and beat analysis of acoustic musical signals," Journal of Acoustical Society of America 103, 588-62.

[17] "Scratch," http://www.scratchmovie.com

[18] "ScratchRobot," http://www.scratchrobot.com