IMPROVISATION BASED ON IMITATING HUMAN PLAYERS BY A ROBOTIC ACOUSTIC MUSICAL DEVICE ROMI BUILT AS A SELF PLAYING COMPOUND ACOUSTIC MUSICAL ROBOT

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

KUBİLAY KAAN AYDIN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

JANUARY 2013

Approval of the thesis:

IMPROVISATION BASED ON IMITATING HUMAN PLAYERS BY A ROBOTIC ACOUSTIC MUSICAL DEVICE ROMI BUILT AS A SELF PLAYING COMPOUND ACOUSTIC MUSICAL ROBOT

submitted by **KUBİLAY KAAN AYDIN** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _________________
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Ismet Erkmen _________________
Head of Department, **Electrical and Electronics Engineering**

Prof. Dr. Aydan Erkmen _________________
Supervisor, **Electrical and Electronics Eng. Dept., METU**

Examining Committee Members:

Prof. Dr. Semih Bilgen _________________
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Aydan Erkmen _________________
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Veysel Gazi _________________
Electrical and Electronics Engineering Dept., Istanbul Kemerburgaz Uni.

Assoc. Prof. Dr. Afşar Saranlı _________________
Electrical and Electronics Engineering Dept., METU

Assoc. Prof. Dr. Erol Şahin _________________
Computer Engineering Dept., METU

Date: ______24.1.2013________

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name :

Signature :

ABSTRACT

IMPROVISATION BASED ON IMITATING HUMAN PLAYERS BY A ROBOTIC ACOUSTIC MUSICAL DEVICE ROMI BUILT AS A SELF PLAYING COMPOUND ACOUSTIC MUSICAL ROBOT

Aydın, Kubilay Kaan
Ph.D., Department of Electrical and Electronics Engineering
Supervisor : Prof. Dr. Aydan Erkmen

January 2013, 71 pages

In this thesis we introduce the robotic device ROMI together with its control architecture, the musical state representation and focus on parameter estimation for imitation of duo players by ROMI. ROMI is aimed at jointly playing two instruments that belong to two different classes and improvises while assisting others in an orchestral performance. A new improvisation algorithm based on parameter estimation process for the imitation control is introduced. This new improvisation algorithm adds the capability of velocity control and polyphonic music processing on top of existing improvisation algorithms. The musical state representation we have developed for controlling ROMI is a feature extractor for learning to imitate human players in a duet. ROMI's intelligent control architecture also has the ability to provide sample owners' identification and performance training.

Keywords: Control, Imitation, Improvisation, Musical State Representation, Signal Processing

ÖZ

INSAN MUZİK ENSTRÜMANI ÇALMA UYGULAMLARINI TEMEL ALAN EMPROVİZASYON YETENEĞİNE SAHİP ROBOTİK AKUSTİK MÜZİKAL CİHAZIN KENDİ KENDİSİNİ ÇALABİLEN BİRLEŞİK AKUSTİK ROBOT ROMI OLARAK YAPILMASI

Aydın, Kubilay Kaan
Doktora, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Aydan Erkmen

Ocak 2013, 71 sayfa

Bu tez çalışmasında bir robotik akustik müzik cihazı olan “ROMI” ni için imitsayon kontrol değişkenlerinin otomatik yakınsama işlemine dayalı yeni bir doğaçlama algoritması sunulmaktadır. ROMI’nin kontrolü için geliştirmiş olduğumuz müzikal durum gösterimi, bir düet halinde çalan insan müzisyenleri benzeten bir özellik ayrıştırıcısıdır. ROMI’nin akıllı kontrol mimarisi aynı zamanda müzisyeni tanıma ve çalma performansını geliştirme kabiliyetine de sahiptir. Bu tez çalışmasında robotik cihaz ROMI’yi kontrol mimarisi, müzikal durum gösterimi ile sunuyor ve düet yapan müzisyenlerin benzetimi için kontrol değişkenlerinin otomatik yakınsaması üzerine odaklanıyoruz. ROMI’nin amacı iki farklı müzikal alet sınıfından müzik aletini bir arada çalabilir ve doğaçlama yapabilirken bir orkestraya eşlik edebilmesidir.

Anahtar Kelimeler: Kontrol, Imitasyon, Doğaçlama, Müzikal Durum Gösterim, Sinyal İşleme

ACKNOWLEDGMENTS

The author wishes to express his deepest gratitude to his supervisor Prof. Dr. Aydan Erkmen and to Prof. Dr. Erol Kocaoğlan for their guidance, advice, encouragement and patience throughout the research.

The author would also like to thank Assoc. Prof. Dr. Afşar Saranlı and Assoc. Prof. Dr. Erol Şahin for their suggestions and comments.

The author wishes to thank his father Cengiz Aydın for his help in building the acoustic musical parts used in this study.

The author wishes to thank his uncle Dr. Tümay Tümer for his constant encouragement to finish his doctoral studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

MOTIVATION

In the past years, researchers have made mechanical music playing devices to investigate the mechanism of generating sounds under various conditions. The primary study of musical engineering began with the development of musician robot WABOT-2 in 1984 by Waseda University as an evolution of WABOT-1. WABOT-2 was able to play music with a concert organ. One year later, another musician robot, WASUBOT, performed a concerto with NHK symphony Orchestra. In 1989, the University of Electro-Communication in Japan developed MUBOT to function as an interface between the musical instrument and a human being. It played a violin or cello automatically.

Current research linking musical composition and computer science has traditionally focused on the performance through virtual musical instruments, such as within the area of electronic music synthesis using modern digital signal processing techniques. Synthesized sound, however, achieves only limited accuracy and expressive translation from actual performance gestures due to errors. For example, when converting violin string vibration into MIDI pitch and velocity data. Music performed by robots on real instrument offers the audience live experience very similar to listening to a human musician. The cognitive foundations and realism within real instrument performance, such as the physical vibration of a violin string, provides a much stronger music expressiveness than in the case of electronic music synthesizers. By controlling the parameters involved in music expressions through computer controlled mechanical entities, robot musicians are designed to bypass several technical difficulties that are typically encountered in human musicians, such as intonation, repetitive gestures, and complex articulation patterns, as well as speed and clarity of musical events. A robotic musician, potentially, could have more degrees of freedom in real-time performances and reach a much higher level of performance difficulty, flexibility and quality in terms of specific and idiomatic demands. As an example, one can imagine a violin being played by a robot musician with hands that have 12 fingers, generating a 12 polyphonic sound that human violinists are not capable of.

Besides surpassing human capabilities, biomimetic has also been another focus. Among these works, robotic imitation of human players many different computational methods have been developed. Research on imitation of playing musical instruments and reproducing acoustic music have been investigated from many different perspectives: employing mirror neuron activity in a artificial recurrent neural network, as a pattern recognition problem, as a hierarchical modular approach, as a supervised learning approach and as a fixed mapping of human sensorimotor actions.

Biomimetism has been advanced by adding improvisation capability on top of the existing imitation systems; grammars for automated improvisations have been developed, while rule based algorithms for music improvisation have modeled the structure of an artist's style with a set of rules. Genetic algorithms using evolutionary operations (mutation, selection, and inheritance) have been frequently used, artificial neural networks have been trained to generate melodies in certain styles of a single musician. And Markov Chains of different complexity have been successfully employed.

Robots imitating humans have generated a field that has acquired a decent maturity while new robotic or computer based electronic musical instruments can go beyond imitation, into the domain of improvisation. Computer synthesized music is the main research area for improvisation due to its controlled environment. Imitation of human players for playing acoustic musical instruments is one research area in robotic imitation. There are very few works trying to merge the robotic imitation of human acoustic musical instrument playing, with improvisation generated by computer synthesizing.

The present thesis work addresses this area by designing and implementing a robotic device, which is itself an acoustic musical instrument with a control architecture capable of improvisation. Therefore the motivation in our study is to achieve musical improvisation by an acoustic instrument playing robot. Our contribution to the existing knowledge on improvisation is the addition of polyphony and loudness measures into the improvisation algorithms.

OBJECTIVE AND GOALS

The objective in our work is to develop a **RO**botic acoustic **M**usical **I**nstrument, named ROMI, that will be jointly playing at least two instruments while assisting others in an orchestral performance and also learn to imitate other players using the same instruments. This intelligence in imitation also provides sample owners' identification and performance training: there the system identifies a certain player based on the sequence of errors. In performance training the system can guide the performer by his/her own error measures. In this thesis, we introduce the robotic device together with its control architecture involving imitation and improvisation subsystems that are based on our novel Musical State Representation (MSR), and automatic parameter estimation. Instead of using an existing multi-purpose robotic device to play an existing acoustic musical instrument, we targeted to build the robotic parts around two modified acoustic musical instruments. This new acoustic musical instrument is designed not only with acoustic properties in mind but also engineered for robotic control. This way the study can achieve improvisation at the computer control level and result in a new acoustic musical instrument that plays standalone. This new acoustic musical instrument then can join the ranks of many of its kind, with a distinctive property; improvisation capacity.

Building an all new acoustic musical instrument which plays itself by learning from human players and being capable of improvisation is therefore the main focus of our research. In our work, instead of observing teachers who are experts in one acoustic musical instrument playing, we propose to observe groups of teachers playing instruments from two different musical groups, namely strings and percussion.

In order to achieve this goal, the steps in our study were determined as: learning the state of art in robotic musical instrument building and gaining musical background, designing and manufacturing ROMI, developing control algorithms for imitation and improvisation by ROMI, experimenting with the system to correct its problems also addressing its weaknesses, running sensitivity analyses to understand the effects of control parameters on the whole system.

CONTRIBUTION

Our contribution to the existing control algorithms for imitation and especially for improvisation in musical robots, is in two areas: The first novelty is to include loudness or velocity information into the improvisation algorithms. The second one is addressing polyphonic music.

All existing improvisation works with musical robots assume that the music performing system has constant loudness while playing notes. However, the loudness measure is an integral part of musical quality. If a musical piece is played with constant loudness, the music will sound dull and mechanical.

One of the most important qualities that an experienced musician introduces into the musical piece he/she is playing, is the loudness changes in the notes. This information is only implied in a given musical score. Therefore it is an area where the musician is free to introduce his/her interpretation. Although the notes being played are exactly the same with the score and the overall tempo is correct, the freedom in interpreting the loudness changes in a musical piece is some form of improvisation.

Our first contribution to improvisation in computer synthesized music is the integration of loudness information into the imitation and improvisation architecture.

Improvisation in computer synthesized music use algorithms that are designed to work with monophonic sound. The reason for this approach is to test the improvisation algorithms in a controlled environment. Some of the existing works on improvisation in computer synthesized music give a clue on how improvisation algorithm can be extended to polyphonic music. However, for most of the existing work on improvisation in computer synthesized music, it is not clear if it is possible or how it is possible that the existing algorithms in the literature would work with polyphonic sound.

Music is in general polyphonic. Even if a single musical instrument is being played, still, the number of notes that are performed simultaneously is usually more than one. However, polyphony is a complex issue to work on when it comes to computer synthesized music. Imitation of a polyphonic system has many control issues to deal with which has been addressed successfully in existing works. However, improvisation has been investigated only in a monophonic environment for acoustic musical robots. There are very few existing works on polyphonic music improvisation on synthesized music and none on acoustic musical robots.

Chapter 2 provides the literature survey, that enhances the gap that exists before this thesis, in improvisation with polyphony.

Our second contribution to improvisation in computer synthesized music is the integration of polyphonic sound processing capability into the imitation and improvisation architecture. The frequency range of ROMI is 110-440Hz. and it is 12 polyphonic. ROMI is at the same technological level with its modern day counterparts.

ROMI on its own is another contribution to existing robotic designs that are acoustic musical instruments themselves. ROMI adds two more acoustic musical instruments to the list of robotic musical instrument designs; tubular bells and clavichord. The acoustic sound has been fine tuned to be chromatic which enables ROMI to participate in musical performances with other of robotic musical instruments or with human players.

ROMI will be introduced together with its control architecture in the third chapter.

The fourth chapter describes our Musical State Representation (MSR), that is used for controlling ROMI in detail. The imitation and improvisation processes are demonstrated by examples in the same section as well.

The experimental results based on the parameter estimation process will be presented and a sensitivity analysis of imitation parameters is discussed in the fifth chapter.

Chapter six concludes the thesis with clues to possible future work.

CHAPTER 2

LITERATURE SURVEY AND MUSICAL - MATHEMATICAL BACKGROUND

In this chapter the literature survey of existing work is presented first. This section also gives the background information about improvisation in jazz, which is one of a unique music genre where improvisation has some musical structure. The underlying musical and mathematical background is presented in the second section of this chapter. Examples of existing robotic designs that are acoustic musical instruments themselves are introduced in the third section of this chapter.

2.1 Literature Survey

Research in robotic musicianship opens the door for new and interesting methods of creating and performing computer music. The combination of software and mechanics provides not only a unique opportunity but limitation for developers, composers, and performers. Robotic musicians greatly differ from the pure software based computer music applications which bestow an unlimited library of timbre, floating point precision, and extremely low latency. Some of the most distinctive differences prevalent in robotic musicians are their ability to create natural acoustic sound using traditional instruments, their mechanical limitations, encompassing everything from the rate of movement to the mere fact that their extremities occupy physical space. Their ability to provide visual feedback through a much more natural and engaging experience for an interacting human musician is of value too. Many of the studies on imitation and improvisation of musical instrument playing has been facilitated through the use of computer generated music [11] and various new electric musical instruments, including a wearable one a PDA based one and a graphical one have been proposed [12]-[14].

2.1.1 Literature Survey on Improvisation

2.1.1.1 Grammars in Jazz Improvisation

Automated jazz improvisation for which grammar have been developed is based on concept of terminals. Terminals contain duration and one of several categories of notes that are relevant to jazz playing. Each production rule has a probability, allowing different terminal strings to be produced each time.

Improvisation in music has some form of a structure in jazz. Therefore, we investigated in details jazz structures in order to learn the systematics being brought to improvisation in this genre of music. Jazz encompasses a wide range of categories in a constantly evolving landscape of music. Since people have varying notions of what defines jazz, when it originated, and what styles it grew out of, strict definitions tend to be controversial. Some characteristics, however, bridge the gap between eras and represent what the average person might picture when imagining jazz music. Two such distinguishing elements are improvisation and a swing feeling. Swinging basically consists of playing a steady beat over syncopated rhythms, as opposed to classical music, where the tempo is less strict and the rhythms adhere more to the beat. The swing feeling in jazz is in large part defined by the swing eighth note pattern, in which rather than playing two notes of equal duration, the first note is held for twice as long as the second, in effect dividing beats into two thirds and one third instead of equally in half. The swing feeling sets the stage for improvisation, a form of composition done concurrently with performance. While the rest of the band maintains a steady pulse, a soloist, or lead player, can play complex lines and freely diverge from the melody of the song. Although improvisation is largely spontaneous, players do practice ideas prior to performance: most players have a vocabulary of licks (short phrases) that they draw upon when constructing solos. Jazz bands traditionally start by playing the melody ("the head") of a song, and then lead players to take solos in turns while the rhythm section (usually drums, bass, and piano or guitar) accompanies. Since the late 1960s, modern jazz has merged with popular music and branched into a wide range of styles such as smooth jazz, nu jazz, and acid jazz, which stray in varying degrees from their roots. Music played in these styles today is sometimes known as "straight ahead jazz." Any jazz musician will tell you that the essence of jazz is in the subtleties with which it is performed. Timbre, inflection, and slight rhythmic nuances that make playing expressive where human factors are rarely noted and sometimes overlooked. Although in this thesis, we do not attempt to capture all of the expressiveness and spirit that make music "jazzy," it is important to consider the context and manner in which the melodies we examine are played.

Grammars for automated jazz improvisation in which non-terminals can have a counter associated with them to indicate when to stop expanding has been developed. The key idea in this system is manifested in the terminals. The terminal duplets containing duration are one of several categories of notes that are relevant to jazz playing. The counter denotes a number of beats, so to generate a solo over 32 beats, the start symbol receives parameter 32. Given a start symbol P, expansions are of the form (P 32) # (Seg2)(P 30), where (Seg2) eventually expands to a sequence of terminals that will produce two beats of music. Each production rule has a probability, allowing different terminal strings to be produced each time. Categories are defined as follows:
1. Chord tones: tones of the current chord.
2. Color tones: complementary tones for the current chord. These are tones that are not in the chord, but which are individually sonorous with it.
3. Approach tones: non-chord tones that make good transitions to chord-tones or color tones.
4. Other: tones that do not fall into one of the categories above.
The note categories are represented in the grammar with specific symbols, namely:
1. C a chord tone.
2. L a color tone.
3. A an approach tone.
4. X an arbitrary tone.
5. R a rest.
6. H a "helpful" tone, defined to be any one of the above.
7. S a scale tone, a member of a scale designated as being compatible with the chord.

The grammar might generate the sequence (A8 C8 L4), representing an 8th note approach tone, followed by an 8th note chord tone, and finally a quarter note color tone. Applying the terminal strings generated by the grammar to lead sheets, and based on the chords and a few additional constraints such as a bound for maximum interval between notes, they determine the final melody [4].

2.1.1.2 Rule Based Algorithms

The idea of using a rule based algorithm for music improvisation is to model the structure of an artist's style with a set of rules, taken characteristics from music theory, the programmer's own preferences, or a corpus of data. These rules apply to particular conditions and can sometimes be represented by a set of conditional probabilities with a Markov Chain. Rules such as those in the transition matrix would produce melodies that tend to sound musical but lack direction. These matrices deal only with harmony, so they are used in conjunction with another matrix for specifying rhythm or expanding their context to combine rhythm and interval into a single state, yielding matrix categories of the form: (duration, interval). Such simple sets of rules have been shown to generate interesting, if not musical, pieces [5].

2.1.1.3 Genetic Algorithms

Genetic algorithms, inspired by the principles of natural selection, are generally used as heuristics in optimization or search problems. Given a large space of possibilities, genetic algorithms start with a set of initial solutions, and through various evolutionary operations such as mutation, selection, and inheritance, develop new generations of solutions iteratively until some convergence criteria are met. Genetic algorithms consist of several basic steps:

1. Initialize a population of individual solutions.
2. Determine the fitness of each individual in the population.
3. Choose a set of the fittest individuals to be parents.
4. Breed new individuals.
5. Initialize next generation with new individuals.
6. Repeat steps 2-5 until the population reaches an acceptable fitness level.

The method to determine fitness of individuals, called a fitness function, usually requires human input. In some cases, the optimization is done on the fitness function itself, so that after sufficient evolution performed on the fitness function, parse the program can evolve its population without further input. GenJam, is a genetic algorithm that learns to improvise jazz. In many problem-solving scenarios, composers and improvisers devote much of their efforts to looking for a melody or chord that sounds right. The population in GenJam consists of measuring melody lengths, together with phrases, which are made up of four measures. Melodies are represented by "chromosomes" of bits, and modifying the chromosomes yields variations of the melodies. During the program's training phase, a human listens to a piece and rates as she listens. Each rating applies to both a phrase and one of the measures in the phrase, and the set of ratings then determine which melodies will be selected as parents [6].

Given a large space of possibilities, in the form of existing musical parts, genetic algorithms use evolutionary operations such as mutation, selection, and inheritance to develop new generations of solutions iteratively. Those solutions are improvisations that meet some convergence criteria [7]. Improvisation which is a difficult topic in robotic imitation has been investigated in the well defined musical improvisation domain of jazz [8]-[10].

2.1.1.4 Neural Networks

Artificial Neural Networks are systems inspired by neuron connections in the brain. Components in a neural network have weighted connections to one another that can be strengthened or weakened based on training and experience. Given a set of possible solutions to a problem, neural networks can come up with similar solutions by triggering random initial output and then repeatedly adjusting connections until the output sufficiently resembles the input. Neural networks can learn unsupervised, using only the training data, or with supervision, where a user gives grades to output, which in turn affect the connections of the network [17]. CONCERT is an Artificial Neural Network trained to generate melodies in the style of Bach. CONCERT looks to learn information about note-to-note transitions as well as higher-level structure of songs [16]. This approach is particularly successful when the neural network is trained in a single musicians works. This approach to improvisation fails when the training samples belong to different musical genres.

2.1.1.5 Markov Chains

Experiments in Musical Intelligence (EMI), have produced accurate imitations of composers. Programs like EMI, work by taking as input a corpus of works from a composer, analyzing the music, and obtaining from it a set of rules. The key component is recombination. A corpus of 370 Bach chorales, pieces usually consisting of four voices, has been used as a basis for new Bach influenced chorales. The training data is divided into beat-length or measure-length sections and then recombined in a Markovian process by looking for extracted parts that, if placed sequentially, follow the rules of Bach's voice leading. Every extracted section is marked by its first set of notes that sound together as well as the first set of notes that follow the part. Extracted sections are grouped into lexicons, each of whose members has exactly the same set of starting pitches. So, in recombination, a current state's next set of notes points to a particular set of lexicons from which EMI can choose a next state. In this process, which can be described by a bigram Markov chain, EMI only chooses harmonic transitions that have appeared at least once in the training data. Clearly, a sufficiently large set of training data is necessary for successful composition of new pieces. If given more than one choice for the next state, EMI does not choose the actual transition that occurred in the corpus. With too small a data set, often the only possibilities that follow the correct voice leading rules would be the particular sections that appeared in order in one of the chorales, so the program would simply reproduce a piece from the training data. Although these rules ensure logical connectivity between beats or measures, the approach yields pieces lacking in larger scale structure. Higher-order Markov chains are suggested as a vehicle for additional structure that larger N-grams are insufficient, since "In music, what happens in measure 5 may directly influence what happens in measure 55, without necessarily affecting any of the intervening measures." [18]. The key component is recombination. The training data is divided into beat-length or measure-length sections and then recombined in a Markovian process by looking for extracted parts that, if placed sequentially, follow the rules of Bach's voice leading [19].

2.1.1.6 Other Related Work on Improvisation

Some research works have focused on the presence of a single improvisation model which is monophonic and which always performs the same type of fixed length improvisation parts regardless of the length of the original musical piece [20], [21]. A fixed-function mapping based improvisation supporting system has also been proposed in [22]. This approach uses a previously developed table of improvisation parts to be used irrespective to the original musical pieces characteristics.

Rules building the transition matrices are used to produce melodies that tend to sound musical but lack the characteristics of the original musical piece [23]. This work has a rule based algorithm which models the structure of an artist's style with a set of rules based on music theory, the programmer's own preferences, or a corpus of data. These rules apply to particular conditions and can be represented by a set of conditional probabilities.

Gesture based musical accompaniment systems that use a camera to monitor the human teachers facial or hand gestures to control the improvisation parameters have been developed in [24]. Other gesture related input devices like wearable gloves have also been used in various works [25].

Synthetic music improvisation using only simple random variations which are matched to relevant aspects of the original musical piece have also been proposed in [26]. Some have simplified the problem of improvisation to a limited observable behavior of the first 16 notes with a limited response of 4 notes [27]. Fixing the mapping between the original musical piece and the improvisation generated in response has been proposed with very limited improvement in the success [28]. These works have shown that using a random variation or a fixed mapping are not viable solutions to improvisation if the expected musical quality is to match human performances.

2.1.2 Literature Survey on Imitation

Imitation learning has gained considerable interest from the artificial intelligence community over the last years. The work on imitation learning is closely related to that of building a control architecture for an embodied agent. Architectures for imitation learning explicitly state that the agent learns by observing others, producing the same motor actions. Imitation learning can thus be seen as a more specialized branch of research on architectures for motor control and learning.

Research for imitation of playing musical instruments and reproduction of acoustic music has been investigated as a pattern recognition problem in [15]. A hierarchical modular approach where the modules are designated with different tasks have been proposed in [31]. The low-level modules are responsible for short-term critical tasks, such as following a given tempo, whereas the high-level modules implement goal-directed behavior such as imitating a melody. The low-level modules can override the high-level behaviors, since they represent fundamental behaviors. The designer of the system determines how the modules are able to override each other, as well as the capabilities of each module. Behaviors using competence modules have been investigated in [32]. These competence modules are connected and spread activation throughout the network, based on fulfilling certain pre and post conditions that are defined by the designer of the network. Some modules only activate other modules, whereas some issue motor commands. Imitation is achieved by adjusting the pre and post conditions. Another modular approach where both the structure and functionality of each module is predefined is given in [33]. The modules represent the different sensorimotor processing stages, such as perception, recognition and action selection. Similar to this architecture is the work of Mataric in [34], which proposes an architecture for learning by imitation that explicitly deals with visual processing and motor behavior in separate modules.

Supervised learning is dependent on a teacher signal, and Jordan and Rumelhart showed in [35], how a supervised approach could be used when there is no explicit teacher signal, only consequences of the actions. By using an internal model of the consequences of the actions, the forward model can be used to transform the error signals in sensory space to error signals of the motor space. This allows for particular solutions to be found in the action space, as opposed to direct inverse modeling where the input/output relationship of the environment is reversed to train the inverse model - this is not necessarily a correct inverse model, since there are many ways to achieve a desired goal. This work forms the basis for Wolpert's model [36]. Similarly, Demiris and Hayes employ an architecture with paired inverse and forward models in [37]. What separates these architectures from Jordan and Rumelhart is that they use multiple paired inverse and forward models to imitate motor actions. The inverse models learn behaviors, and the forward models predict the outcome of these behaviors. Based on these predictions, the best suited inverse model for the task is chosen. The approach of using multiple paired inverse/forward models differ from the approaches of Cangelosi, and Riga given in [38], since the latter uses inhibition as a control mechanism among modules, whereas a multiple paired inverse/forward approach coordinates which inverse model is best suited to control the robot based on the performance of the forward models.

Employing mirror neuron activity in a recurrent neural network for imitation is given in [39]. Special neurons self-organize to specific activations both when producing and observing an action. It is also possible to manually activate the parametric bias neurons, and the network will then generate the corresponding activity. A neural network to ground symbols from actions; the student learns actions and corresponding symbols by observing the teacher is given in [40]. The robots play, using imitation to ground the symbols of the actions. A student robot imitates a teacher robot, acquiring sensory data and symbols at the same time, storing the information in the network.

It also acquires its playing style by looking at changes in tempo and loudness, using string kernels to form the internal representations [41]. Imitation of a musical piece by first-order logic to represent tempo and dynamics, and then a clustering algorithm to group similar phrases has been investigated in [42]. This representation describes how a pianist would play a classical piece of music. Hidden Markov Models to create an instrument called "The Continuator" that plays with musicians in real time has been developed by Pachet in [43]. The human pianist plays a phrase, and the Continuator plays a second phrase that is a continuation of the first phrase. The Markov Models represent the probability that a note will follow the other, focusing on melodic signature instead of tempo and dynamics. Case-based reasoning to imitate the expressive qualities of how the music should feel (i.e. joyful or sad) and to change the tempo of the song, still maintaining the musical expressiveness has been discussed in [44].

"Music Plus One" is a system that models user-specific variations in tempo when playing a piece of classical music [45]. Classical musicians interpret the score and vary the intensity and tempo of the notes. This is why MIDI1 playback of classical pieces sound very machine-like. This system allows soloists to practice along with the computer, and the system is then able to follow the score of the soloist during changes in tempo and dynamics. This system is used in an evolutionary approach to generate drum tracks based on melodic input. The user of the system selects which rhythm figures should be allowed to evolve, and new rhythm figures are created in an evolutionary manner. The evolved networks vary dynamics as well as beats, but not timing, which is an important part of human expressiveness. The system does not model human expressiveness but tries to mimic a natural feel by introducing variations. The user directs the randomized search towards a desired result. A demonstration on how human drummers are able to play faster than the delays of the neuromuscular system, showing how impedance modulation can be used to overcome limitations of slow actuators has been investigated in [46]. The control system was set up as a lumped-element second order model, using springs and dampers to represent the fingers, drum stick and head. The focus was not on building an intelligent model of the drumming behaviour, but to mimic how humans effectively vary the stiffness of the drum stick by modulating grasp force, increasing the frequency of the drumming.

A theoretical framework for specifying imitation of playing YMCA for robots, based on the idea that movement is built up of dynamic primitives is given in [47] and [48]. These primitives can then be combined to create rhythmic and discrete movements. To demonstrate rhythmic movements, a robot was set to perform a drumming task. The primitives are made up of nonlinear differential equations, and provide a mathematical approach to generating motor control. Related to this work is the research by switching and superposition between discrete and rhythmic movements for robotic imitation is investigated in [49]. This is achieved by using oscillators, movement trajectories are generated on-line for a humanoid robot. The humanoid robot plays the drums according to a score given to the system, and the corresponding arm trajectories are produced from the score using the oscillators. This allows the system to be robust towards perturbations of the trajectories. The Haile drummer [50] created plays the Native American Pow-wow drum. It has both a database of learned patterns, as well as the ability to imitate and modify patterns when interacting with human drummers. The control mechanism consists of using an environment called Max/MSP, which makes it easy to implement beat detection, frequency analysis and other operations necessary for musical interaction and performance. The Haile robot was later extended to play the xylophone, using a genetic algorithm to generate novel melodies while playing.

Synchronizing to a conductor or another drummer to keep tempo has been demonstrated by [51]. It fuses auditory and visual perception to keep time with the conductor and the human drummer. The conductor and drummer can then change the tempo of their drumming, and the robot will change accordingly. The patterns played by the drummer are very simple; one beat for each gesture made by the conductor. However, the generation of complex drum patterns is not the focus of research, it is instead studying how robots can synchronize in social tasks. This work is an extension of the research by Williamson [52], which employs oscillators in a drumming task to exploit the natural dynamics of the robot system when performing rhythmic movement. This adaptation was made possible since the oscillators could make use of feedback sensor signals. The system focuses on how impulse forces can be used to overcome torque limitations of the actuators. The goal is to formulate an impact dynamics that can deal with the various issues that occur when large impulse forces are used, which are crucial for stability and security of robot systems. Trajectories for the arm movements were calculated using cubic splines, and the robot was able to play along to a musical score.

A survey of AI methods in composing music is given in [29] and melody extraction as the basis of improvisation by AI methods has been investigated in [30]. The literature survey that is summarized above showed us that there are some research areas that has not been covered yet. From these possible research areas we decided to focus our research on polyphonic improvisation with velocity control. Results of our work on improvisation based on imitation of human players by a robotic acoustic musical device has been presented in [1], [2]. The MSR that is used in controlling the imitation process has been presented in [3].

2.2 Musical and Mathematical Background

The industry standard for computer synthesized music representation is called “MIDI”. The notes are represented in their absolute values in MIDI files. Our system that will be clearly exposed in Chapter 3, use MIDI files in the early preprocessing stage of imitation. These files are transformed into our musical state representation for further processing.

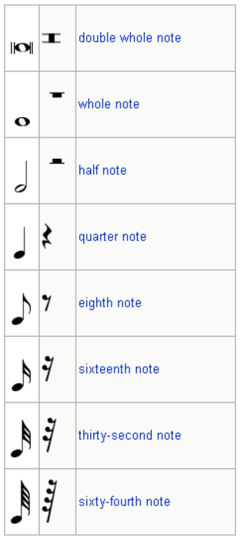
Our Musical State Representation (MSR) is different from MIDI in the sense that in MSR only note and loudness difference (delta) values are stored in a discrete time model. This choice is due to the fact that it suits our improvisation needs better than the MIDI representation. In fact, these two representations can be converted to each other. To convert from MIDI to our MSR it is enough to calculate the differences between consecutive note and loudness values. To convert from our MSR to MIDI, all that is needed is the addition of the starting note and loudness values such that the absolute note and loudness values are then calculated from the delta values for each consecutive note.

If the goal was to playback music, then perhaps the absolute value stream (MIDI files) would have been a better candidate. However, when dealing with improvisation, the absolute note values are of little help. Both melody and rhythm are directly in conjunction with the delta values of notes and loudness. An average listener that is, one without “absolute pitch”, can differentiate between two consecutive note differences easily. However, an average listener can not tell the difference between two performances of the same melody if one is played one note higher or lower than the other. This clearly shows that the human ear is usually more sensitive to note changes rather than note values.

From most important to least important; note duration, loudness, silence, relative note pitch and starting note has been considered in our musical state representation design.

Note duration is usually denoted as 1, ½, ¼ of a whole note, but the whole note duration is not an absolute value. It depends on the tempo of the musical part, which may even be altered within a singe musical part.

The tempo of a part is typically written at the start of a musical notation and in modern music is usually indicated in beats per minute (BPM). This means that a particular note value (for example, a quarter note or crotchet) is specified as the beat and the marking indicates that a certain number of these beats must be played per minute. The greater the tempo, the larger the number of beats that must be played in a minute. Mathematical tempo markings of this kind became increasingly popular during the first half of the 19th century, after the metronome had been invented by Johann Nepomuk Mälzel, although early metronomes were somewhat inconsistent. Beethoven was the first composer to use the metronome. We use 120 BPM default for ROMI with user adjustment. So each quarter note lasts 0.5 seconds. The most widely used denotations for note length values are shown in Figure 1 below:



𝅜	𝄺	double whole note
𝅝	𝄻	whole note
𝅗𝅥	𝄼	half note
♩	𝄽	quarter note
♪	𝄾	eighth note
𝅘𝅥𝅯	𝄿	sixteenth note
𝅘𝅥𝅰	𝅀	thirty-second note
𝅘𝅥𝅱	𝅁	sixty-fourth note

Fig 1. Denotations for note length values

The importance of note duration (or tempo) is apparent in musical nomenclature. No special names or attributes have been given to note values but as shown by the following classification of tempo, this has emotional results on human listeners.

Allegro — fast or "march tempo" (120–168 bpm)
Allegro moderato — moderately quick (112–124 bpm)
Andante — at a walking pace (76–108 bpm)
Largamente — very, very, very slow (10 bpm)
Largo — very, very slow (30–45 bpm)
Lento — very slow (40–60 bpm)
Moderato — moderately (108–120 bpm)
Prestissimo — extremely fast (more than 200bpm)
Presto — very fast (168–200 bpm)

Another example of the lesser importance of note values is as follows: Two notes with fundamental frequencies in a ratio of any power of two (e.g. half, twice, or four times) are perceived as very similar. Because of that, all notes with these kinds of relations can be grouped under the same pitch class. In traditional music theory pitch classes are represented by the first seven letters of the Latin alphabet (A, B, C, D, E, F and G). The eighth note, or octave, is given the same name as the

first, but has double its frequency. The name octave is also used to indicate the span of notes having a frequency ratio of two. To differentiate two notes that have the same pitch class but fall into different octaves, the system of scientific pitch notation combines a letter name with an Arabic numeral designating a specific octave. For example, the now-standard tuning pitch for most Western music, 440 Hz, is named a' or A4 (la). We intend to use 440 Hz, 220Hz and 110Hz A notes for ROMI.
The notes as used in western music producers is denoted in Figure 2 below:

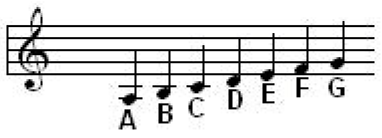
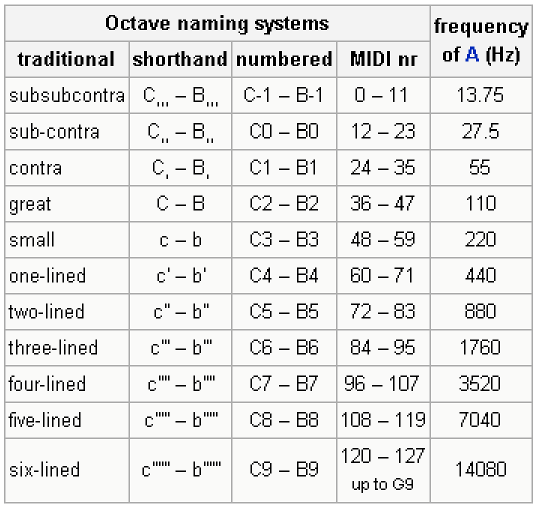


Fig 2. Denotations for notes in western music

Table 1. Frequency ranges for notes



Octave naming systems				frequency of A (Hz)
traditional	shorthand	numbered	MIDI nr	
subsubcontra	C,,, – B,,,	C-1 – B-1	0 – 11	13.75
sub-contra	C,, – B,,	C0 – B0	12 – 23	27.5
contra	C, – B,	C1 – B1	24 – 35	55
great	C – B	C2 – B2	36 – 47	110
small	c – b	C3 – B3	48 – 59	220
one-lined	c' – b'	C4 – B4	60 – 71	440
two-lined	c" – b"	C5 – B5	72 – 83	880
three-lined	c"' – b"'	C6 – B6	84 – 95	1760
four-lined	c"" – b""	C7 – B7	96 – 107	3520
five-lined	c""' – b""'	C8 – B8	108 – 119	7040
six-lined	c""" – b"""	C9 – B9	120 – 127 up to G9	14080

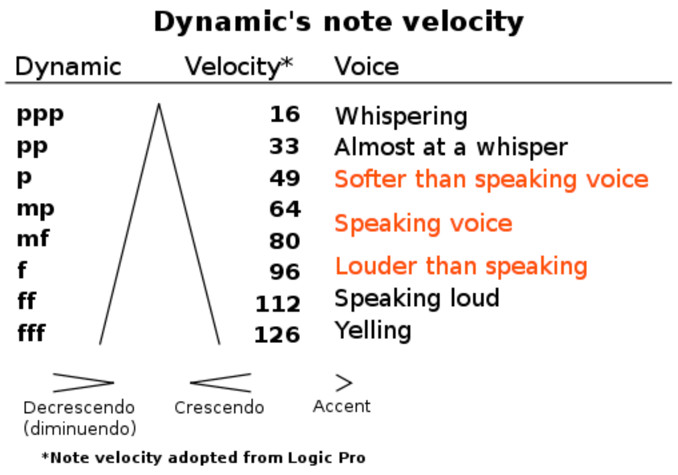
The frequency ranges are given in Table 1 above: Loudness (or velocity) of a note is more apparent to a human listener than the note values. The following classification of loudness is used by classical western music producers.

The two basic dynamic indications in music are:

p or piano, meaning "soft."
f or forte, meaning "loud" or "strong".
More subtle degrees of loudness or softness are indicated by:
mp, standing for mezzo-piano, meaning "moderately soft" and
m*f*, standing for mezzo-forte, meaning "moderately loud".
Beyond f and p, there are also
ff, standing for "fortissimo", and meaning "very loud",
pp, standing for "pianissimo", and meaning "very soft",

To indicate an even softer dynamic than pianissimo, ppp is marked, with the reading pianississimo ("very, very soft") or pianissimo possibile ("softest possible"). Each additional "p" adds another "iss" to the word. The same is done on the loud side of the scale, with *fff* being "fortississimo" ("very, very loud").

Table 2. Logic Pro's dB values with regard to whispering noise [55]



Dynamic's note velocity

Dynamic	Velocity*	Voice
ppp	16	Whispering
pp	33	Almost at a whisper
p	49	Softer than speaking voice
mp	64	Speaking voice
mf	80	
f	96	Louder than speaking
ff	112	Speaking loud
fff	126	Yelling

>	<	>
Decrescendo (diminuendo)	Crescendo	Accent

*Note velocity adopted from Logic Pro

Loudness is represented by "velocity" numbers in digital music sequencers. These numbers are dB values with regard to whispering noise. The explanation in Table 2 shows one such number scale for a specific sequencer called Logic Pro [55].

2.3 Existing Works on Musical Robots

From a practical perspective, building a robot is an enormous investment of time, engineering, money, and effort. Maintaining these systems can equally be a costly and time-consuming process. Furthermore, to design and manufacture all of these systems to operate in real time requires an enormous dedication to building computational architectures and optimized software. Constructing the perceptual, motor, and cognitive skills that are necessary to begin to address the specific problems of robot imitation is extremely difficult. In the literature many researchers have opted to introduce robotic parts to musical setup in order to minimize the cost of building a whole music playing robot. Figures 3, 4 and 5 show examples of robotic parts added to existing musical instruments [53]. To address the improvisation needs for electronic music composing many different approaches has been applied with varying success. The measure of success for evaluating improvisation is subjective and most of these studies use human listener groups used to evaluate improvisation performances success.

Each robot musician given in Figures 3, 4 and 5 adopt a three-module architecture, which consists of the following vertically arranged functional elements: a software module, an actuation module and a motion module. First, the software module interacts with users, provides the programming and composition environment and sends motion commands to the control module through a serial port. In the next step, the control module involves rules that govern application processing and connects the software module with the motion module. The control module is optimized and synchronized with the fast and repeatedly-used equation solving routines. Finally, the motion module - the hands of the robot musician, is provided by powered mechanics, namely, servos and solenoids, which manage and provide access to the actual musical instrument. The software module is implemented as a set of Turbo C++ programs running on the DOS operating system. DOS has provided a robust environment for delivering continuous control commands to the robot musician. The software communicates with the control module through serial port I/O. At the current stage of development, the software consists of two major components: a composition component and a performance component. The solenoid and servo control in real time is a paramount consideration, especially in the real time use of the instruments. By using a computer keyboard to control the motors, one can actually improvise, and capture a spontaneous musical gesture. Such potential has theatrical implications. These improvisations can be recorded and saved as files, which can be played back with precision. A wonderful feature of this composition component is the ability to edit the file, essentially allowing for the "tweaking" of sequential detail. Such an option is yet another level of control in the musical compositional process. First, the composition component provides users with a direct programming environment. Based on a set of pre-defined syntax, users can create C++ programs which control each detailed motion of the robot musician, such as the bow manipulation of a string instrument and the drumstick or wand operation of a percussion instrument [53].

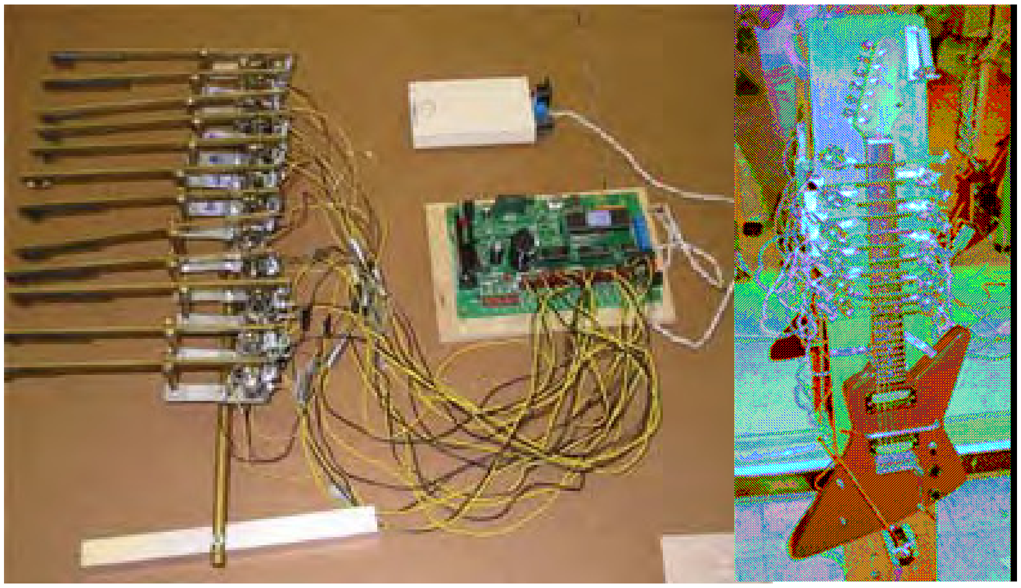


Fig 3. Musical instruments with added robotic parts (Dusty II)

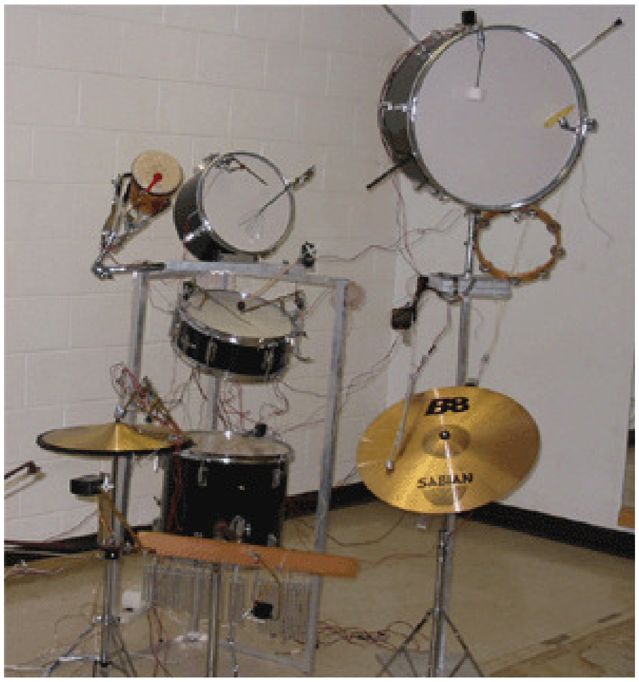


Fig 4. Musical instruments with added robotic parts (Micky)

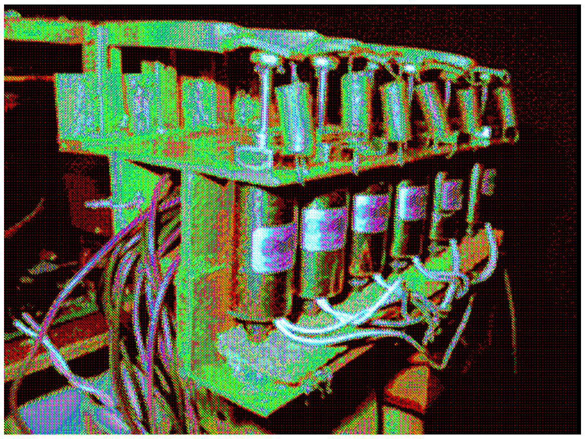


Fig 5. Robotic designs that are acoustic musical instruments themselves (Jasche)

Figures 6 and 7 show examples of existing robotic designs. These are acoustic musical instruments themselves called Expressive Machines Musical Instruments [54]. Featuring 15 strikers of various types (hard plastic, rubber, felt, brushes) situated at various positions from the center to the rim of the drum, as well as a mechanism to switch the snares on and off, MADI is designed to maximize the timbral potential of a single snare drum. Using multiple solenoid-driven striking arms simplifies the design and minimizes constraints and latency. Individual arms are capable of rolling at audio rates (upwards of 60 Hz), and multiple arms can strike simultaneously. MADI was constructed August-September 2008. CADI consists of a modular system of solenoid-driven strikers that can be configured to beat on any object. They can be mounted on microphone stands for flexibility of placement. CADI is designed to work equally well with traditional percussion instruments as well as found objects. CADI was constructed September-October 2009. These robots can receive data from many sources, musical/gestural input devices (e.g. MIDI keyboard/joystick, mouse, etc.), environmental sensors (e.g. photoresistors), composed or algorithmically generated sequences [54]. Messages are sent from a computer via USB to an Arduino microcontroller, used to switch solenoids on and off. Architecturally, the control module consists of an RSV Converter and a custom manufactured motion control card. The motion control card is built on an Aesthetika Robotika Servo Control Rev 4 board, powered by 12V and 500 mA. The original board could support up to 16 servos, 16 solenoids and 4 steppers. It is custom manufactured for this project to support up to 4 servos (4 axes) and 12 solenoids. The converter interfaces the serial port and motion control card. The speed and position servo loop adjustment are optimized according to the mechanical load characteristics in order to achieve a stable and fast response. For the case of a robot musician, the control module receives a list of motion control commands from the software module and computes joint angles and servo motor velocities for each axis. These values are then delivered to the motion module, ensuring smooth servo motor movements [54].

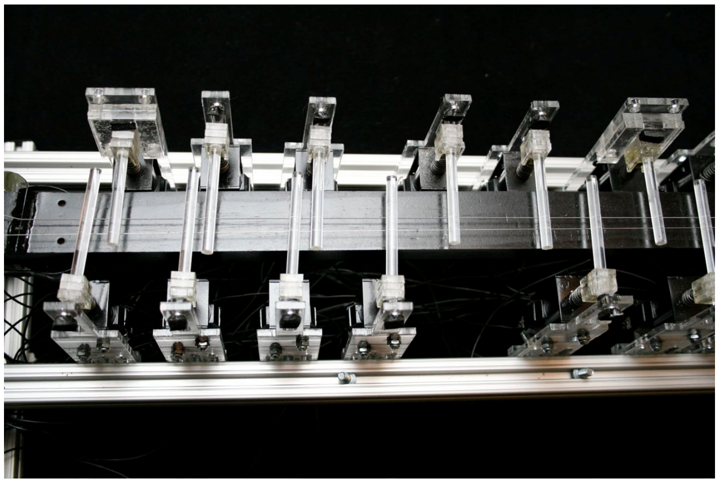


Fig 6. Robotic designs that are acoustic musical instruments themselves (CADI)

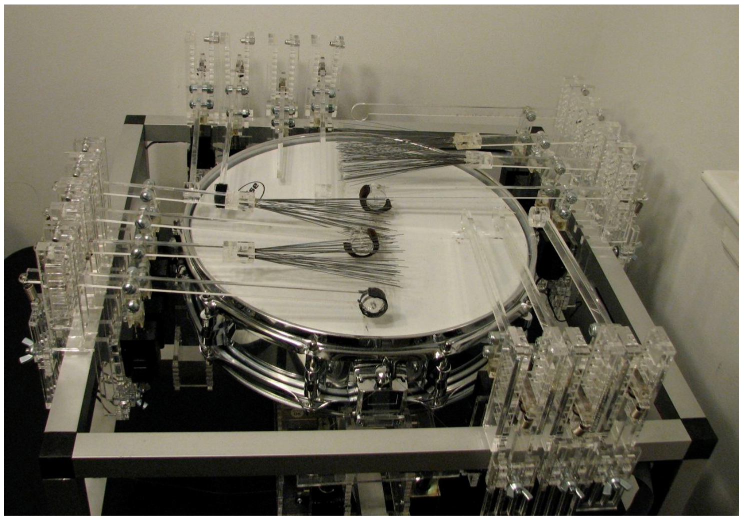


Fig 7. Robotic designs that are acoustic musical instruments themselves (MADY)

CHAPTER 3

ARCHITECTURE OF ROMI

3.1 ROMI Acoustic Subsystems

Two acoustic musical instruments from two different domains have been selected. Clavichord is the father of piano like string instruments where sound is generated by hitting the string with a needle (clavichord) or hammer (piano). It can be seen as a horizontal harp with the addition of a keyboard. This is an easier to build instrument. The second domain is percussion, where we chose tubular bells. This choice is made since tubular bells is one of the few instruments that has a chromatic sound. This means that a note based sound from tubular bells is possible which is not possible for example for drums. General classification of these instruments have been given below.

- 1 Idiophones (1)
  - 1.1 Struck idiophones (11)
    - 1.1.1 Directly struck idiophones (111)
    - 1.1.2 Indirectly struck idiophones (112)
  - 1.2 Plucked idiophones (12)
    - 1.2.1 In the form of a frame (121)
    - 1.2.2 In the form of a comb (122)
  - 1.3 Friction idiophones (13)
    - 1.3.1 Friction sticks (131)
    - 1.3.2 Friction plaques (132)
    - 1.3.3 Friction vessels (133)
  - 1.4 Blown idiophones (14)
    - 1.4.1 Blown sticks (141)
    - 1.4.2 Blown plaques (142)
  - 1.5 Unclassified idiophones (15)
- 2 Membranophones (2)
  - 2.1 Struck membranophones (21)
    - 2.1.1 Directly struck membranophones (211)
    - 2.1.2 Shaken membranophones (212)
  - 2.2 Plucked membranophones (22)
  - 2.3 Friction membranophones (23)
    - 2.3.1 Friction drums with stick (231)
    - 2.3.2 Friction drum with cord (232)
    - 2.3.3 Hand friction drums (233)
  - 2.4 Singing membranes (kazoos) (24)
    - 2.4.1 Free kazoos (241)
    - 2.4.2 Tube or vessel-kazoos (242)
  - 2.5 Unclassified membranophones (25)
- 3 Chordophones (3)
  - 3.1 Simple chordophones or zithers (31)
    - 3.1.1 Bar or stick zithers (311)
    - 3.1.2 Tube zithers (312)
    - 3.1.3 Raft zithers (313)

        - 3.1.4 Board zithers (314)
        - 3.1.5 Trough zithers (315)
        - 3.1.6 Frame zithers (316)
    - 3.2 Composite chordophones (32)
        - 3.2.1 Lutes (321)
        - 3.2.2 Harps (322)
        - 3.2.3 Harp lutes (323)
    - 3.3 Unclassified chordophones (33)
- 4 Aerophones (4)
    - 4.1 Free aerophones (41)
        - 4.1.1 Displacement free aerophones (411)
        - 4.1.2 Interruptive free aerophones (412)
        - 4.1.3 Plosive aerophones (413)
    - 4.2 Non-free aerophones (wind instruments proper) (42)
        - 4.2.1 Edge-blown aerophones or flutes (421)
        - 4.2.2 Reed aerophones (422)
        - 4.2.3 Trumpets (423)
    - 4.3 Unclassified aerophones (43)
- 5 Electrophones (5)

The classification is the most widely used Hornbostel-Sachs classification, this is also an open ended classification where all new instruments like ROMI can be added to the world classification list. Figures 8 and 9, show examples of both instruments.

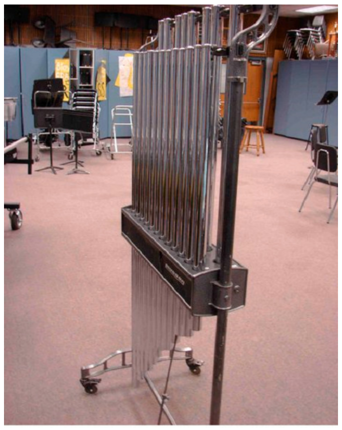


Fig 8. Tubular bells, Hornbostel-Sachs classification: 111.23

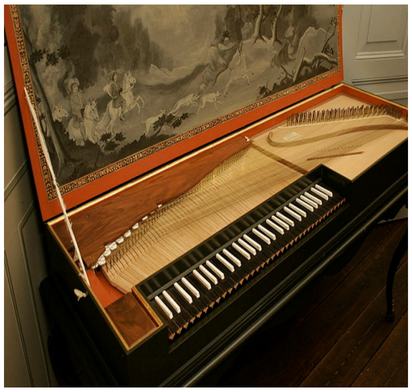


Fig 9. Clavichord, Hornbostel-Sachs classification: 314.122-4-8

3.2 Infrastructure of ROMI

ROMI consists of two separate body parts. The tubular bells part and the clavichord part have been separately developed. Both parts are fine tuned to produce chromatic sound. This is useful when participating in a joint performance with human players or other musical robots. ROMI covers a voice frequency range between 110-440Hz which means that it is 3 octaves wide in the musical sense. The solenoids that are used are not current controlled so a direct velocity control is not intended. ROMI is at the same technological level with its modern day counterparts, which also do not employ current controlled solenoids. The shortest note length of ROMI is determined by the response time of the solenoids, which is approximately 2ms. This means that ROMI can play up to 1/32th note lengths for a 120bpm tempo. This speed is also comparable with other existing musical robots. The pendulum effect, that will be introduced later in this chapter, restricts the shortest note length of the tubular bells section. ROMI can play 1/32th note lengths for a 120bpm tempo on the clavichord section and 1/8th note lengths for a 120bpm tempo

The pictures for the test setup of ROMI for tubular bells and clavichord sections are given in Figure 10.

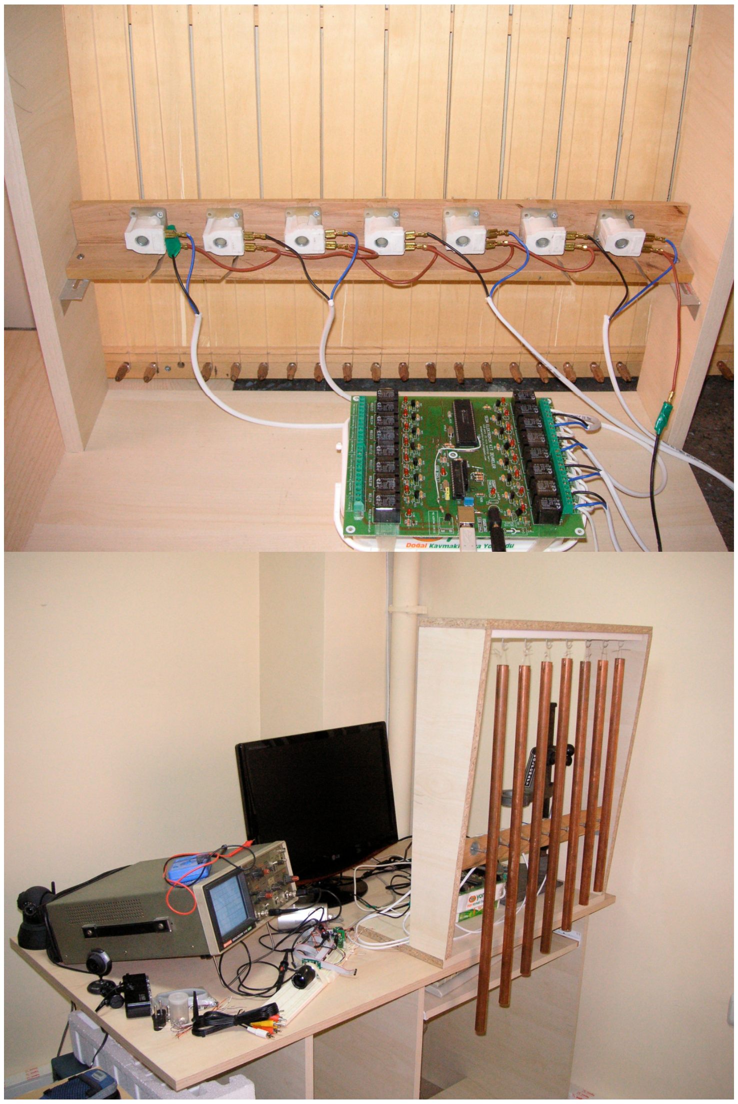


Fig 10. ROMI tubular bells and clavichord sections test setup

A commercially available relay control card has been used as shown in Figure 11. This card can be connected to a PC via the USB port. The choice for using the USB communication port over RS232 serial communications was due to its wider availability and higher communication speed. The USB 2.0 standard was chosen over USB 1.0 or 1.1 due to its higher communication speed.

The relay card can be programmed to control the 220VAC, 7A relays. The choice for AC relays over DC relays is due to the use of AC solenoids. The choice of AC solenoids over DC solenoids is due to their higher power rating. At 3A maximum current rating the used AC solenoids can give a power output of 600W peak. This power is needed to generate acoustic sound. The relays control the current of the solenoids in an ON/OFF configuration. The block diagram of this card is given in Figure 12. The resultant control can simulate ON/OFF note commands of a sequencer. In order to control the currents of the solenoids, such that the impact force of the solenoids on the strings of the clavichord or tubular bells is variable, requires one D/A converter for each solenoid and a current adjustable circuit for driving the solenoids. To keep the system simple another approach has been taken to control the impact force; namely pulse width adjustment.

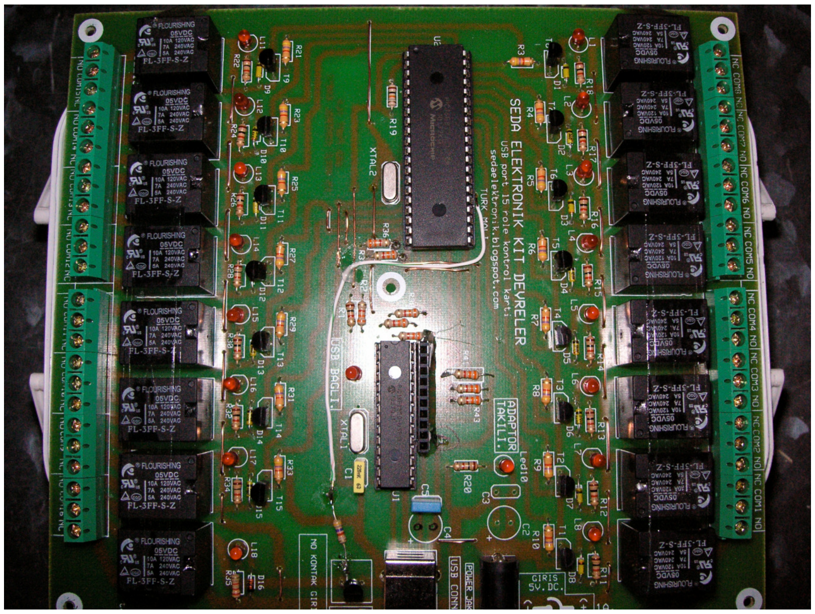


Fig 11. Picture of relay control card

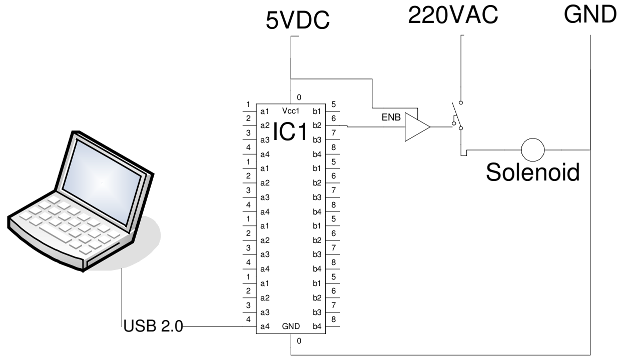


Fig 12. Block diagram of relay control card

Due to the lack of a D/A converter for each solenoid, velocity control is implemented to some extend, with a very simple approach using 5 different pulse widths. The relay card can apply a current pulse on a given solenoid effectively changing the velocity of the played note. However, this can be done only at 5 distinct levels. The resultant simple velocity control is not part of the control architecture of ROMI and is implemented on the relay card only. This applies to both the clavichord and the tubular bells section. Metal bars are pushed to hit the tubular bells while metal needles with plastic tips is used to slide by the side of the strings of the clavichord section. The limited velocity control has the same effect on both parts.

Figure 13, shows how the pulse width is calculated. If the time for a solenoid to hit its associated tubular bell at maximum force exertion is denoted by t_{hmax}, then any pulse width t_l supplied to the solenoid which is smaller than t_{h100} , will result in a sound in lower loudness. If a pulse supplied to the solenoid has a wider width $(t+1)_l$ than t_l it will have a higher loudness. The maximum loudness that can be achieved is t_{hmax} which means that the solenoid fully impacts its associated target while it is at rest. The width of the pulse can be used with limited precision to implement velocity control of a note.

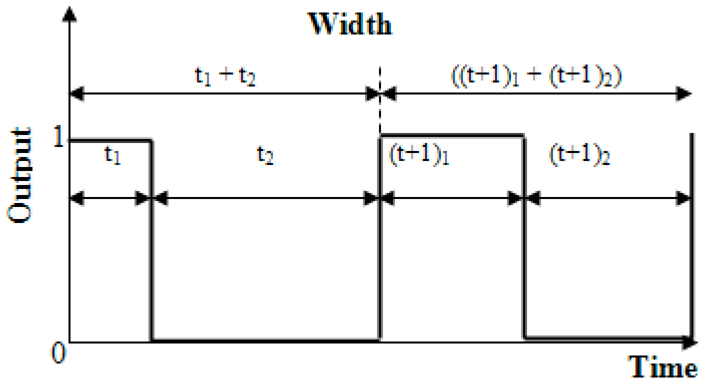


Fig 13. Pulse width adjustment calculation

Any pulse width larger than t_{h100} has the potential of causing an unwanted secondary sound on the target, therefore t_{hmax} must be smaller than $t_l + t_2$ which is the minimum time duration between two consecutive notes played on the same target.

3.3 Clavichord Section of ROMI

ROMI utilizes a two octave string section with “A” frequencies of 440 and 220 Hz. Figure 14, shows the clavichord section of ROMI. The clavichord section uses needles with plastic tips to slide by the strings of the clavichord to generate an acoustic sound. However, the acoustic sound that is generated is not loud enough to be used together with the tubular bells section. To increase the sound volume and to add vibrato to the sound of the clavichord strings a wooden enclosure acting as a acoustic chamber has been designed and built. The strings are wound at the external part of the acoustic chamber while the rack that carries the solenoids is mounted inside the acoustic chamber. This design helped masking the sound generated by the operation of the solenoids. The noise level dropped by -0.2db after the utilization of the acoustic chamber. The acoustic sound level is increased by +0.4db, but it is still not in par with the tubular belles section. Therefore, microphones are employed to amplify the sound. The microphones are placed inside the acoustic chamber close to the middle section. Trials showed that bandbass filters with very sharp and narrow frequency responses are necessary to avoid crosstalk between these microphones. The acoustic chamber that is designed and built for the clavichord section accommodate the microphones, the rack carrying the solenoids and adds vibrato to the sound of the clavichord section.

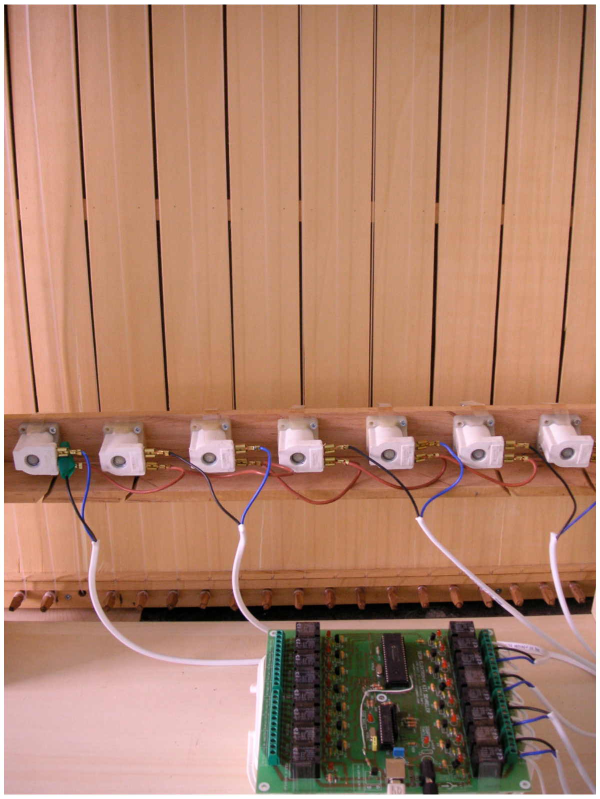


Fig 14. ROMI clavichord section

3.4 Tubular Bells Section of ROMI

The tubular bells section of ROMI utilizes one octave percussion section with “A” frequency of 110 Hz. Figure 15, shows the tubular bells section of ROMI. The tubular bells section has an acceptable acoustic sound level, therefore no amplification is used for this section. The copper tubes used are sensitive to temperature changes, so the tuning of this section is guaranteed only at room temperatures. There is no way to tune the copper tubes for differences in ambient temperature since their frequency response is a function of their geometry. The tubular bells section is chromatic for a temperature range of 22-26C.

There is a noise due to the operation of the relays and solenoids which is inaudible when the control card with the relays is placed in a sound proof box.. The noise level dropped by -0.9db after the utilization of the sound proof box.

The operation of tubular bells presented an unpredicted problem: swinging. Once a solenoid hits a tubular bell, a momentum is induced to the tubular bell which is proportional to the bells weight. This is shown in Figure 16.

This swinging motion is a pendulum like, if the solenoid is very well aligned with the center of the tubular bell. If not, then the motion is not one dimensional which further complicates the problem. Our setup for the tubular bells allows us to individually adjust the location of the tubular bells with respect to the solenoids. We align the tubular bells using this setup, such that the resultant motion can be modeled as a single dimensional pendulum with negligible deviations in a second dimension.

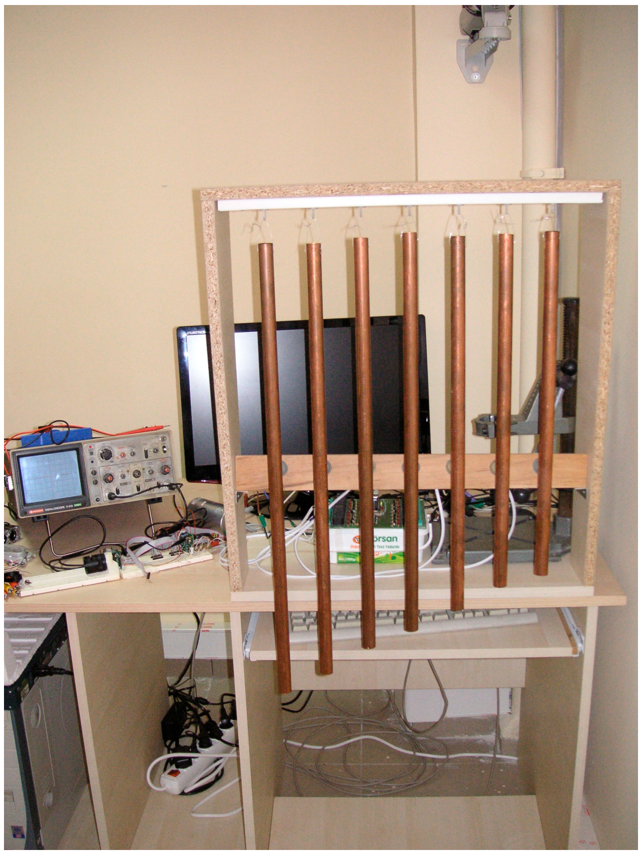


Fig 15. ROMI tubular bells section

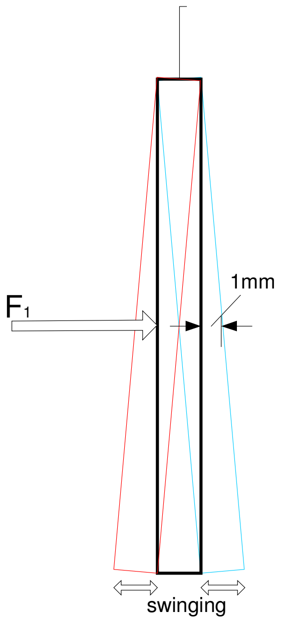


Fig 16. Pendulum motion of tubular bells

After a tubular bell is hit by the solenoid the swinging motion is found to fade away in a finite time. However if a second note on the same tubular bell is to be played before the pendulum swinging motion has become negligible, than the solenoid results in hitting a location other than the standard location which is that of the tubular bell when it is in rest.

The problem is that, since the solenoid will hit a swinging tubular bell slightly before or after the intended time, the velocity control implemented by adjusting the pulse width will become unpredictable.

As an example, if the tubular bell for note A in our setup is hit in the middle by our solenoids as it is at rest, it will swing at a maximum of about 1mm at the center. The swinging motion is reduced to a minimum by adjusting the rack carrying such that the solenoids to hit the tubular bell close to the hinge it is connected to.

Our tubular bells setup accommodates for 7 notes. If a musical piece is not in the C key but at another key, the setup can be modified manually by exchanging the tubular bells with bells tuned for other notes. At the moment we have additional tubular bells for notes A and B at lower pitch. If four more bells are added to this collection, all musical pieces can be played regardless of their key. However if musical pieces from different keys are to be played in a single session, a manual change of the bells is required.

3.5 Abilities and Limitations of ROMI

One octave of tubular bells and two octaves for the clavichord are enough to play almost all existing musical pieces. The sound coverage is 3 octaves and the frequency range is from 110Hz to 440Hz. Limited velocity control, in the sense that only 5 distinct velocity levels can be applied to the clavichord strings or the tubular bells, has been implemented at the relay card level. ROMI is 12 polyphonic, which means it can play up to 12 notes simultaneously. The hardware architecture is modular. More relay cards can be connected via a second USB 2.0 interface and up to 32 solenoids can be controlled. Since the system uses 220VAC, it can be easily plugged into any mains socket for power. The control laptop, relay cards and solenoids, tubular bells and clavichord sections are separate modules. Therefore the system can be upgraded in parts.

The clavichord section looses its accord in prolonged operation. This is due to the fact that the acoustic sound is generated by needles brushing to the strings applying a high tension on them. This causes the strings to loose their tension. A standard set of harp strings is used for the clavichord section. The acoustic chamber is made from 3mm thick pine wood.

The accord of the tubular bells are fixed by their geometry. The tubular bells are from 20mm diameter copper tubes with a thickness of 2mm, where the length of the tube for A note is 62,3cm. Due to the pendulum effect, the shortest note than can be played on the tubular bells section is a half note for 120bpm tempo.

CHAPTER 4

IMITATION AND IMPROVISATION

4.1 Problem Definition

Our contributions to imitation and improvisation in musical robots, are to include loudness (velocity) information and polyphonic music into the imitation and improvisation algorithms.

The first area requires representation of the loudness information. Therefore, our proposed control architecture must integrate loudness information into the imitation and improvisation architecture. This integration must not degrade the note reproduction quality and must not overload the computation of the control commands. This is a critical requirement since the imitation process is followed by a reproduction of the imitated acoustic music generated by the instrument parts of ROMI. Moreover, the improvisation process must generate notes within the octave range of ROMI and must sound coherent with the imitation parts. Our proposed control architecture must be able to process up to 12 polyphonic sound without loosing coherence between the monophonic parts and must not overload the control computation. Furthermore, our proposed control architecture must be able to perform $1/32^{th}$ notes for 120bpm tempo. Under such requirements we now introduce in detail the control architecture, built upon imitation and improvisation capabilities for ROMI.

4.2 Feature Extraction: Generating the Musical State Representation

First a brief explanation of the control architecture given in Figure 17, will be given, all topics will be explained in detail further down in this chapter.

In ROMI's control architecture two sets of musical signals are separately processed, one for the clavichord and the other process for tubular bells. The processing of these signals is never mixed in any of the application blocks.

There are two separate and parallel paths of data processing in the control architecture of ROMI. These two paths are identical for their data processing stages and capabilities but they use two disjoint sets of MSR's and two disjoint data collections for their operation. One of these paths process the data for the clavichord and the other for tubular bells. The data flow and imitation parameters for each of these paths are disjoint as well. In learning mode, the human teachers play either the clavichord or the tubular bell in an acoustically noise free environment The human teacher must indicate to the control system which instrument is going to be played prior to the sound recording. Depending on this selection the control system initiates the respective data processing path.

These sound samples are recorded by a microphone and further isolated from possible background noise by the application of a 0-50Hz low pass filter for the tubular bells and a 200-800Hz band pass filter for the clavichord and stored as a sound signal in WAV format. Then a commercial software is used to extract the musical notes in the sound signal, the result is an industrial standard file called MIDI where music is represented as note ON and OFF note commands.

The MIDI file is then processed by our “Feature Extraction” stage and all recorded samples are stored in a “Sample Collection”. The feature extraction process converts the MIDI files into our “Musical State Representation” (MSR) that is introduced in the next section. From this point on, all musical data is represented as three number streams $\boldsymbol{N}$, $\boldsymbol{M}$ and $\boldsymbol{A}$.

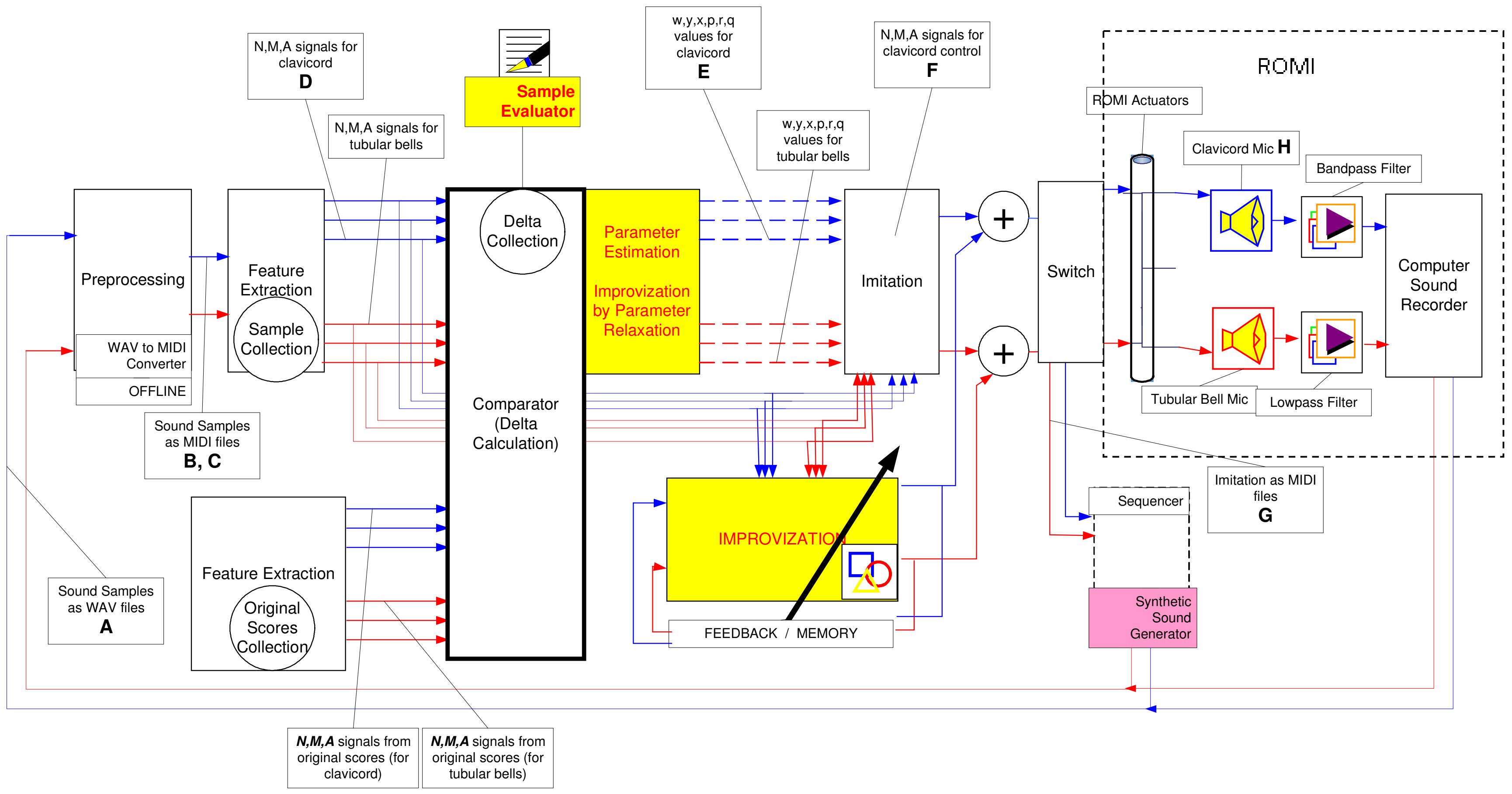

N,M,A signals for clavicord D
Sample Evaluator
w,y,x,p,r,q values for clavicord E
N,M,A signals for clavicord control F
ROMI
ROMI Actuators
N,M,A signals for tubular bells
w,y,x,p,r,q values for tubular bells
Clavicord Mic H
Bandpass Filter
Preprocessing
Feature Extraction
Sample Collection
Delta Collection
Parameter Estimation
Improvization by Parameter Relaxation
Imitation
+
+
Switch
Computer Sound Recorder
WAV to MIDI Converter
OFFLINE
Comparator (Delta Calculation)
Tubular Bell Mic
Lowpass Filter
Sound Samples as MIDI files B, C
Imitation as MIDI files G
Sequencer
IMPROVIZATION
Feature Extraction
Original Scores Collection
Sound Samples as WAV files A
Synthetic Sound Generator
FEEDBACK / MEMORY
N,M,A signals from original scores (for clavicord)
N,M,A signals from original scores (for tubular bells)

Fig 17. Control architecture of ROMI, showing 8 signal points **A** to **H**

The original scores that are readily available in MIDI format are used to construct the “Original Scores Collection”. These MIDI files are also converted into our MSR and is stored in the “Original Scores Collection” for easy processing at further stages of the control system. This data is then used for player identification and parameter estimation. All data at these stages: the sample being processed, the Samples Collection being learned by ROMI in previous sessions and the Original Scores Collection are converted into our MSR format.

The sample being processed is compared with the corresponding original score at the “Comparator” and a “Delta” vector is calculated as the distance of the sample from the original score. All delta vectors are stored in the “Delta Collection”, therefore the system not only stores the MSR for each sample but also stores the delta vector for each sample as well. This information is utilized by the “Parameter Estimation” process to estimate the six imitation parameters w, y, x, p, r, q.

Sound is reproduced by ROMI and this reproduction is fedback to the system via microphones and control is achieved by minimizing the difference between the musical information stored in the MSR and the music generated.

In the control architecture of ROMI, there are 8 points in Figure 17 denoted with large capital letters from A to H, which will be explained in detail below.

Point A: At the start of the process the sound files are either coming from the sequencers voice export function or as recorded sound files by the computer after being conditioned by filters. These files contain the uncompressed, uncoded raw sound signal in the form of a WAV file, which is one of the industry standards for sound files. The sound samples are recorded by a microphone and further isolated from possible background noise by the application of a low pass filter for the tubular bells and a band pass filter for the clavichord and stored as a sound signal in WAV format.

Point B: A commercial software is used to convert the raw sound information into ON/OFF note commands. This information is then stored in industry standard MIDI files. It is possible to get the MIDI file out of the sequencer directly, but not from recorded samples. Sample sound recordings have been collected from numerous musicians playing a piano. These recordings are then utilized for the development of the imitation algorithms after they are converted to MIDI format by this commercial software. We developed a software converter which converts these MIDI representations, which are incomprehensible to the human eye, into recognizable note sequences. This enabled us to gain insight to the musical notes being played.

Point C: We have written a small code to convert MIDI files into readable files. An example output of our code after it processes a MIDI file is given in Figure 18. Here the file is readable in the sense that we can read the ON/OFF note commands.

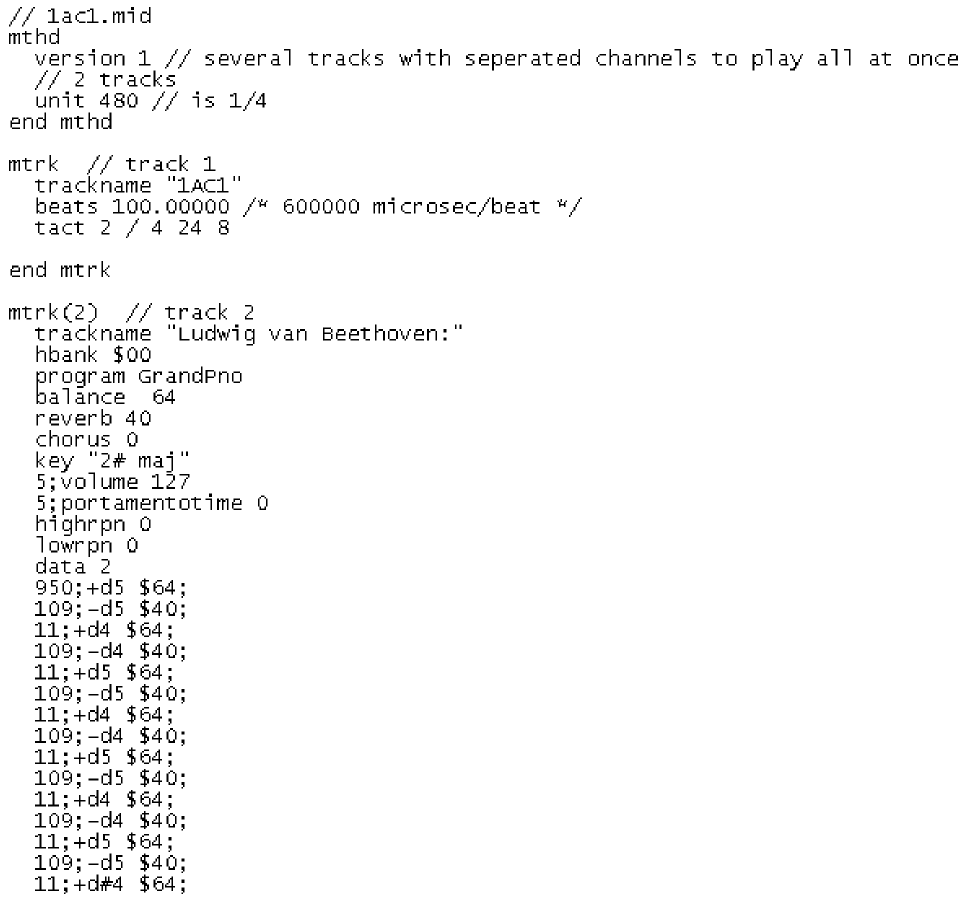
Point D: The feature extraction process changes the absolute note values found in the MIDI file as ON/OFF note commands into our MSR. All calculations are done on these number streams that are kept as arrays in the computers’ storage. Further examples will be presented in this chapter together with algorithms working on these streams. The MIDI file is processed by our “Feature Extraction” stage and all recorded samples are stored in a “Sample Collection”. The feature extraction process converts the MIDI files into our “Musical State Representation” (MSR)

Point E: The imitation control parameters are used by the imitation process. The parameter estimator process tries to minimize the difference between an imitation output and the perfect sample that is found in the original scores collection.

Point F: The imitation process uses the imitation control parameters supplied by the parameter estimator to produce the MSR for the imitation.

Point G: At the switch, if the sequencer is being used to test and train the system then the signal is converted into a MIDI file and routed to the sequencer.

Point H: At the switch, if ROMI is being used to test and train the system then the signal drives the solenoid control card to generate acoustic sounds from ROMI.



```
// 1ac1.mid
mthd
  version 1 // several tracks with seperated channels to play all at once
  // 2 tracks
  unit 480 // is 1/4
end mthd

mtrk  // track 1
  trackname "1AC1"
  beats 100.00000 /* 600000 microsec/beat */
  tact 2 / 4 24 8

end mtrk

mtrk(2)  // track 2
  trackname "Ludwig van Beethoven:"
  hbank $00
  program GrandPno
  balance  64
  reverb 40
  chorus 0
  key "2# maj"
  5;volume 127
  5;portamentotime 0
  highrpn 0
  lowrpn 0
  data 2
  950;+d5 $64;
  109;-d5 $40;
  11;+d4 $64;
  109;-d4 $40;
  11;+d5 $64;
  109;-d5 $40;
  11;+d4 $64;
  109;-d4 $40;
  11;+d5 $64;
  109;-d5 $40;
  11;+d4 $64;
  109;-d4 $40;
  11;+d5 $64;
  109;-d5 $40;
  11;+d#4 $64;
```

Fig 18. Example readable MIDI file after process

4.3 Definition of Imitation Parameters in MSR

The MSR consists of three number streams: stream ***N*** records the relative pitch difference between consecutive notes while stream ***M*** records the relative loudness difference between consecutive notes. Stream ***M*** itself is a record of the duration of all notes. When there is a change in the current note, at least one of the two number streams register this event in the array structure by recording a non zero number. Number stream ***A*** is an event indicator similar to a token state change in a Petri Net, where ***A*** values can assume any rational number. The event indicator ***A*** number stream is important in our improvisation algorithms. The addition of the event indicator ***A*** to the MSR has eased the detection of tempo in musical parts.

In the MSR, time (t) is sectioned into 1/64th note duration slots. The maximum musical part length is set to 1 minute in our application for simplicity. This gives 1920 slots of time for each musical part (this is based on the fact that the control algorithms are set for 120bpm). Currently our musical state representation can work with a maximum of 256 different musical parts. Each musical part has a "Sample Collection" of maximum 128 samples performed by human teachers. MSR for each distinct sample "j" for a given musical part "g" (MP_g) are stored in the "Sample Collection" at the "Feature Extraction" level as shown in Figure 17. Our reason to choose a collection mode instead of a learning mode, where each new sample updates a single consolidated data structure, is to keep all available variations alive for use in improvisation.

The number streams in MSR are defined as follows:

Definition 1: Number stream "***N***" stores note change information in the MSR in the form of whole numbers. ***N*** streams do not indicate absolute note values but represent the note as its tonal difference from the previous note. Therefore a starting note is necessary to generate the note change stream from an ***N*** stream. The note length is implicitly defined in an ***N*** stream by the length of "0" values (no new notes) following a note change (a non-zero value in the stream).

Starting note value is recorded for each musical part. ROMI's cognition system is mostly focused upon duration, loudness and pitch difference taken in this order of importance. In the ***N*** number stream the amount of note change (delta) from the current note to the next note is recorded as shown in Figure 19, below.

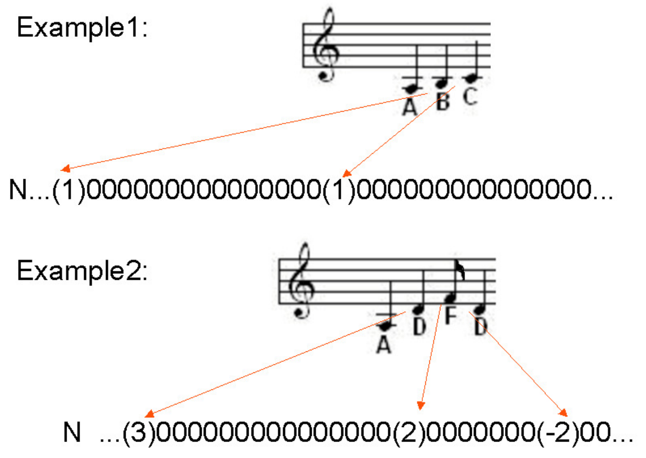

Example1:
A
B
C
N...(1)000000000000000(1)000000000000000...
Example2:
A
D
F
D
N ...(3)000000000000000(2)0000000(-2)00...

Fig 19. Calculation examples for number stream ***N***

Definition 2: Number stream "***M***" stores loudness change information in the MSR in the form of whole numbers. ***M*** streams do not indicate absolute loudness values but represent the loudness value as its difference from the previous note. Therefore a starting note loudness value is necessary to generate the loudness change stream from an ***M*** stream. The note length is again implicitly defined in an ***M*** stream by the length of "0" values (no new notes) following a note change (a non-zero value in the stream).

Starting note velocity (loudness) value is recorded for each musical part. In the ***M*** number stream the amount of loudness change (delta) from the current note to the next note is recorded as shown in Figure 20, below.

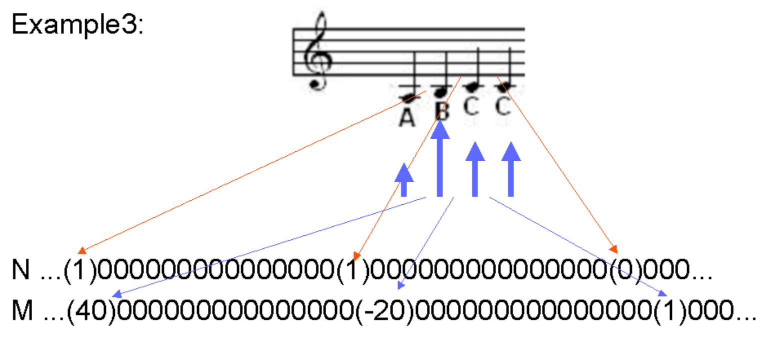

Example3:
A
B
C
C
N ...(1)000000000000000(1)000000000000000(0)000...
M ...(40)000000000000000(-20)000000000000000(1)000...

Fig 20. Calculation example for number stream ***M***

Definition 3: Number stream "***A***" stores event indication information in the MSR in the form of whole numbers. A non-zero value in a number stream ***A*** indicates that there is either a tonal change or a loudness change from the previous note. Number stream ***A*** cannot be used to generate notes on its own, it is only an indication of an event in the ***N*** or ***M*** streams. The need for an event indicator, such as number stream ***A***, is a direct result from our experiments with ROMI. Experimental results showed that the structure of our MSR needs such an event indicator to compensate for the time granularity of the system.

Number stream "***A***" is calculated from the available number streams ***N*** and ***M***, according to the following procedure. This procedure is shown in Figure 21.

```
Initialize A(t)=0
for t = 1 to 1920
{for i = 1 to 12,   A(t)=A(t)+1 if $N_i(t) \neq 0$ or $M_i(t) \neq 0$; else A(t)=A(t)}
```

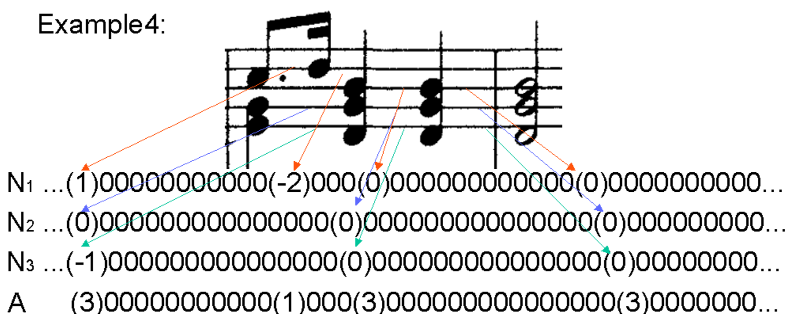

Example4:
N1 ...(1)00000000000(-2)000(0)000000000000(0)0000000000...
N2 ...(0)000000000000000(0)000000000000000(0)000000000...
N3 ...(-1)000000000000000(0)000000000000000(0)00000000...
A (3)00000000000(1)000(3)000000000000000(3)0000000...

Fig 21. Calculation example for number stream ***A***

Based on the defined ***A***, ***N*** or ***M*** streams we further introduce imitation parameters w, y, x, p, r, q that will also be used for improvisation. These parameters are defined as follows:

Definition 4: Imitation parameter "w" defines a percentage value to determine the existence of a note in a given time window "p", based on note change information stored in ***N*** number streams. When a sufficient number, "w" percent of samples, have the same note change value within a time window of "p" slots, the imitation process executes a note change (plays a note) on ROMI. Imitation parameter "w" is the main note generator using the note change information, in the ***N*** number streams.

Definition 5: Imitation parameter "p" defines the width of the time window which groups note values in an ***N*** stream together, in order to compensate slight tempo variations or less than perfect human teacher performances. A considerable amount of notes are sounded about 1/64th of a note before or after they are in the original score. If a time window to group such unintentional small time variations is not employed then the MSR will have far too many variations than intended. The control unit places the note at the time slot where majority of the ***N*** values are situated. When there is a draw, the first such slot will be chosen.

Definition 6: Imitation parameter "y" defines a percentage value to determine the existence of a note in a given time window "r", based on loudness change information stored in ***M*** number streams. When a sufficient number, "y" percent of samples, have the same note change value within a time window of "r" slots, the imitation process executes a note change (plays a note) on ROMI. Imitation parameter "y" is the secondary note generator using the loudness change information, in the ***M*** number streams. A new time window parameter "r" has been defined other than "p", in order to gain more control on imitation performance.

Definition 7: Imitation parameter "r" defines the width of the time window which groups note loudness values in an ***M*** stream together, in order to compensate slight tempo variations or less than perfect human teacher performances. A considerable amount of notes are sounded about 1/64th of a note before or after they are in the original score. If a time window to group such unintentional small time variations is not employed then the MSR will have far too many variations than intended. The control unit places the note at the time slot where majority of the ***M*** values are situated. When there is a draw, the first such slot will be chosen.

Definition 8: Imitation parameter "x" defines a percentage value to determine the existence of a note in a given time window "q", based on event indication information stored in ***A*** number streams. When a sufficient number, "x" percent of samples, have some event indication value within a time window of "q" slots, the imitation process executes a note change (plays a note) on ROMI. Imitation parameter "x" is used for generating notes that are in the original score but were not produced by the note generators driven by the "w" and "y" imitation parameters. "x" parameter is used to track the changes in ***N*** and ***M*** streams and generate a note where there has been sufficient changes in ***N*** and ***M*** streams to hint the existence of a note. A new time window parameter "q" has been defined other than "p" and "r" in order to gain more control on imitation performance.

Definition 9: Imitation parameter "q" defines the width of the time window which groups event indicators in an ***A*** stream together, in order to compensate slight tempo variations or less than perfect human teacher performances. A considerable amount of notes are sounded with a timing difference of about 1/64th of a note compared to the original score. If a time window is not employed so as to group such unintentional small time variations, then the MSR will have far too many variations than intended. The control unit places the note at the time slot where majority of the ***A*** values are situated. When there is a draw, the first such slot will be chosen.

The example in Figure 22, below is a representation of the data in the MSR for a monophonic sound with three separate samples. This is how data is stored in all of the collections.

```
N11 c2(0)000000000000000(0)000000000000000(0)000000000000000(0)000000000000000
M11(100)000000000000000(1)000000000000000(1)000000000000000(1)000000000000000
A1     (1)000000000000000(1)000000000000000(1)000000000000000(1)000000000000000

N12 c2(0)000000000000000(0)000000000000000(0)000000000000000(0)000000000000000
M12(100)000000000000000(1)000000000000000(1)000000000000000(1)000000000000000
A2     (1)000000000000000(1)000000000000000(1)000000000000000(1)000000000000000

N13 c2(0)000000000000000(0)000000000000000(0)000000000000000(0)000000000000000
M13(100)000000000000000(1)000000000000000(1)000000000000000(1)000000000000000
A3     (1)000000000000000(1)000000000000000(1)000000000000000(1)000000000000000
```

Fig 22. Example number streams for three samples of the same monophonic sound after the feature extraction process (for Nij and Mij and Aj i=1,2,3; j=1,2,3)

During recording, each Musical Piece (MP) sample from a human teacher is preprocessed and then the feature extraction step is applied in the MSR by converting the musical piece from MIDI to MSR. There can be at most, 256 different music parts stored in memory. The index "g", which is between 1 and 256, differentiates between these 256 possible different musical pieces. Therefore the notation MP_g, is used to denote different musical pieces in MSR. There can be at most 128 collection samples per MP_g stored in the memory for each musical part. The index "j", which is between 1 and 128, differentiates between these 256 possible different samples for a given MP_g. For each monophonic sound in a given MP_g the ***N*** and ***M*** streams are assigned such that, an $N_i(t)$ and $M_i(t)$ stream pair is recorded into the MSR. The index "t", which is between 1 and 1920, differentiates between the 1920 possible time slots available in the memory for each monophonic sound for each sample of a given MP_g.

The MSR can be polyphonic; the whole musical data is stored up to 12 distinct, ***N*** and ***M*** number stream pairs, meaning that 12 polyphonic music pieces can be processed. If there is a polyphonic sound then the "i" is appended to the ***N*** and ***M*** notations to distinguish between different monophonic voices in the polyphonic sound. However, this is not always the case and the "i" index is only used for polyphonic sound when needed. Otherwise this index is just kept at 1 and not used in the processes. For each monophonic sound in a given MP_g an index "i", which is between 1 and 12, is used to differentiates between 12 possible monophonic sounds, resulting in a system with 12 polyphony.

Each monophonic voice is represented by two number streams "***N***" and "***M***", where the number values in the streams are integer numbers between -127 and 128. "0" value for ***N*** and ***M*** and "-1","1", "-127" values for ***M*** streams have special meanings. Stream ***N*** records the relative pitch difference between consecutive notes. Stream ***M*** records the relative loudness difference between consecutive notes. The stream itself is a record of the duration of all notes. When there is a change in the current note, at least one of the two number streams register this event in the array structure by recording a non zero number. The number streams ***N*** and ***M*** consists of "0" values as long as there is no change in the current note. Each number in these streams are equivalent to a 1/64th note duration. In the real world a 1/64th note is almost incomprehensibly short.

Silence is considered as a note with starting loudness value of -127. When silence ends, the ***M*** stream resumes from the last note loudness value attained before the silence. If a note has the same note value as the previous note then the ***N*** stream will record a "0" but the ***M*** stream will record the loudness change value of "1" if loudness remains the same. If two notes are played sequentially with the same loudness value, then the ***M*** stream will record a "1" value at the note start, therefore "1" value is not used for loudness changes in ***M*** streams. "Time" or "sequential order" of the notes is represented by whole numbers starting from 1, and is denoted by the independent variable "t" in parenthesis.

4.4 Imitation

During playback, ROMI imitates a musical piece using imitation parameters by playing all notes where $N_{ij}(t)$ has identical value with at least "w" percent of all j iterations, within a time window of p slots, with average value of all available non zero $M_{ij}(t)$ values. Then a secondary run is made to playback notes that have not been defined by a note change, but is defined by a loudness change by playing all notes, where $M_{ij}(t)$ has a loudness value in at least "y" percent of all j iterations, within a time window of r slots, with average value of all available $N_{ij}(t)$ values. As a result of our experiments, the need for an event indicator was necessary to play existing notes not generated by the note or loudness changes. Therefore a third run is made playing all notes, where $A_j(t)$ is not "0" for "x" percent of all available j iterations, within a time window of q slots, with average value of all available $N_{ij}(t)$ values and with average value of all available $M_{ij}(t)$ values. If $A_j(t)$ lengths are different, the longest available length is selected as the music piece length, with gradually decreased loudness, starting the decrease with the shortest available length

The effects of these parameters on imitation performance are discussed in chapter 5. The following figures present a visualization of our proposed MSR. Here the opening part of Ludwig van Beethoven`s Ecossais has been used as the musical part being imitated that is used in the studies for this thesis work. The MIDI file for the original score of this musical part can be found in [56]. Since this is a MIDI file it will be played back by the default sound generation software with a fixed tempo and default electronic instrument which usually is a piano. Figure 23, shows how the original recording is represented based on our MSR notation. Note that, the data is in fact a one dimensional array of whole numbers. To aid in visualization, this one dimensional row has been folded in an array at every 64th element. Thus the vector wraps up in an array by continuing from one line below for each 64 consecutive array elements. The numbers in the first column represents this arrangement. The first note is a special character which stores the information of its value and velocity. After the first note, all information is stored as the difference between two consecutive notes. As long as there are no note changes streams ***N*** and ***M*** consists of "0" values and are shown by mid level gray tone in Figures 23 and 24. As shown by the legend to the right of the figures, lighter tones of gray indicate a positive change in ***N*** and ***M*** streams; and darker tones of gray indicate a negative change. Therefore, every move from the mid level gray tone indicates a note change. In the rare event of a higher than 5 note pitch difference between two consecutive notes, the graph shows such changes with pure white or black regardless of the absolute difference value. Note that the changes in ***M*** streams has a larger scale. This is due to the fact that velocity changes can occur in wider absolute value range. Pure black array elements represent a "silence" in ***M*** streams.

Figure 24, shows the MSR for one of the sample performances of the same musical piece that ROMIlearned to imitate by identifying one of our human teachers playing it on a piano. It is possible to “see” the difference with the original score where the recording imitated from human teacher has small deviations from the original score.

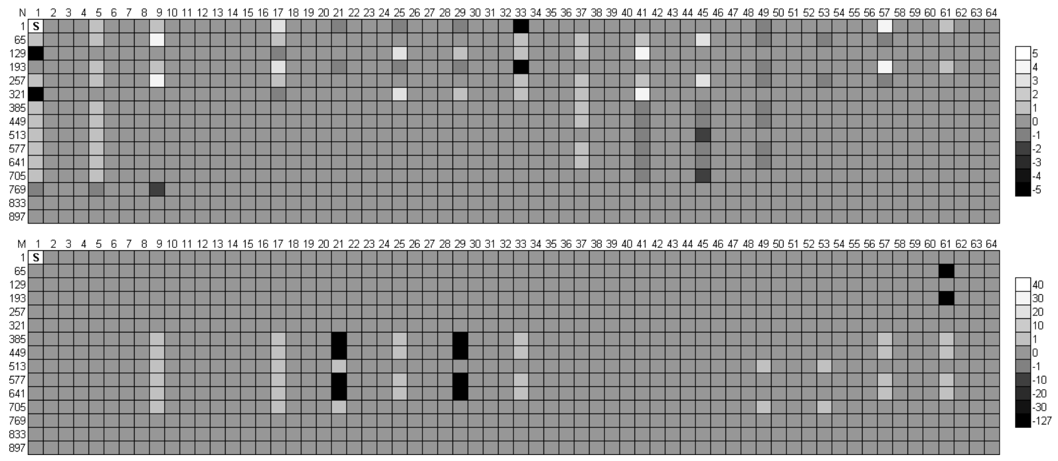


Fig 23.   ***N*** & ***M*** number streams for Original Score in MSR

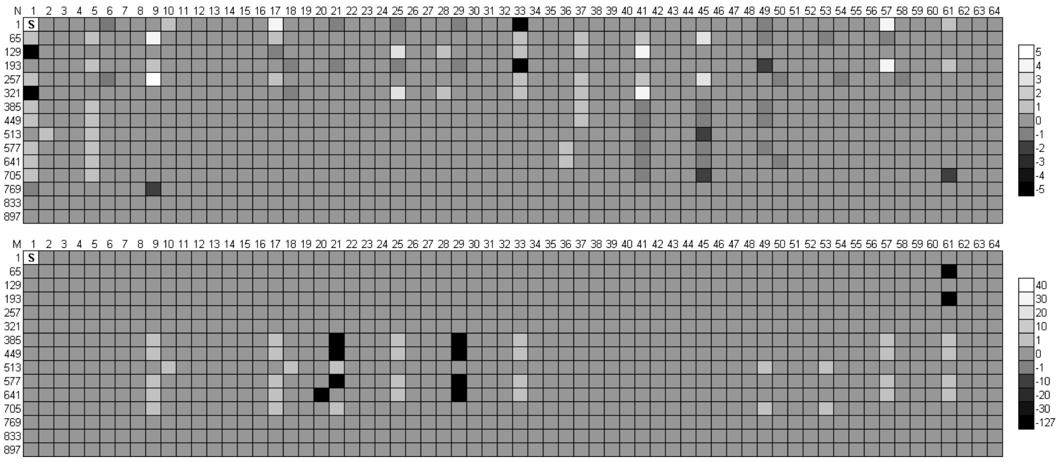


Fig 24. ***N*** & ***M*** number streams for Played Sample in MSR

Figure 25, shows the difference (Delta Vector) between the original score and the learned sample from the musical piece played on a piano by one human teacher. In the representation of the Delta Vector the value zero is shown with pure white color since the absolute value of the difference is of importance. In this figure all non-zero array elements represent a note being played by the human teacher either with a wrong value or at the wrong time with respect to the original score. The number of non-zero (non-white) elements and their intensity is a measure of how good the performance of the human teacher was learned for imitation.

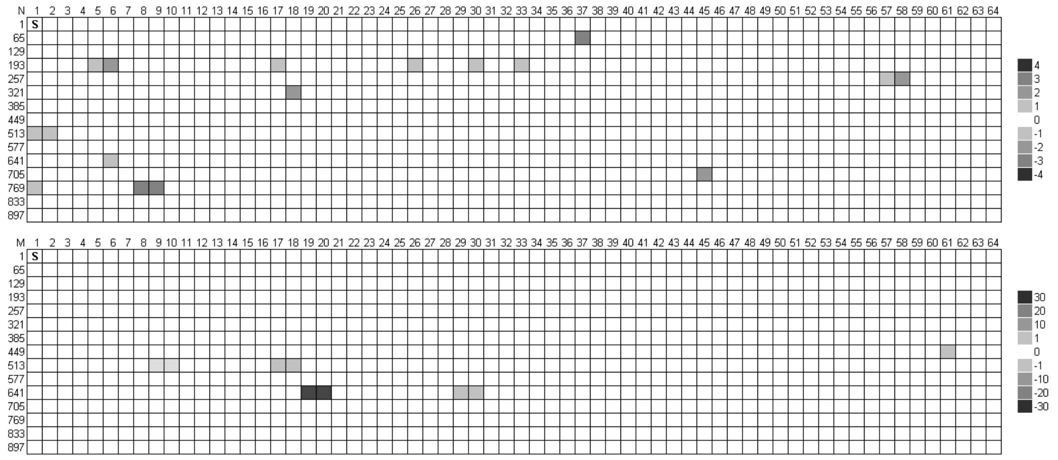


Fig 25. Delta in ***N*** & ***M*** number streams between Original Score and the Played Sample in MSR

4.5. Improvisation

4.5.1 Improvisation by Relaxation of Imitation Parameters

The values of imitation parameters that minimize Delta_k produce an output very similar to the original score, or the median of the samples. Therefore to achieve improvisation we have to relax the imitation parameters so as to use values that result in non-minimum Delta_k. This will provide controlled variability from the original sound under controlled relaxation. Our tests showed that the imitation parameter values that produce less notes, for any given range of time slots, than the number of notes in the same range of time slots of the original score, are less suitable for improvisation. If the w, y and x values are above 90, then the resultant improvisation will have much less number of notes compared to the original score for any given range of time slots and the resulting improvisation receive very low grades from the subjective listener surveys.

The following example illustrates this. Figure 26, shows the ***N*** & ***M*** streams for a test sample generated with relaxed imitation parameters resulting in less notes than the original score. In this example, this is achieved by setting y and w parameters to 100 and the other parameters are set to their near optimal imitation values of x=55, p=3, r=3, q=4. As seen from Figure 26, there are less notes than the original score given in Figure 23. The Delta_k is however higher than that of the sample given in Figure 24. This can be seen if the Delta Vector for Improvisation Sample 1, given in Figure 27, is compared with the lower value Delta_k sample given in Figure 25.

Please note that Improvisation Sample 1, is one of the random samples available. When the imitation parameters are relaxed, there may be many possible different output samples (improvisations) and only one such possible output sample is given in Figure 26. The system produces the same output sample (improvisation) for a given set of relaxed imitation parameters. For different sets of imitation parameters, different output samples are produced with different Delta_k values. The sample given in this example is one with an average Delta_k. On its own it can not be classified as an improvisation but as a bad imitation sample.

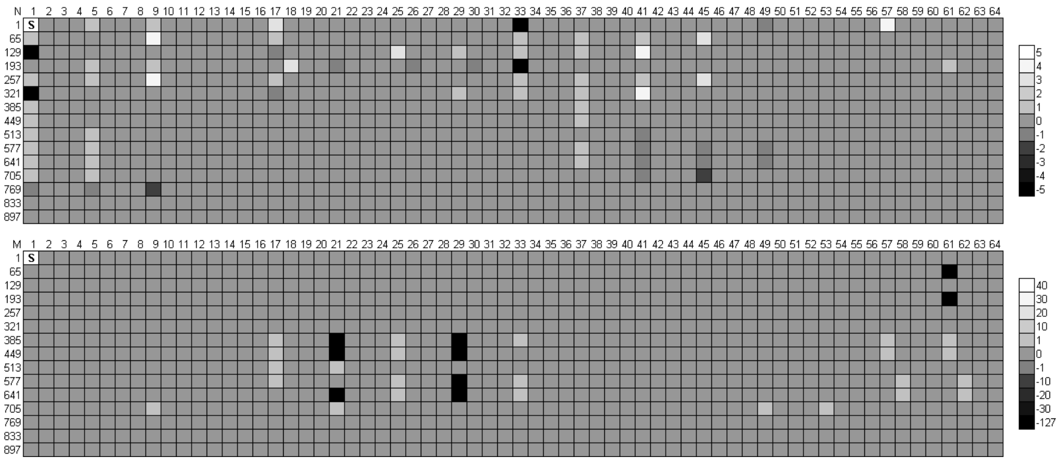


Fig 26. ***N*** & ***M*** number streams for Improvisation Sample 1

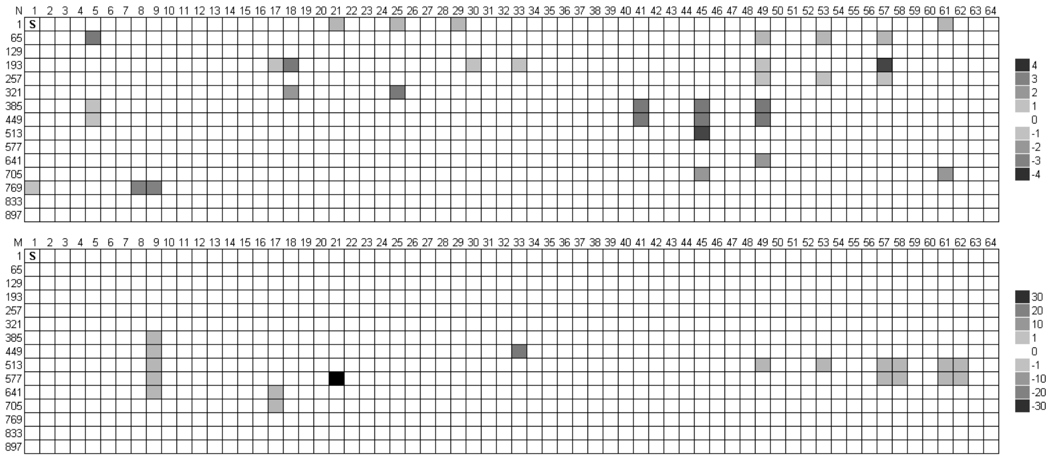


Fig 27. Delta in ***N*** & ***M*** number streams between Original Score and Improvisation Sample 1

Experimenting with other parameters where the resultant output has fewer notes than the original score give similar results. For example if y and w are kept at their near optimal imitation values of 85 and p and r are set to 2 with x=55, q=4, the result is an output with high Delta_k value due to the fact that the output sample has considerably less notes than the original score.

In order to achieve better improvisation by relaxation of the imitation parameters, using values that result in both non-minimum Delta_k and produce more notes than the original score are more suitable. The following example illustrates this. Figure 28, shows the ***N*** & ***M*** streams for a test sample generated with relaxed imitation parameters resulting in more notes than the original score.

In this example, this is achieved by setting y and w parameters to 75 and the other parameters are set to their near optimal imitation values of x=55, p=3, r=3, q=4. As seen from Figure 28, there are more notes than the original score given in Figure 23. The Delta_k is similar to the sample given in Figure 24. This can be seen if the Delta Vector for Improvisation Sample 2, given in Figure 29, is compared with the lower value Delta_k sample given in Figure 25. Again, Improvisation Sample 2, is one of the random samples available.

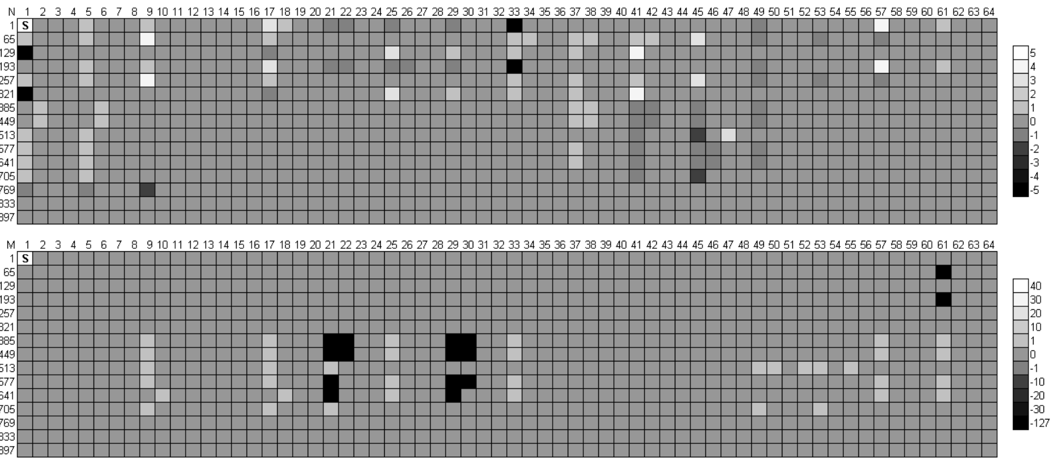


Fig 28. ***N*** & ***M*** number streams for Improvisation Sample 2

There is limited success in improvisation for this sample. Experimenting with other parameters where the resultant output has more notes than the original score give similar results. For example if y and w are kept at their near optimal imitation values of 85, p and r are set to 5 with x=55, q=4, the result is an output with high Delta_k value due to the fact that the output sample has more notes than the original score.

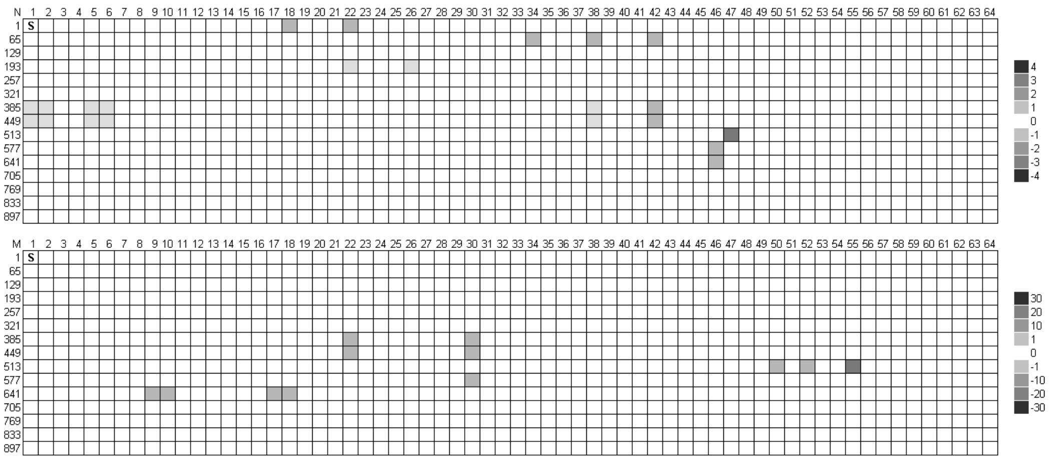


Fig 29. Delta in ***N*** & ***M*** number streams between Original Score and Improvisation Sample 2

If imitation parameters are further relaxed the system tends to drift too much out of the original scores note frequency range. This result is due to the fact that our MSR is based on note differences and when the imitation parameters are relaxed to a large extend, the notes produced by the system will be in another octave range than the original score, which will cause the improvisation to sound like a different musical piece. For example, if the imitation parameters are set at y=10, w=10, x=90, p=1, r=1, q=2; the resultant improvisation has so many more notes than the original score that it will drift out of the octave range of the original score and may contain notes from up to 4-5 octaves above or below the original scores' octave range.

We propose to use the improvisation by relaxation of imitation parameters partially in the musical piece, where the improvisation parts are embedded in the imitation parts. For example, a mask as shown in Figure 30, can be applied. In this mask the white slots represent where the original score will be played back by the imitation algorithm and the black slots represent where the imitation parameters are very relaxed. Parameters for the “black” slots are set at y=50, w=50, x=50, p=2, r=2, q=3; and imitation parameters for the “white” slots are set at y=85, w=85, x=55, p=3, r=3, q=4.

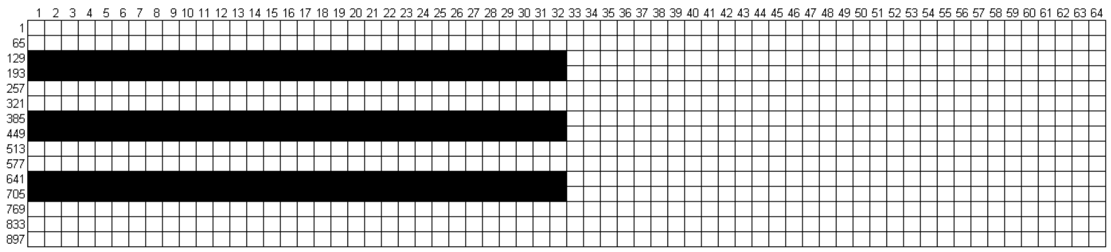


Fig 30. Mask for improvisation intervals, black slots represent where imitation parameters are very relaxed

There can be many other choices for defining such a mask. For example, there can be masks with not only two sets of imitation parameter values (one for imitation and one for improvisation) but with more sets of varying values. Such an approach will add even more variations into the musical part. But then the obvious question is what controls the selection of such masks? The answer maybe “a higher level of improvisation algorithm”. So it is possible to use Improvisation by Relaxation of Imitation Parameters (IRIP) as a low level tool, whose parameters are defined by time intervals determined by a higher level improvisation algorithm. We propose to use a periodic function similar to the one given in Figure 30. Best results are obtained if the length, in terms of time slots, of each period of the function is a multiple of 8.

There are some ground rules that we have discovered during test runs. These rules can be outlined as:

1. Short periods of IRIP does sound as if a wrong note has been played. Therefore we suggest that the minimum duration for an IRIP part must be at least 1 seconds (or 2 quarter notes for 120bpm).
2. Long periods of IRIP tend to drift out of the scale of the musical piece being played. This is due to our MSR. The most common result of the IRIP is the addition of the same note at a very close time interval of the original note. Since MSR is a difference representation, this addition of new notes in improvisation drifts the note sequences out of the scale of the musical piece. This yields the rest of the musical piece being played at a different pitch. To limit this effect, our proposition is that these intervals should not be larger than 2 seconds (or 4 quarter notes for 120bpm). And at certain intervals the musical piece should be reset to one absolute note value. This of course requires a higher level algorithm that starts an IRIP part, to calculate the absolute note at the start of the IRIP and return (reset) the musical piece to that absolute note value at the end of the IRIP.
3. The starting time of an IRIP should be snapped to a grid of 1/8th note durations. This helps to ensure that the IRIP has the same tempo as the musical piece.
4. The duration of an IRIP should be multiples of 1/8th note durations. This helps to ensure that the IRIP has the same tempo as the musical piece.
5. If more than one IRIP is going to played in a musical piece. We advice to put imitation parts between them that are at least the same length of the last IRIP being played. This gives the listener the necessary clues of what the modal of the musical piece is.

The following example in Figure 31, helps to visualize these ground rules. In this mask the white slots represent where the original score will be played back by the imitation algorithm and the black slots represent where the imitation parameters are very relaxed. The gray slots are where we recommend the start of an IRIP should be snapped to. Figure 31, gives the IRIP rules in the same order as stated above; the first rule that requires at least 1second long IRIP parts, is shown by indicating that, an IRIP part which is only 0.2 seconds for a tempo of 120bpm, is too short. The second mask shows that an IRIP part which is 2 seconds for a tempo of 120bpm is too long. The gray points show the advisable points to start and stop an IRIP part. The last mask shows that IRIP parts which have durations that are not multiples of 1/8th note durations, are not good choices since the improvisation part will sound out of tempo.

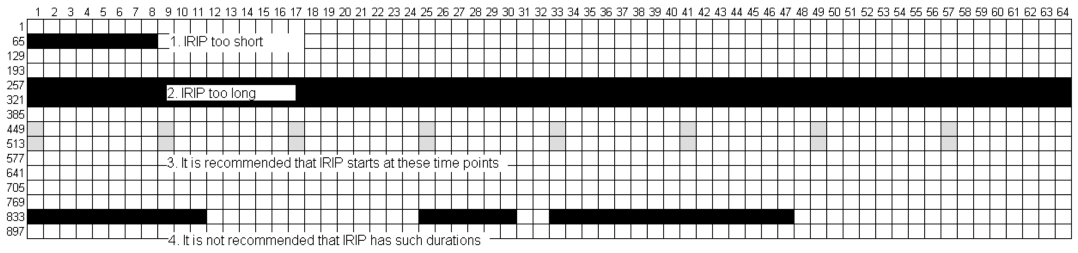

1. IRIP too short
2. IRIP too long
3. It is recommended that IRIP starts at these time points
4. It is not recommended that IRIP has such durations

Fig 31. Mask for improvisation intervals, black slots represent where imitation parameters are very relaxed

Another approach to a more vivid improvisation can be the utilization of polyphony. Since the control algorithm of ROMI is 12 polyphonic, this can be implemented in many creative ways. The most basic one would be to reproduce the imitation from the frequency range of the original scores while playing the improvisation on top of it from a higher or lower octave. Test results have showed us that for polyphonic music, the number of distinct monophonic parts where IRIP will be employed should not exceed more then one fourth of the total monophonic parts creating the polyphonic musical piece. Therefore, for 12 polyphonic music the highest advisable number of monophonic parts to employ IRIP, should be at most 3.

The results discussed in this section are based on our subjective evaluation of the improvisation results. The proper way of evaluating these results is by employing a group of listeners and asking them to rate the results. Appendix B, contains the Improvisation Survey Form that we have developed for this purpose. The results of the survey are presented in section 5.3.

4.5.2 Patching as an Improvisation Locator Tool

When an IRIP based on the first 5 rules defined in the previous section is generated, there is an open possibility of inserting this improvisation part at different points of the piece being imitated.

Experiments showed us that generating better quality IRIP parts can be achieved, if a mechanism like "patching" is employed in selecting these different points by using a cost function Since the IRIP can be placed at different points that minimize this cost function, a selection criterion can be defined.

If the imitation part has some properties that are helpful in patching. The first property is that, the imitation signal has absolute bounds defined by the octave range of the original score. This is due to the fact that all musical devices work within bounds, defined in octaves in western music. Since musical instruments cannot sound outside their octave range, the notes intended to be played on them are composed within the instruments octave range, or absolute bounds. This brings us to the second property which we call "variation". Since the imitation and improvisation signals move within bounds, when patching is to be applied, it is more frequent that the patching algorithm will fit a variation rather than the slope of the signal. Notes usually do not move in one direction but they usually "vary" between local bounds that are within the absolute bounds of the signal. Using these two properties, the experimental results showed us that using a "1/8th note wide" header and footer for the application of the patching process is adequate. Note that this length is not an absolute value and depends on the "tempo" of the musical part. If this length is selected smaller than 1/8th, then the possible patching points in the musical part increase but the patching accuracy decreases. When the header and the footer of the musical part to be compared is shorter, then there are more points in the whole musical part where a high match can be achieved implying higher accuracy. For example, if the header and footer are only "1/4th note wide" then are twice as many possible positions in the whole musical piece, compared to "1/8th note wide" header and footer, where the patch can be inserted. When this length is higher than 1/8th, then it becomes harder to find a patching point, although the patching accuracy would be higher. For example, if the header and footer are "1/2th note wide" then are only half as many possible positions in the whole musical piece, compared to "1/8th note wide" header and footers, where the patch can be inserted.

Patching accuracy is defined as a positive whole number, which is computed as the distance between the notes in the imitation part that are replaced by the header and footer of the improvisation part. The patching process tries to minimize this distance to increase the accuracy of the patching which in turn will increase the quality of the improvisation. This is a critical point since, even though the generated improvisation part would stay the same, a change in the patching point would effect the musical quality of the improvisation.

The patching process first converts the MSR of notes into a graph by keeping the graph value constant for the notes' length. For example, if the current note change value at the given time slot is "1" then the graph will keep this value constant for the duration of that note. The graph changes only with a note change command and will change the graph value as the amount of the note change. This is not done for the whole musical part but is done for only candidate patching entry and exit points defined by the 5 rules given in the previous section. For example, a note sequence such as "1000000020000000-1000000020000000" is represented by the following graph given in Figure 32, below.

All such comparison graphs are assumed to start from the zero value since only the absolute note change values are of importance. The graph immediately changes value at the second time slot since the note change "10000000" is received. The graph keeps this "1" value on the vertical axis for a total number of 8 time slots as given by the note length "10000000". When the second note change command "20000000" is received the graph moves from "1" to "3" in the vertical axis to reflect that this is a note change 2 units (notes). The graph keeps this "3" value on the vertical axis for a total number of 8 time slots as given by the note length "20000000". When the third note change command "-10000000" is received the graph moves from "3" to "2" in the vertical axis to reflect that this is a note change of -1 units (notes). The graph keeps this "2" value on the vertical axis for another 8 time slots as given by the note length "-10000000". When the fourth note change command "20000000" is received the graph moves from "2" to "4" in the vertical axis to reflect that this is a note change of 2 units (notes). The graph keeps this "4" value on the vertical axis for another 8 time slots as given by the note length "20000000".

The vertical axis is the mean note variation in stream ***N***.

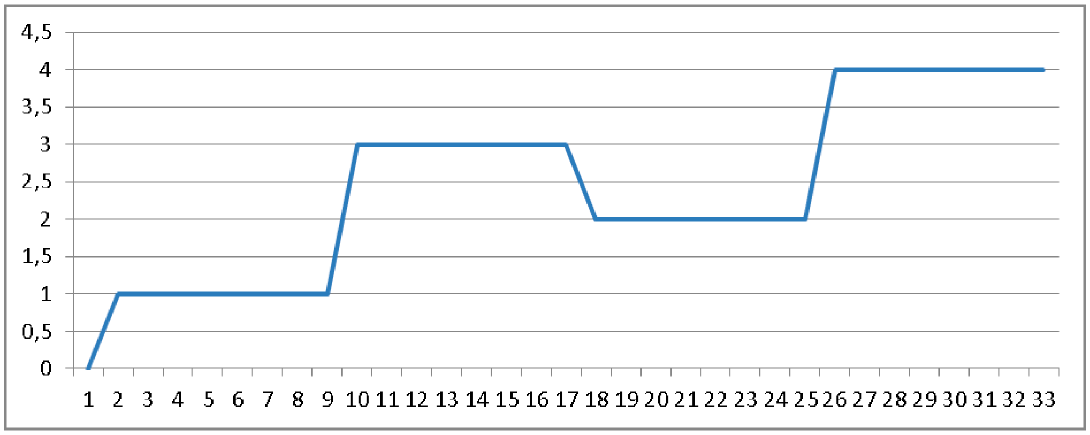


Fig 32. Example graphical illustration of a note sequence used in patching

The horizontal axis is like "time" for an electrical signal but is not in absolute values since it depends on the "tempo" of the music. The signal is always assumed to start from "0" value since the notes in our MSR are not absolute values but are differences between note values. The patching process also does not need the absolute values of the notes since it is based on a distance calculation.

The following example in Figure 33, illustrates how the patching process calculates the distance of the header of an improvisation part to be inserted onto an imitation part. Here the solid line represents the imitation signal and the dotted line represents the improvisation signal. The shaded area is the distance between the notes. The patching process being used tries to minimize this area, or distance at both the header and footer of the improvisation signal without any priority being given to either to header or footer parts. The minimization process will calculate the area shown in gray in Figure 33, for all possible header and footer locations in the whole musical piece. Then the point which yields the minimum value for all calculated difference values is selected as the insertion point of the improvisation. The length of the header and footer effect the computation cost, with longer headers and footers requiring more computation. Experiments with the system showed that the footer part of an improvisation is as important as the header. This is because of the fact that the improvisation quality is degraded by sudden changes in the underlying motif of a musical piece, which can happen at the beginning or the ending of an improvisation part.

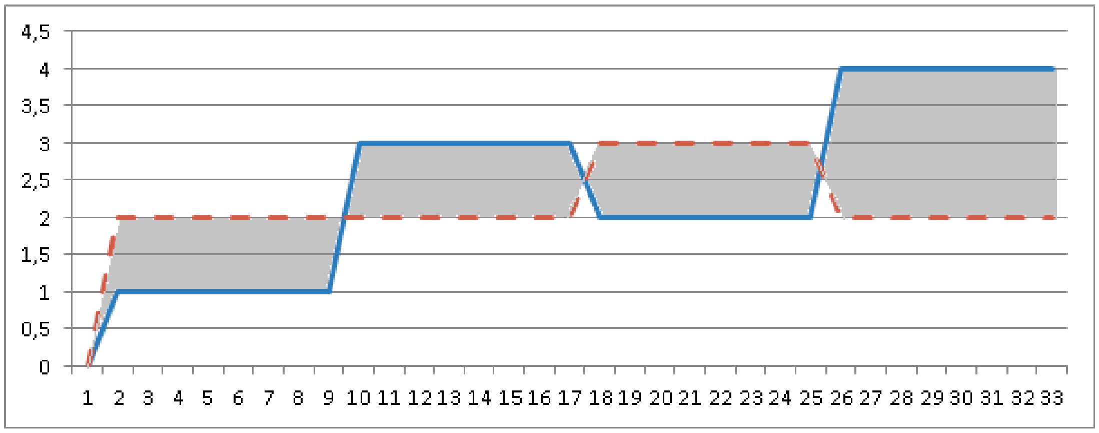


Fig 33. Example graphical illustration of distance calculation in patching

A monophonic example on the opening part of Ludwig van Beethoven`s Ecossais is given below. Only ***N*** stream values are used by the patching process, therefore the example is illustrated on ***N*** stream values. The first array of note changes (***N*** stream) in Figure 34, is the imitation result for a given set of imitation parameters. The second array that consists of only 128 time slots is the generated IRIP. The third array is the insertion of the IRIP into the imitation with a “higher” distance than the insertion of the IRIP into the imitation given by the fourth array. The distance values for both insertions are given. Inserting the 128 time slot long IRIP given in the second array into the imitation given in the first array at time slot 257 yields the third array with a patching distance cost function of 84. Inserting the 128 time slot long IRIP given in the second array into the imitation given in the first array at time slot 257 yields the third array with a patching distance cost function of 84. Inserting the 128 time slot long IRIP given in the second array into the imitation given in the first array at time slot 193 yields the third array with a patching distance cost function of 52.

4.5.3 Improvisation by n-grams

Some existing work has utilized Markov Chains to model improvisation as a probability function [17], [18]. These work use monophonic sound and do not make any use of loudness information. These work disregard the note length which is surprising since the resultant melody will be much different if note duration values are considered equal. The n-gram approach which is a form of Markov Chain has been applied to improvisation with success. Our contribution to the existing n-gram approaches is the addition of polyphonic sound and loudness values.

N-grams are being used for determination of a given musical piece from a library of musical pieces, given the first few notes. The main idea is to calculate the conditional probability $P(Y \mid X) = P(Y \cap X) / P(X)$ of a state Y (a note), given the previous n number of states (notes). This approach is good if there is already a library of the musical pieces, since the n-grams can be readily calculated from the library. Therefore, this approach yields good results for determining what a musical piece is by looking at only a few starting notes (states). Our control architecture also has this kind of information. Our system incorporates an original scores collection and also all training data are kept in the form of our MSR as well. By incorporating the velocity data in our MSR we generated a new n-gram based higher level improvisation algorithm.

An n-gram represents a melody as sequences of n notes which can be generalized as sequences of n states. An example is given below in Table 3, for a musical piece with starting notes of B B D C F D.

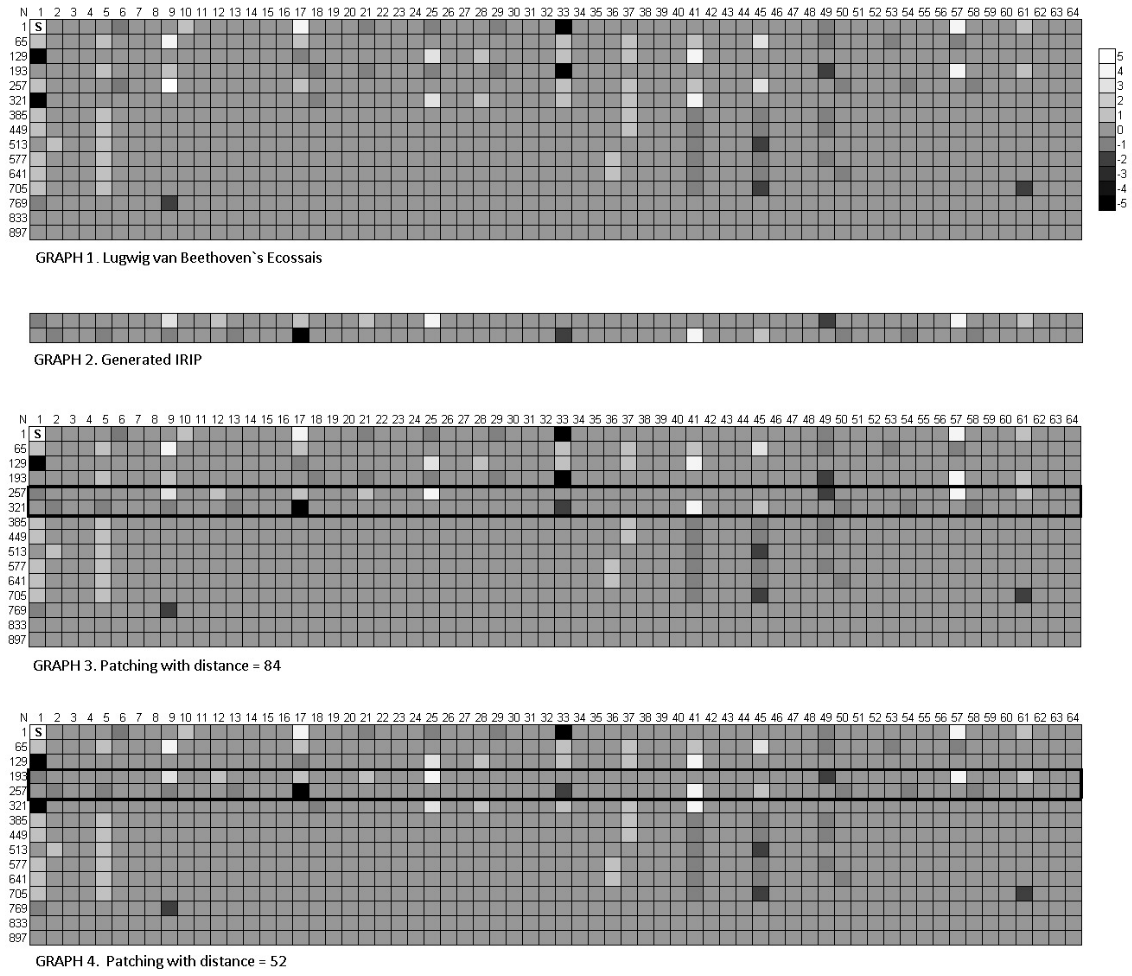


Fig 34. Graphical illustration of patching examples

Table 3. n-gram representations

1-gram	2-gram	3-gram
B	B,B	B,B,D
B	B,D	B,D,C
D	D,C	D,C,F
C	C,F	C,F,D
F	F,D	
D		

3-grams implicitly regard the current state as being represented by the last two notes played; 1-grams and 2-grams consider the current state as just being the current note, but 1-grams consider each note independently from previous notes. Depending on the size and nature of the data set, the value of the information captured by different order n-grams varies. For example, a 10-gram representation might capture less information than a larger n-gram.

In the example given above the musical information is adequate if the sole purpose is to determine what musical piece is this. Bu it is not adequate for improvisation since note durations are not specified and there is no way to represent a silence as well. Velocity information is also not employed, but this is very common with all existing works on improvisation [5]-[18].

In order to use n-grams for improvisation we must incorporate duration (of notes) into this approach and we need to represent silence. If we can incorporate velocity information as well, then this will be one of the very few studies doing so. Our MSR gives us the necessary information for all these. Its novelty resides in its nature to deal with differential values rather than absolute values of notes. Table 4, shows how the ***N*** and ***M*** streams are for this example note sequence. The following example in Table 5, shows the same musical piece (B B D C F D) in MSR and the resultant 1,2,3-grams similar to Table 3.

Table 4. ***N*** and ***M*** steam representations

N	B	0	0	0	0	0	0	0	-2	0	0	0	1	0	0	0
M	100	0	0	0	1	0	0	0	-10	0	0	0	1	0	0	0
N	-3	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0
M	-10	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0

Table 5. n-gram representations in MSR format

1-gram	2-gram	3-gram
B 100	B,0000 100,1000	B,0000,-2000 100,1000,(-10)000
0000 1000	0000,-2000 1000,(-10)000	0000,-2000,1000 1000,(-10)000,1000
-2000 (-10)000	-2000,1000 (-10)000,1000	-2000,1000,-30000000 (-10)000,1000,(-10)0000000
1000 1000	1000,-30000000 1000,(-10)0000000	1000,-30000000,20000000 1000,(-10)0000000,(20)0000000
-30000000 (-10)0000000	-30000000,20000000 (-10)0000000,(20)0000000	
20000000 (20)0000000		

Table 5, above is how n-grams are shown in our MSR. Here each entry has two rows, the first row is for the ***N*** stream values and the second row is for the ***M*** stream values. Note that now the notes also have durations. If there were a silence it would have been represented by a –127 value in the ***M*** stream. Note that our MSR records the first note in absolute value. With this information, the absolute note values for all notes can be calculated. In this example, the first four notes have a duration of 1/16th note and the last two notes have a duration of 1/8th note.

This n-gram approach would require more data to reach the same accuracy of a standard n-gram approach, since it differentiates between note values with different durations. This is natural since it also bares more information: duration of the notes.

The ***M*** stream values can have values between 2 to 100 and –1 to –100. But representing them in this accuracy for n-gram approach will necessitate too many samples. Therefore it is a better approach to limit the changes in ***M*** streams to multiples of 10. This approach looses some of the loudness change information.

Table 6. MATLAB representation of MSR

		1-gram	2-gram
MSR	N M	0000 1000	0000,-2000 1000,(-10)000
MATLAB	N M	0:3 1:3	0:3,-2:3 1:3,(-10):3

The ***M*** streams have been calculated as multiples of 10 to decrease the total number of possible nodes in the Markov Chains to a manageable level. However, the system required too many samples to improvise in an acceptable quality. To ease the computing we tried MATLAB for the n-gram based improvisation algorithms. This proved to produce better results. The MATLAB representation of our MSR had to be changed slightly. The number 0's in the ***N*** and ***M*** streams are utilized to determine the duration of a note. We tried to change the data in our MSR in a run-length-coding type of approach. In this approach instead of writing all the 0's into the representation we only write the number of 0's. An example is given in Table 6: Table 6, presents the transformation in representation where our MSR notation is converted into a format that we can utilize in using MATLAB.

This is especially useful for notes with long durations. The MATLAB representation uses two numbers to represent a possible state instead of a variable number of numbers depending on the notes length. After these modifications the results of the n-gram based improvisation improved.

The handling of polyphony works implicitly due to structure of our MSR. Each distinct sound existing in the polyphony is represented as different monophonic ***N*** and ***M*** stream. All such monophonic streams are handled on their own and added to the polyphony at the end of the improvisation step. This approach works fine for up to 4 polyphonic sounds. But the question on how this approach is handling cross note relations between the monophonic streams is open. To address this question a music theory that would relate cross note relations in a polyphonic sound is required. To our knowledge no such theory exists. If a rule set or musical approach is to be implemented to achieve some form of cross note relation handling, it can be readily inserted into our improvisation algorithm.

4.6 Sample Owners' Identification

Sample owners' identification can be achieved with limited success using the MSR. This identification does not involve any style analysis of the human teacher recording the sample. Sample owners' identification functionality, relies only on simple systematic error analysis that a human teacher makes while recording samples. These errors are analyzed in two groups; timing errors and note errors.

When the number of distinct human teachers using the system, are more than a certain finite number, in the order of 100, then the owner identification requires more information than can be driven out of the MSR with its current functionality. This is due to the fact that when the number of distinct human teachers increase, the probability that two different human teachers make the same type of systematic errors increase. So, 20 is selected as the maximum number of distinct human teachers as our current limit on the number of distinctly identifiable human teachers. The system is still useful in terms of pointing out systematic errors that a given human teacher makes, regardless of the number of distinct human teachers.

If the samples recorded by a given human teacher, differs from the MIDI representation of the musical piece by a constant amount of timing difference, then this systematic error can be communicated as a performance feedback to the human teacher or this information can be used to identify the sample owner if the performer is not known a priori.

Table 7, below gives an example list of systematic errors recorded by three different human teachers. This list has been derived from the MSR by a subroutine that seeks to pinpoint systematic errors that had been recorded during our experiments. This subroutine called as "Sample Evaluator" works on the Delta Collection within the Comparator stage of the control architecture. "Sample Evaluator" marks a certain type of error as "systematic", if it repeats for at least 3 instances for any given musical piece MP_g in the whole sample set. When this threshold value is set below 3, the subroutine marks too many errors and most of them are not systematic. This is not the goal of this application. The application is not trying to catch all errors, which would be the case if the threshold would be 1. When the threshold is increased above 3, then the subroutine marks very few errors, missing some of the systematic errors. When the threshold is increased above 4 the errors that are marked by the subroutine converges to zero.

"Sample Evaluator" marks a certain type of error as "systematic", if it repeats for at least 15 instances for the whole sample set for all recorded musical pieces. . When this threshold value is set below 15, the subroutine marks many errors, most "not systematic". If this threshold is below 3 then it marks all errors. When the threshold is increased above 15, then the subroutine marks fewer errors, missing some of the systematic errors. When the threshold is increased above 20 the errors that are marked by the subroutine converges to zero. Timing errors and note errors are counted separately and are not additive.

The length of the "systematic" error is limited to 3 distinct events, marked by the event indicator number stream $\boldsymbol{A}$. This limitation is necessary to compare systematic errors in a logical sense. If there had been no such limitation, then there would be systematic errors with little differences, such that the "Sample Evaluator" does not classify as a systematic error because their instances do not add up to 3 or 15.

In the example list above, if we assume that the human teacher is not known for samples 13, 30 and 62. "Sample Evaluator" can predict that the sample owner for sample 13, is human teacher ID 2; sample owner for sample 30, is human teacher ID 1; sample owner for sample 62, is human teacher ID 3. "Sample Evaluator" also shows these users, the systematic errors they are making while playing the given MP_g.

In the example list of systematic errors given in Table 3, it can be seen that human teacher ID 2, is making three types of systematic errors while playing MP ID 3. Two note errors and one timing error. The interesting point is that he is making both systematic note errors on one particular note sequence; 10003000-5000. For inexperienced human teachers like human teacher ID 2, it is easier for the system to determine the sample owners' identity since he is making more errors than an experienced human teacher like human teacher ID 1; who did only one systematic timing error in only one of his whole recordings, making him hard to tell apart from the other human teachers. For evaluating experienced human teachers, the number of distinct events being evaluated must be increased higher than three and the number of samples per MP_g must also be increased significantly, to try catching specific and more elaborate systematic errors that an experienced human teacher make.

Table 7. Example systematic errors indicated by "Sample Evaluator"

Sample #	**Teacher ID**	**MP ID**	**Recorded Note Sequence**	**MIDI Note Sequence**	**Type of error**
11	2	3	10003000-4000	10003000-5000	Note difference
12	2	3	10003000-4000	10003000-5000	Note difference
15	2	3	10003000-4000	10003000-5000	Note difference
19	2	3	10002000-4000	10003000-5000	Note difference
22	2	3	10002000-4000	10003000-5000	Note difference
26	1	4	20003000-4000	20003000-3000	Note difference
27	1	4	20003000-4000	20003000-3000	Note difference
28	1	4	20003000-4000	20003000-3000	Note difference
11	2	3	20002000-3000	2000200-30000	Timing Difference
12	2	3	20002000-3000	2000200-30000	Timing Difference
14	2	3	20002000-3000	2000200-30000	Timing Difference
26	1	4	20000200-3000	2000200-30000	Timing Difference
28	1	4	20000200-3000	2000200-30000	Timing Difference
29	1	4	20000200-3000	2000200-30000	Timing Difference
58	3	5	3000-20000-100	3000-2000-1000	Timing Difference
60	3	5	3000-20000-100	3000-2000-1000	Timing Difference
61	3	5	3000-20000-100	3000-2000-1000	Timing Difference
13	?	3	10003000-4000	10003000-5000	Note Difference
30	?	4	20003000-4000	20003000-3000	Note Difference
13	?	3	20002000-3000	2000200-30000	Timing Difference
62	?	5	3000-20000-100	3000-2000-1000	Timing Difference

The human ear quickly picks up the error and corrects the recording of the rest of the musical piece after the error. Human teachers achieve this by playing either the right note value after a wrong one or by playing the next note shorter or longer after a wrong one. Therefore, the error they make during the recording does not propagate to the rest of the recording, at the expense of recording at least one more wrong note, with respect to the MIDI representation.

CHAPTER 5

EXPERIMENTAL RESULTS AND SENSITIVITY ANALYSIS

In this chapter, some experimental results are given to clarify how the imitation parameters work with our MSR to produce music. Then the sensitivity analysis of the imitation parameters is conducted. The results of the survey that we developed to evaluate our different improvisation approaches will be presented with a final discussion.

The opening part of Ludwig van Beethoven`s Ecossais has been used as the musical part in the experimental studies presented in this chapter. Figure 23, shows how the original recording is represented based on our MSR notation.

5.1 Experimental Results

Experimental results will be presented as ***N*** and ***M*** number stream graphs and their associated delta vector graphs. This group of examples will demonstrate how the automatic parameter estimation process converges to local minimum delta, in an imitation.

Our proposed parameter estimation process incorporates an "Original Scores Collection" where each distinct musical part is in the form of our MSR. Therefore, each musical part has ***N***, ***M*** and ***A*** number streams in this collection. This original recording is considered as the "nominal" MSR for a given musical part and the distance "Delta" for each recorded sample by human teachers can be evaluated. If identity of each human teacher is known a priori for each sample, so it is possible to track the performances of each human musician, if not this process becomes that of sample owners' identification.

Original score information for each musical part enables our proposed system to measure the "quality" of each imitated sample j, for a musical part that exists in the Original Scores Collection. The "nominal" sample which is the imitation performance of ROMI, is tested for "quality" in imitation mode but not during improvisation mode. The difference between the MSR of the nominal sample and the MSR of any given sample j yields difference "Delta Vector" for each recorded sample j. All delta vectors for known musical parts are stored in a separate "Delta Collection".

The following algorithm outlines the main steps in the Delta calculation. In the following algorithm, number streams from the original score MSR are denoted in bold and italic, number streams from human teacher MSR are denoted in normal fonts to avoid confusion.

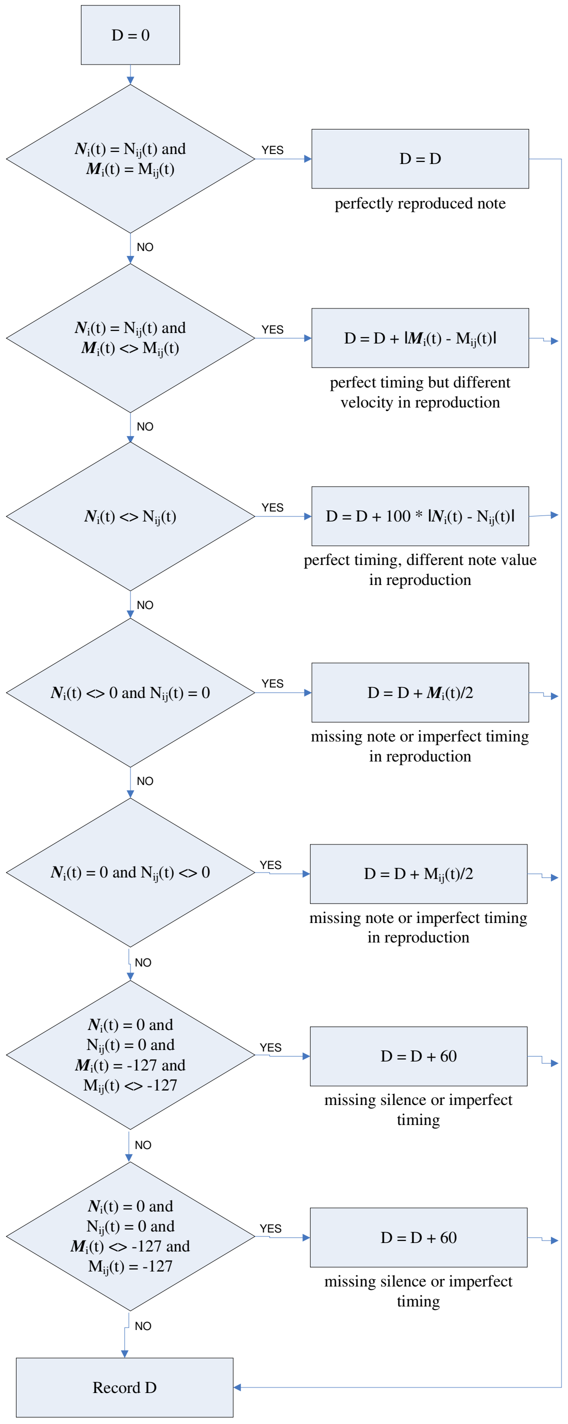
The imitation process uses the six user defined parameters. Three of these parameters, w, y, x, define an averaging factor to be used in note reproduction by the imitation process of ROMI. The other three, p, r, q, define a time window in which this averaging function will be used. Changing these parameters effect the output quality.

The idea used to calculate Delta, can be used in a similar approach to estimate these user defined parameters controlling the imitation process. For each recorded sample set, collected from the same musician for a given part, modifying the w, y, x, p, r, q parameters to find a minimum for the associated Delta is possible. This is the output of the 3rd line in the algorithm below. Delta is not calculated for each separate sample but it is calculated for all the available samples by the same human teacher playing the same musical part.

This algorithm fills the Delta Collection with one Delta value for each sample (Delta_j) and acquires an average Delta for each distinct human teacher (Delta_k) and an overall Delta (Delta_a) for all available samples reproduced by ROMI's imitation process for a given musical part. Also a "Delta Vector" is stored in the Delta Collection which is similar to the ***N*** and ***M*** streams for each sample.

Delta Calculation Algorithm

1. For sample j by human teacher k;


D = 0
$\boldsymbol{N}_i(t) = N_{ij}(t)$ and $\boldsymbol{M}_i(t) = M_{ij}(t)$
YES
D = D
perfectly reproduced note
NO
$\boldsymbol{N}_i(t) = N_{ij}(t)$ and $\boldsymbol{M}_i(t) <> M_{ij}(t)$
YES
$D = D + |\boldsymbol{M}_i(t) - M_{ij}(t)|$
perfect timing but different velocity in reproduction
NO
$\boldsymbol{N}_i(t) <> N_{ij}(t)$
YES
$D = D + 100 * |\boldsymbol{N}_i(t) - N_{ij}(t)|$
perfect timing, different note value in reproduction
NO
$\boldsymbol{N}_i(t) <> 0$ and $N_{ij}(t) = 0$
YES
$D = D + \boldsymbol{M}_i(t)/2$
missing note or imperfect timing in reproduction
NO
$\boldsymbol{N}_i(t) = 0$ and $N_{ij}(t) <> 0$
YES
$D = D + M_{ij}(t)/2$
missing note or imperfect timing in reproduction
NO
$\boldsymbol{N}_i(t) = 0$ and $N_{ij}(t) = 0$ and $\boldsymbol{M}_i(t) = -127$ and $M_{ij}(t) <> -127$
YES
D = D + 60
missing silence or imperfect timing
NO
$\boldsymbol{N}_i(t) = 0$ and $N_{ij}(t) = 0$ and $\boldsymbol{M}_i(t) <> -127$ and $M_{ij}(t) = -127$
YES
D = D + 60
missing silence or imperfect timing
NO
Record D

2. Record $D_j = D / f$ for sample j to Delta Collection, where "f" is a constant to adjust the Delta range for easier visualization of the results.

3. For each distinct “k”, calculate Delta$_k$ as the overall difference value for each different human teacher, there may be different number of samples recorded by different musicians so we need to normalize by the number of samples.
4. Calculate overall Delta$_a$ covering all available samples, for j=1 to n and normalize it by “n”, where “n” is the number of all available samples for musical part MP$_g$.

The following three examples show how the automatic parameter estimation for ROMI's imitation process starts with a random set of the six imitation parameters and converge to a local minimum Delta$_k$ value of an optimum reproduction out of imitation.

Example 1: Starts with a random set of six imitation parameters:

w= 32, y= 69, x= 19, p= 7, r= 6, q= 2

With these imitation parameters one of the resultant outputs and its corresponding delta vector is given in Figure 35. The resultant Delta$_k$ value is 366.

When the graph in Figure 35, is analyzed it can be seen that there are many different notes compared to the original which is defined by the MIDI file. This causes the delta vector to be very crowded and the Delta$_k$ value to be very high. This is caused by the fact that the averaging parameters are relatively low causing the system to produce almost all activities into a note. The time window parameters are also too high combining many consecutive notes into a single note. This situation is the worst possible, where imitation generate unintentional notes, also loosing some other notes in the process. The resultant reproduction of the original score is by no means accurate for this case. The musical piece seems to have tempo variations and the underlying melody is not perceivable.

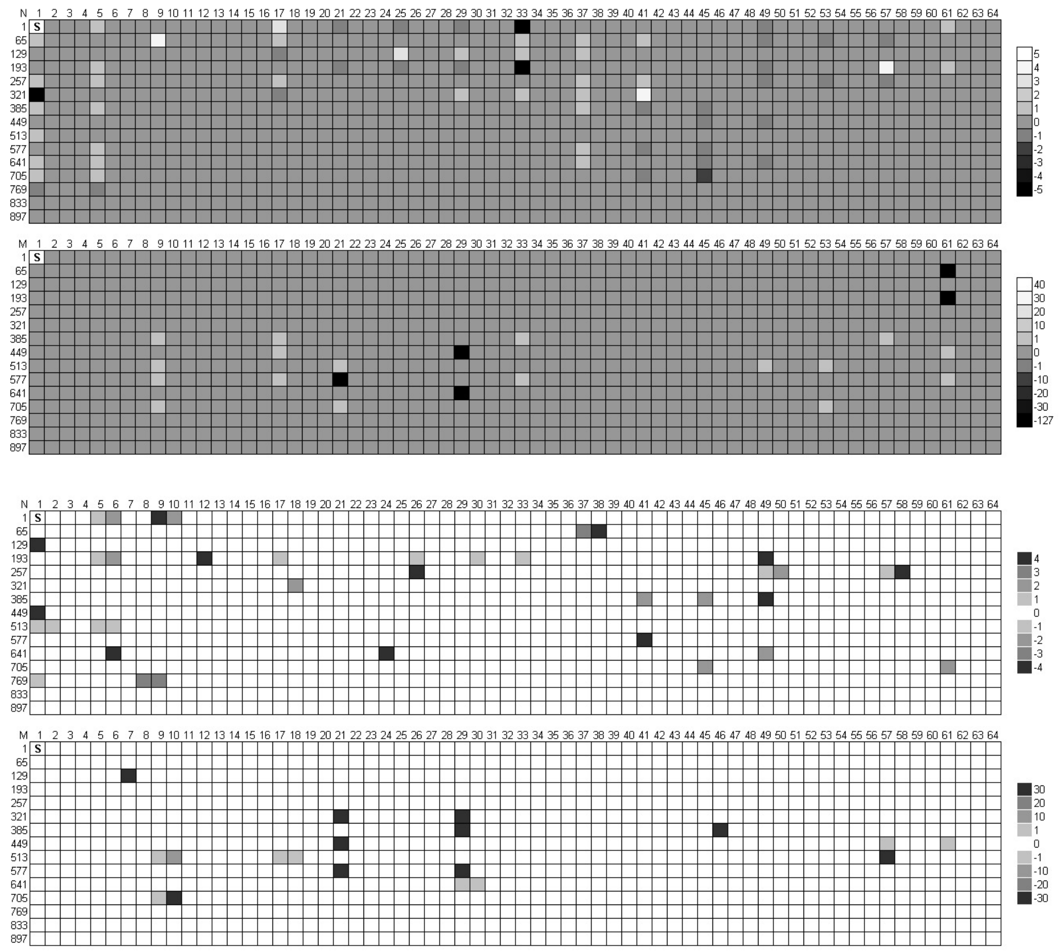


Fig. 35 Initial imitation and corresponding delta vector

Example 2: After 10 automatic parameter estimation cycles the system converges to the following six imitation parameters:

w= 67, y= 82, x= 40, p= 4, r= 3, q= 3.

With these imitation parameters one of the resultant outputs and its corresponding delta vector is given in Figure 36. The resultant Delta_k value is 240.

When the graph in Figure 36, is analyzed it can be seen that the number of different notes have been reduced. This causes the delta vector to be less crowded and the Delta_k value to be moderately high. This is caused by the fact that the averaging parameters are still low causing the system to produce some unintended activities into a note. The time window parameters are also still high combining some consecutive notes into a single note. This situation is where the played musical piece starts to sound like the original one and the imitation is successful.

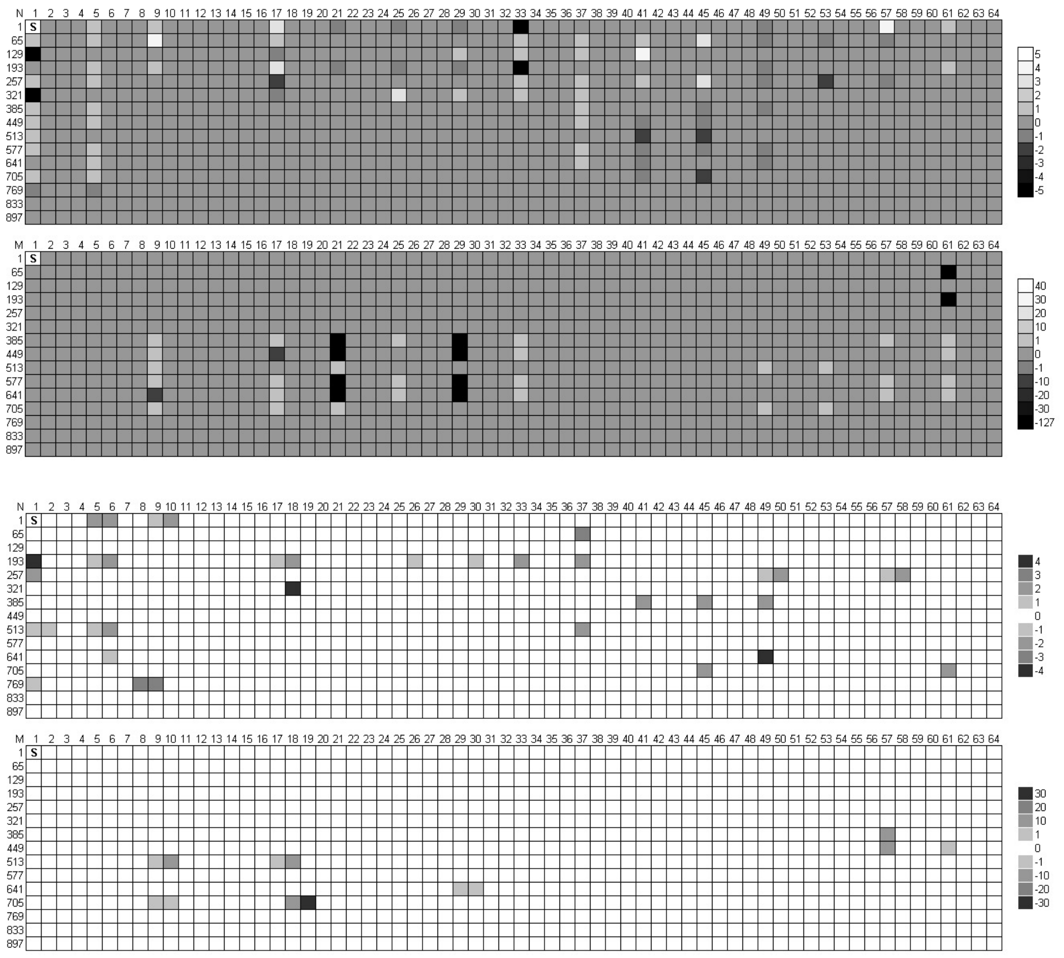


Fig. 36 Resultant imitation after 10 automatic parameter estimation cycles and corresponding delta vector

Example 3: After 25 automatic parameter estimation cycles the system converges to the following six imitation parameters:

w= 81, y= 85, x= 70, p= 2, r= 3, q= 3

With these imitation parameters one of the resultant outputs and its corresponding delta vector is given in Figure 37. The resultant Delta_k value is 146.

When the graph in Figure 37, is analyzed it can be seen that the number of different notes have been reduced significantly. This causes the delta vector to be sparse and the Delta_k value to be nominal. Our experimental studies have showed us that Delta_k values less than 160 for 1920 time slots yield a local minimum. This situation is where the played musical piece sounds like the original one and the imitation is very successful.

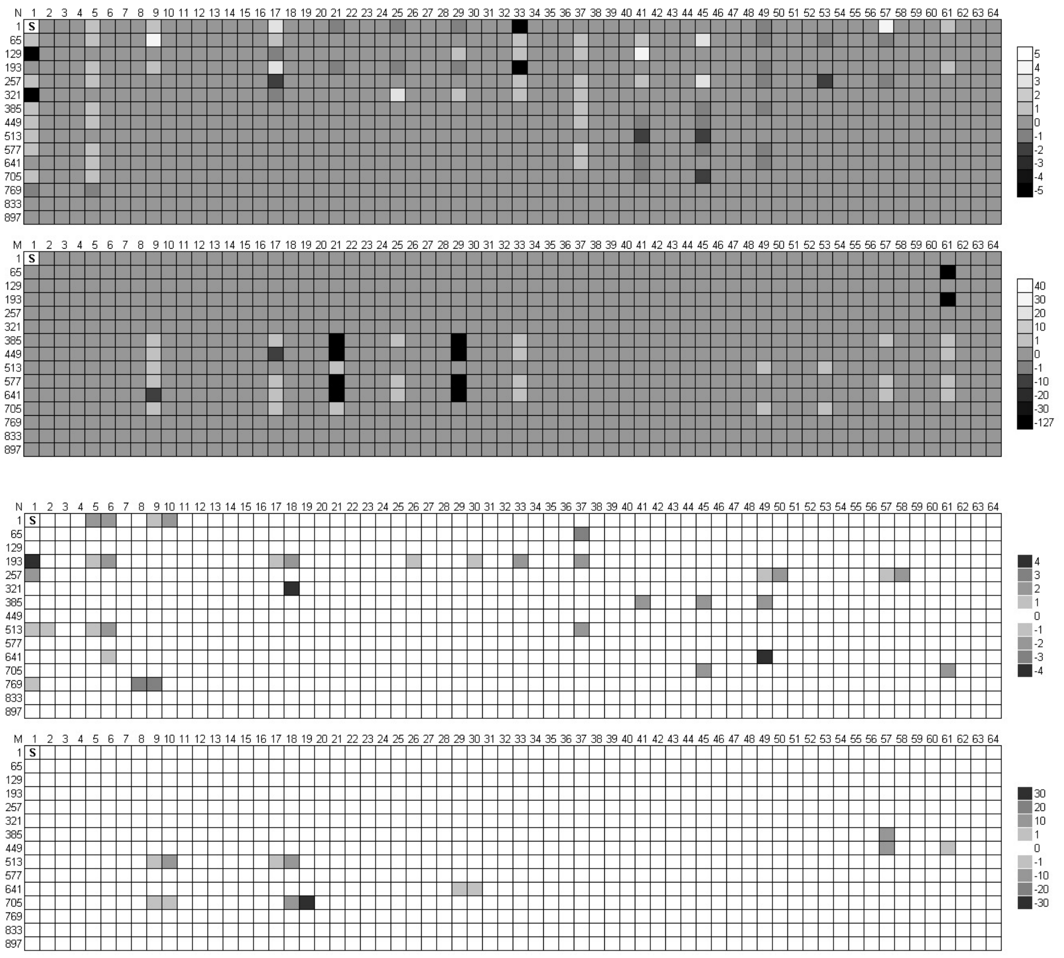


Fig. 37 Resultant imitation after 25 automatic parameter estimation cycles and corresponding delta vector

The next section discusses the effects of imitation parameters on imitation performance in detail.

5.2 Sensitivity Analysis

There are six imitation parameters used in our proposed control system. These parameters are not independent as explained in detail further in this section. The experimental studies showed us that the six imitation parameters can have a range of values generating similar Delta values. Therefore, there is a set of values for each imitation parameter groups of six imitation parameters for a given musical piece, where the resultant Delta values are close and the imitation is of similar quality. This also implies that the automatic parameter estimation process, has no unique value set for minimizing the delta for the six imitation parameters. There is a range of parameter values producing similar quality imitation performances.

The test runs showed us that choices for p, q, r parameter values are more limited than the w, y, x parameters that can attain larger ranges. This is due to the fact that their value is connected to the time granularity (resolution) of the MSR. For a given tempo, the time granularity (resolution) of the MSR can not be less than 1/32th of a note, since this will cause the shortest possible note the system can reproduce to an unacceptable value of 1/16th of a note. The time granularity should not be more than 1/128th of a note, since this only would add the possibility of using 1/64th of a note as the shortest note, but such notes only exist for less than %0.000001 of existing musical scores.

For the test runs of the parameter estimation process, six samples for piano part Ecossais from Ludwig van Beethoven has been recorded by ROMI from two different human teachers.

The effects of different values for the imitation parameters are shown in the following figures. Each graph in these figures have been generated from the imitated piano part music reproduction being compared with the original score. Some imitation parameters are set to fixed values to show the effects of changing other parameters. Delta_k values have been used for one of the human teachers, total number of samples processed is six.

Figures 38 and 39, show how Delta_k values are effected by changes in the main note generator parameters “y” and “w”. Both parameters effect the imitation performance in a similar way. Values below 20 for either parameter generate many notes that are not in the original score resulting in high Delta_k values (bad imitation). If both of these parameters are kept around 70-90 the imitation performance is of acceptable quality. Note that due to the nature of the calculation for Delta_k values, it is not possible to zero out the Delta_k values. The range of Delta_k values are effected by the number of samples processed with larger number of samples resulting in higher Delta_k values. However this does not change the shape of the given graphs with the local minimum still being achieved around 70-90 for these parameters. Values above 95 for either parameter generate less notes than the original score resulting in higher Delta_k values thus degenerating imitation.

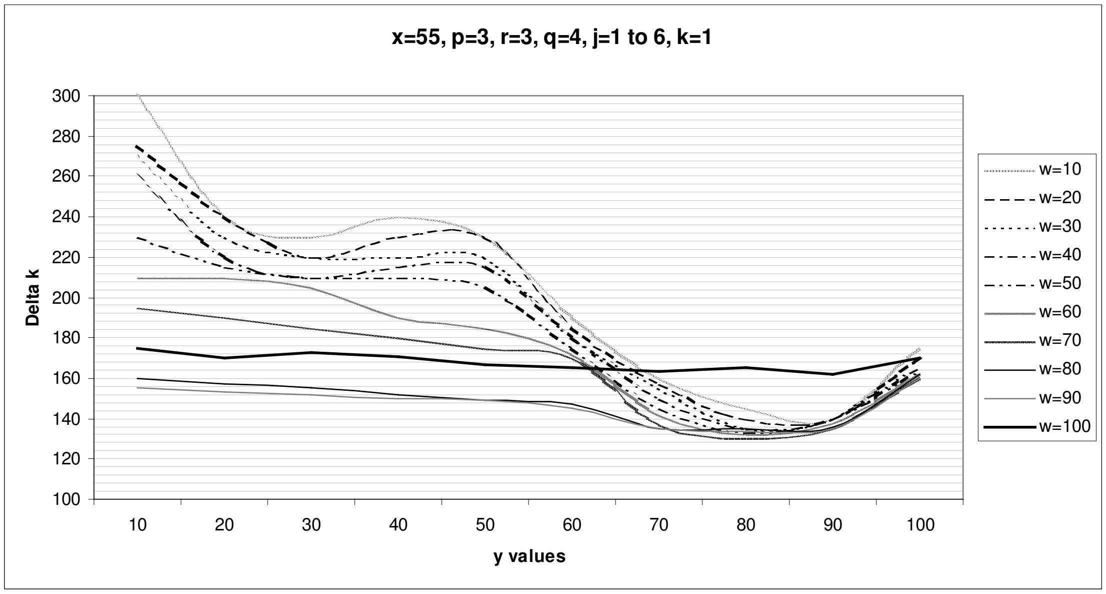


Fig 38. Delta_k for varying “y” values with 10 different “w” values

Figure 40, shows the effects of parameter “x” on imitation performance. This parameter has lower impact on imitation performance compared to “w” and “y” parameters. This is understandable since the imitation algorithm generates notes based on ***N***, ***M*** and ***A*** streams in this order. This results in most of the notes already being produced by the ***N*** and ***M*** streams with ***A*** stream having fewer opportunity to generate a note and effect the imitation performance. For values below 25 this parameter generates notes that are not in the original score thus hindering imitation. For values above 95 it generates fewer notes than the original score and degrading imitation in a different way.

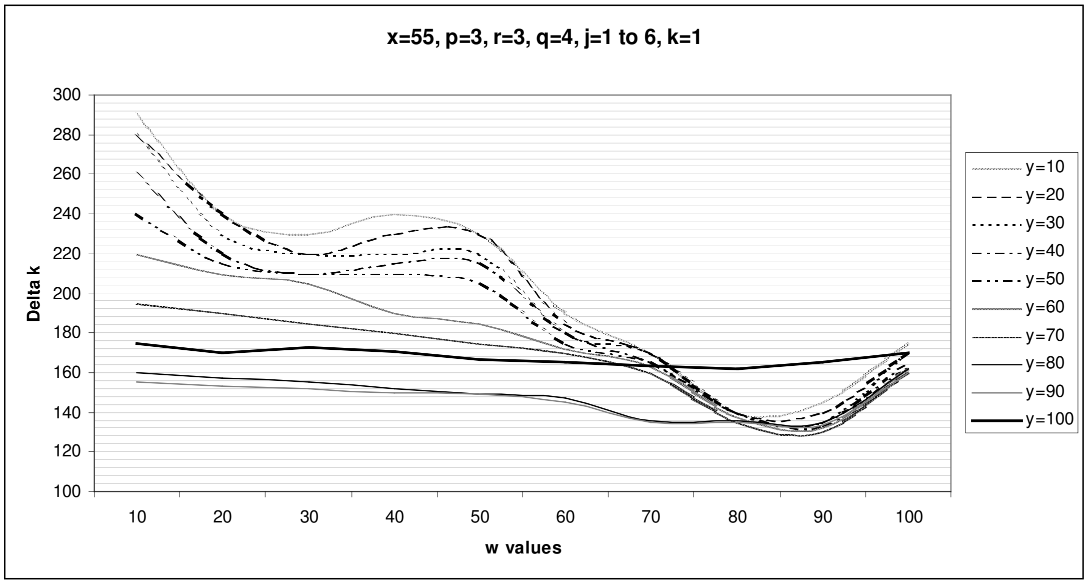


Fig 39. Delta$_k$ for varying “w” values with 10 different “y” values

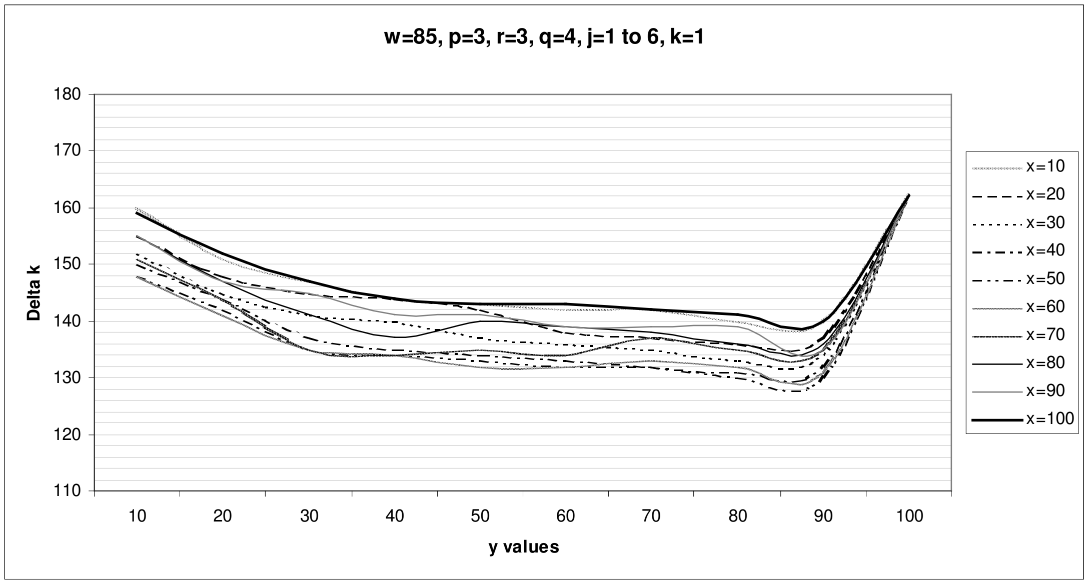


Fig 40. Delta$_k$ for varying “y” values with 10 different “x” values

Figure 41, shows the effects of parameter “p”. Graphs for parameter “r” have the same shape and effect the imitation performance in a similar way as explained here for parameter “p”. This parameter is used by the first note generator using ***N*** streams and has the greatest impact on note production. The value 1 will produce less notes than the original score. Values 2 and 3 yield optimal imitation. Values above 3 produce more notes than the original score by combining two consecutive notes into one, thus increasing Delta$_k$. The second jump in Delta$_k$ at “p” value 8 is due to the fact that more notes that are not in the original score are produced for every note shorter or equal to a quarter note within the time window defined by “p”. Even bigger jumps in Delta$_k$ should be expected for values 12 and 16 for this parameter.

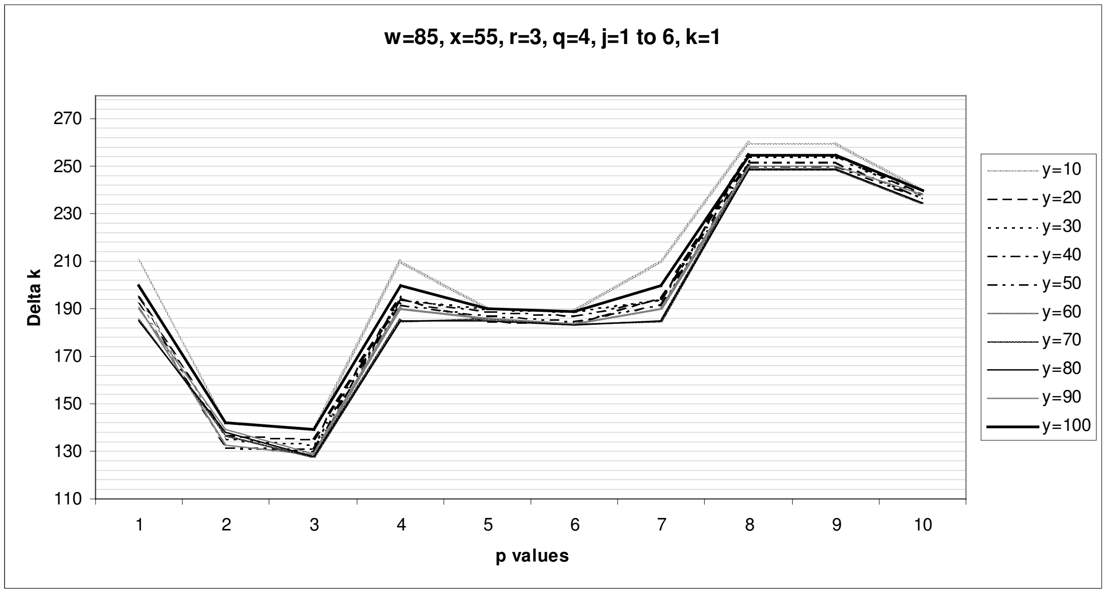


Fig 41. Delta$_k$ for varying "p" values with 10 different "y" values

The examples we have given so far on the effects of the imitation parameters on performance have been in three variables only. The reason for this is that, up to three variables it is possible to plot the overall effect of the variables over their range of attainable values. We have also conducted sensitivity analysis using all six variables using MATLAB. This gives the best values for all six variables resulting in the minimum delta possible. The results for best imitation parameter values resulting in minimum delta for 5 different musical pieces is given in Table 8, below:

Table 8. Best imitation parameter values resulting in minimum delta

k	Delta$_k$	w	y	x	p	r	q
1	126	89	92	87	2	2	2
2	134	87	91	93	2	2	2
3	131	85	92	87	2	3	2
4	126	87	90	90	2	2	3
5	129	89	91	86	2	2	3

The sensitivity analysis shows us that the six imitation parameters are not independent. This is a result of our MSR, the three time window defining parameters p, r and q could have been selected as a single parameter. We wanted to implement more control into the MSR and decided to go with three distinct time window parameters instead of one. However, the sensitivity analysis shows that only a single time window parameter would give similar results. But having three distinct averaging parameters, w, y and x gives us more control over the imitation performance. However, the w, y and x parameters are interdependent in a different way. Parameter x depends on the other two, since the number stream ***A,*** it works on, is a direct function of the other two number streams ***N*** and ***M***. Number streams ***N*** and ***M*** are interdependent on each other since they are coupled in another domain: music.

Using the Delta Vectors lead to the possibility of user profiling by ROMI. In fact the data is readily available in the MSR of each sample to assess many aspects of the musician playing the musical part.

When listening to the Delta$_j$ "sound" we noticed that most high delta results were due to a constant error in the timing of the notes, which indicates that the musician is playing with a wrong tempo. If such samples can be processed to change their tempo by a fixed amount then their corresponding Delta$_j$ would be much less. This gives the first output of a user profiling application.

When Delta_j of a sample is minimized by changing its tempo, then the resultant amount of tempo change is communicated to the user to correct his/her playing style.

Analyzing the imitation, if Delta_j values do not decrease by big proportions when tempo is changed, then such samples are in fact imitated from start to finish with the wrong “key”. This resulted in many of the notes being played by a higher or lower pitch. Even though the timing of the notes are good enough, due to this constant pitch difference, the Delta_j of the signal remains high. Here feedback from ROMI imitation can make ROMI change the key of its musical reproduction.

5.3 Survey Results

Some improvisation algorithms produce high performance results for some musical parts while they do not for others. There seems to a implicit link between the musical parts unseen musical properties and the improvisation algorithm in use. The success of EMI may be in the fact that it only works with Bach chorales. Therefore, we believe that a joint study in musical arts and computer programming aiming to model implicit musical attributes for improvisation can be of value. Our proposed control architecture does not delete the high delta samples and it also does not delete the generated improvisations with low ratings. This adds a memory to our system. If our sole goal was to imitate then these samples would be unnecessary. The memory however can be used to generate more improvisations.

The resolution of the n-gram effects the improvisation quality of ROMI in the following way: If a 1-gram is used then there is a limited number possible states. Of course our approach where note lengths and loudness changes are effecting the improvisation results adds to the possible states. The quality of improvisations are then found to be acceptable. If a 2-gram is used then the number of possible states reach a large number that is almost unmanageable due to our addition of note lengths and loudness changes. The resultant improvisations are found to be more vivid. However, some of the improvisation results deviate from the original music in such a way that they sound incompatible. If a 3-gram is used there is no way to manage the number of possible states due to our addition of note lengths and loudness changes. Most of the possible states attain so low probabilities that they almost do not contribute at all. One way to ease this problem is to expand the original scores collection. However, with each addition some possible states get more probability but the problem becomes worse since the new samples are more of these very low probability states. We believe that if time can be spent to add a massive amount of music in the collections, than the system will reach a saturation point where the addition of new music increases the state possibilities without adding many new possible states.

Since improvisation is a subjective topic, a tool for evaluating ROMI's improvisation results is necessary. We followed a similar approach that other studies have followed so far; gathering a listener group and asking them to rate ROMI's different improvisations. The average of ratings given by the group of listeners are used as the rating for a given improvisation sample. This approach is not able to pinpoint musically superior improvisations since the listener group is usually not composed of professional musicians. We used a group of 20 students from METU in our studies. We placed some original improvisation recordings from well-known composers into the listening evaluations to control the responses of the listeners. A rating was considered as valid only if it included high ratings for these well-known human improvisations. The questionnaire used for these ratings is given in Appendix B.

The results of the survey sessions are presented in Figure 39, below:

Lugwig van Beethoven`s Ecossais, monophonic

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	**AV1**	15	16	17	18	19	20	**AV2**	**AVT**
IRIP applied without restrictions	3	4	5	4	7	3	5	4	5	7	3	6	4	5	**4,64**	4	3	4	5	3	3	**3,67**	**4,15**
IRIP applied with the 5 restrictions	7	5	5	4	6	6	6	5	6	8	6	7	5	6	**5,86**	5	6	6	6	7	8	**6,33**	**6,10**
Random notes as improvisation without restrictions	4	3	2	3	2	3	2	3	1	2	3	1	3	4	**2,57**	1	1	2	1	1	1	**1,17**	**1,87**
Random notes as improvisation with tonal context	3	4	5	5	4	4	3	5	4	5	4	3	3	4	**4,00**	2	1	2	2	1	2	**1,67**	**2,83**
3-gram Markov Chain	4	5	7	6	8	8	7	7	6	7	8	8	7	6	**6,71**	8	9	7	8	7	8	**7,83**	**7,27**
IRIP applied with patching and 5 restrictions and tonal context	7	7	4	5	6	7	7	8	8	9	8	7	8	7	**7,00**	10	9	10	10	9	10	**9,67**	**8,23**
IRIP applied with the 5 restrictions and tonal context	5	5	6	6	8	7	6	8	7	8	7	8	8	6	**6,79**	9	9	8	9	8	9	**8,67**	**7,43**
Do you have musical background	N	N	N	N	N	N	N	N	N	N	N	N	N	N		Y	Y	Y	Y	Y	Y		

A Fine Romance, 6 polyphonic

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	**AV1**	15	16	17	18	19	20	**AV2**	**AVT**
3-gram Markov Chain	7	6	6	7	6	7	7	5	6	8	6	6	5	7	**6,33**	7	7	7	8	9	9	**7,83**	**7,08**
IRIP applied with the 5 restrictions, without velocity information	5	4	5	4	7	3	6	4	5	6	4	7	3	5	**4,92**	6	5	5	6	7	7	**6,00**	**5,46**
IRIP applied with the 5 restrictions, with velocity information	7	6	7	5	8	6	7	7	6	7	5	8	6	6	**6,50**	8	7	8	8	9	8	**8,00**	**7,25**
Random notes and random velocity information as improvisation without restrictions	2	2	2	3	2	3	2	3	3	2	2	1	3	3	**2,42**	1	1	1	1	1	1	**1,00**	**1,71**
Random notes with tonal context and in range velocity information as improvisation	2	4	3	4	4	4	3	4	4	2	5	3	4	6	**3,83**	2	1	1	1	2	1	**1,33**	**2,58**
IRIP applied with patching, with velocity information	7	6	7	8	7	8	8	7	6	8	7	7	7	6	**7,17**	8	7	9	8	9	7	**8,00**	**7,58**
IRIP applied with patching, with velocity information, with tonal context	8	7	7	8	9	7	8	7	7	7	8	7	8	8	**7,58**	9	8	8	9	8	9	**8,50**	**8,04**
Do you have musical background	N	N	N	N	N	N	N	N	N	N	N	N	N	N		Y	Y	Y	Y	Y	Y		

Fig 42. The results of the survey

For the survey, 7 different improvisation approaches in a wide range, starting from the simplest form of random notes to the most elaborate ones, have been implemented both for monophonic and polyphonic musical pieces.

A 3-gram Markov Chain based improvisation have been implemented with reference to [18], as explained in detail in section 4.4.3 of chapter 4, as a benchmark to compare our proposed improvisation approaches.

There are 20 musical piece recordings for the monophonic musical piece and 12 recordings for the 6 polyphonic musical piece. These are used as the musical piece sample collection to determine the 3-gram probabilities. Increasing this collection would improve the 3-gram results slightly more. Our proposed MSR is used for implementing the 3-gram Markov Chain. The use of 3-gram Markov Chains is computationally heavy if the collection size grows and most practical applications use 2-gram Markov Chains. Our implementation of the 3-gram Markov Chain includes velocity data of the musical pieces. This adds to the quality of the improvisation generated but adds heavily on the computational side as well.

The monophonic and the 6 polyphonic musical pieces are evaluated separately in order to compare the effects of different improvisation approaches for monophonic and polyphonic music.

The results of our survey sessions can be summarized as follows:

1. The opinions of students attending the survey that have a musical background have much more pronounced differences when evaluating between ROMI's various implementations of improvisation. Especially random notes are much more penalized than students without a musical background. This sharp difference in random note valuations can be due to the fact that students without a musical background might have a limited idea on what to expect from an improvisation.
2. Tonal context implementation, which restricts the MSR from sounding certain note values in the presence of other certain note values derived from existing musical knowledge, improves even the random note valuations. The improvement is also visible in structured improvisation approaches. Please note that, any n-gram based Markov Chain approach inherently implements tonal context usage. This result shows that tonal context is an essential element for successful improvisation even for different improvisation approaches.
3. The five restrictions, given in section 4.4.1 of chapter 4, that we are proposing as a result of our experimentation with the control architecture of ROMI improve the IRIP results significantly. Without these restrictions the improvisation quality of the IRIP is very poor. Therefore, it can be concluded that for a successful improvisation implementation, the length of the improvisation and its starting point in the whole musical piece is of high importance.
4. Stripping out the velocity information from our proposed MSR based improvisation approaches degrade the improvisation quality. This is due to the close coupling of the imitation parameters that are used in defining our proposed MSR. Taking out the loudness change information degrades the remaining note change information and thus improvisation is also degraded.
5. The 3-gram Markov Chain based improvisations are better than the IRIP applied with the 5 restrictions for monophonic music while it is almost the same quality for polyphonic music. This is due to the fact that in polyphonic music there are more monophonic sounds that maintain the overall tempo of the musical piece, even if some of the monophonic parts loose tempo due to the introduction of improvisation parts. When tonal context is applied on IRIP with 5 restrictions for monophonic music, the improvisation quality matches that of the 3-gram Markov Chain based improvisation. Since n-gram based Markov Chain approaches inherently use tonal context this closes the gap between our proposed IRIP and a 3-gram Markov Chain implementation.
6. Best results, that are better than the 3-gram Markov Chain based improvisation, are achieved when patching is added on our proposed improvisation approach. This is due to the fact that, any Markov Chain based approach will start and stop the improvisation at given points in the whole musical piece; while through patching, our approach selects the most proper location in the musical piece to start the improvisation by analyzing the whole musical piece.

CHAPTER 6

CONCLUSIONS

ROMI imitates musical pieces as intended and its control architecture produces acceptable quality improvisations. The control architecture based on our MSR works well with both simulations done on a sequencer and with real world acoustic musical hardware. Our proposed MSR is easy to implement and is not computationally heavy. Adding polyphony and loudness information into the improvisation algorithms, employing our proposed MSR, is our main contribution to the existing knowledge on computer generated music improvisation.

Sample owners' identification is achieved with limited success, since this identification does not involve any style analysis of the human teacher recording the sample. Sample owners' identification functionality, relies only on systematic timing and note errors. This identification works better if inexperienced and experienced users are identified first and the threshold for the event indicator chosen for detecting systematic errors for inexperienced and experienced users is adjusted separately. The limit on the number of distinct users that can use this functionality can be increased by increasing the sample sets for each musical pieces in the collection. Communicating systematic timing and note errors to human players is a more generally applicable and usable tool that relies on the same principles of sample owners' identification.

The effects of imitation parameters that are used to control the imitation and improvisation processes have been studied in detail. The results that are presented show the mutual relationship between these parameters and how they can be used to fine tune the system.

Our IRIP based improvisation works comparatively better than the n-gram based Markov Chain approach that has been used as a benchmark. Tonal contex limitations are used adding to the quality of the improvisations as indicated by the survey results.

Patching improvisation parts into the imitation of musical pieces is adding to the quality of the improvisations as indicated by the survey results. Both the improvisation parts and imitation parts are considered as signals and patching works by matching the slopes of imitation and improvisation signals at the entry and exit points of the improvisation.

For future work, the IRIP with patching and tonal context can be implemented as a plugin for well established computer synthesizers as an improvisation tool. Since our MSR is computer friendly, in the sense that it dose not take much memory or processing power to work, we believe it is a viable candidate for computer generated improvisations. ROMI can be equipped with D/A converters for each solenoid for total velocity control. The "Original Scores Collection" can be enriched to include many more musical pieces giving ROMI a wider repertoire.

REFERENCES

[1] K.K. Aydın, A. Erkmen, I. Erkmen, ‘Improvisation Based on Relaxation of Imitation Parameters by a Robotic Acoustic Musical Device’, Engineering Letters, Volume 20, Issue 1, March 2012, pp. 28-41.

[2] K.K. Aydın, A. Erkmen, I. Erkmen, ‘Improvisation Based on Imitating Human Players by a Robotic Acoustic Musical Device’, Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering and Computer Science 2011, WCECS 2011, 19-21 October, 2011, San Francisco, USA, pp. 331-336.

[3] K.K. Aydın, A. Erkmen, `Musical State Representation for Imitating Human Players by a Robotic Acoustic Musical Device`, IEEE International Conference on Mechatronics, Istanbul, Turkey, 2011, pp. 126-131.

[4] K. M. Robert and D. R. Morrison, `A Grammatical Approach to Automatic Improvisation`, Fourth Sound and Music Conference, Lefkada, Greece, 2007, pp. 221-226.

[5] W. Rachael, `Jamming Robot Puts the Rhythm into Algorithm`, Science Alert Magazine, 2008.

[6] R. Rafael, A. Hazan, E. Maestre and X. Serra. ‘A genetic rule-based model of expressive performance for jazz saxophone’, Computer Music Journal, Volume 32, Issue 1, 2008, pp. 38-50.

[7] J. Biles, ‘GenJam: A genetic algorithm for generating jazz solos’. Proceedings of the International Computer Music Conference, Aarhus, Denmark, 1994, pp. 131-137.

[8] G. Mark, Jazz Styles: History & Analysis, Prentice-Hall, inc. Englewood Cliffs, NJ, 1985.

[9] J. Aebersold, How to Play Jazz and Improvise. New Albany, NJ:Jamey Aebersold, 1992.

[10] D. Baker, Jazz Improvisation :A Comprehensive Method of Study For All Players .Bloomington, IN:Frangipani Press, 1983 .

[11] M. Goto, R. Neyama, `Open RemoteGIG: An open-to-the public distributed session system overcoming network latency`, IPSJ Journal 43, 2002, pp. 299-309.

[12] K. Nishimoto, `Networked wearable musical instruments will bring a new musical culture`, Proceedings of ISWC, 2001, pp.55-62.

[13] T. Terada, M. Tsukamoto, S. Nishio, `A portable electric bass using two PDAs`, Proceedings of IWEC, Kluwer Academic Publishers 2002, pp. 286-293.

[14] S. Fels, K. Nishimoto, K. Mase, `MusiKalscope: A graphical musical instrument`, IEEE Multimedia 5, 1998, pp.2 6-35.

[15] E. Cambouropoulos, T. Crawford and C. Iliopoulos, ‘Pattern Processing in Melodic Sequences: Challenges, Caveats &Prospects’, Proceedings from the AISB ’99 Symposium on Musical Creativity, Edinburgh, Scotland, 1999, pp. 42-47.

[16] A. Yatsui, H. Katayose, `An accommodating piano which augments intention of inexperienced players`, Entertainment Computing: Technologies and Applications, 2002, pp. 249-256.

[17] M. C. Mozer, `Neural Network Music Composition by Prediction: Exploring the Benefits of Psychoacoustic Constraints and Multi-scale Processing`, Connection Science, 6 (2-3), 1994, pp. 247-280.

[18] D. Cope, Computer Models of Musical Creativity. The MIT Press: Cambridge, MA, 2005.

[19] D. Cope, Virtual Music: Computer Synthesis of Musical Style. The MIT Press: Cambridge, MA, 2001.

[20] P. Gaussier, S. Moga, J. P. Banquet and M. Quoy. `From perception-action loops to imitation processes: A bottom-up approach of learning by imitation`, Applied Artificial Intelligence Journal, Special Issue on Socially Intelligent Agents. 12(7 8), 1998, pp. 701-729.

[21] Y. Kuniyoshi, M. Inaba and H. Inoue. `Learning by watching: Extracting reusable task knowledge from visual observation of human performance`, IEEE Transactions on Robotics and Automation, vol. 10, no. 6, 1994, pp. 799- 822.

[22] M. Goto, I. Hidaka, H. Matsumoto, Y. Kuroda, Y. Muraoka, `A jam session system for interplay among all players`, Proceedings of the International Computer Music Conference, 1996, pp. 346-349.

[23] T. Eliassi-Rad and J.Shelvik,'A System for Building Intelligent Agents that Learn to Retrieve and Extract Information'. User Modeling and User-Adapted Interaction, Special Issue on User Modeling and Intelligent Agents, 2003, pp. 35-88.

[24] Y. Aono, H. Katayose, S. Inokuchi, `An improvisational accompaniment system observing performer's musical gesture`, Proceedings of the International Computer Music Conference, 1995, pp. 106-107.

[25] A. Billard & K. Dautenhahn, `Grounding communication in autonomous robots: an experimental study`, Robotics and Autonomous Systems. No. 24, Vols. 1-2, 1998, pp. 71-81.

[26] K. Dautenhahn, `Getting to know each other--artificial social intelligence for autonomous robots`, Robotics and Autonomous Systems, 16(2 4), 1995, pp. 333-356.

[27] J. Demiris & G. Hayes, `Active and passive routes to imitation`, Proceedings of AISB'99, Edinburgh, April 1999, pp. 81-87.

[28] S. Schaal. `Robot learning from demonstration`, In International Conference on Machine Learning, San Francisco, USA, 1997, pp. 12-20.

[29] P. George and G. Wiggins, `AI Methods for Algorithmic Composition: A Survey, a Critical View and Future Prospects`, Proceedings of AISB Symposium on Musical Creativity, 1999, pp. 146-152.

[30] K. Youngmoo, W. Chai, R. Garcia, and B. Vercoe, `Analysis of a Contour-based representation for Melody`, Proceedings of International Symposium on Music Information Retrieval, 2000, pp. 312-317.

[31] S. Schaal, (1999). Is imitation learning the route to humanoid robots?. Trends in Cognitive Sciences, 3(6), pp. 233–242.

[32] D. M.. Wolpert, & Kawato, M. (1998). Multiple paired forward and inverse models for motor control. Neural Networks, 11, pp. 1317–1329.

[33] P. Gaussier, Moga, S., Banquet, J. P., & Quoy, M. (1998). From perception-action loops to imitation processes: A bottom-up approach of learning by imitation. Applied Artificial Intelligence, 1(7), pp. 701–727.

[34] M. J. Mataric, (2002). Imitation in animals and artifacts, chap. Sensory-Motor Primitives as a Basis for Learning by Imitation: Linking Perception to Action and Biology to Robotics, pp. 392–422.

[35] M. I. Jordan, & Rumelhart, D. E. (1992). Forward models: Supervised learning with a distal teacher. Cognitive Science, 16, pp. 307–354.

[36] M. Haruno, Wolpert, D. M., & Kawato, M. (2001). MOSAIC model for sensorimotor learning and control. Neural Comp., 13(10), pp. 2201–2220.

[37] Y. Demiris, & Hayes, G. (2002). Imitation in animals and artifacts, chap. Imitation as a dual-route process featuring predictive and learning components: a biologicallyplausible computational model, pp. 327–361.

[38] A. Cangelosi, & Riga, T. (2006). An embodied model for sensorimotor grounding and grounding transfer: Experiments with epigenetic robots. Cognitive Science, 30(4), pp. 673–689.

[39] J. Tani, Ito, M., & Sugita, Y. (2004). Self-organization of distributedly represented multiple behavior schemata in a mirror system: Reviews of robot experiments using RNNPB. Neural Networks, 17, pp. 1273–1289.

[40] A. Tidemann, & Demiris, Y. (2008). Groovy neural networks. In 18th European Conference on Artificial Intelligence, Vol. 178, pp. 271–275.

[41] C. Saunders, Hardoon, D., Shawe-Taylor, J., & Widmer, G. (2008). Using string kernels to identify famous performers from their playing style. Intelligent Data Analysis, 12(4), pp. 425–440.

[42] A. Tobudic, & Widmer, G. (2005). Learning to play like the great pianists.. In Kaelbling, L. P., & Saotti, A. (Eds.), IJCAI, pp. 871–876.

[43] F. Pachet, (2002). Interacting with a musical learning system: The continuator. In ICMAI '02: Proceedings of the Second International Conference on Music and Artificial Intelligence, pp. 119–132.

[44] R. L. de Mantaras, & Arcos, J. L. (2002). AI and music from composition to expressive performance. AI Mag., 23(3), pp. 43–57.

[45] C. Raphael, (2003). Orchestra in a box: A system for real-time musical accompaniment. In IJCAI workshop program APP-5, pp. 5–10.

[46] R. Pfeifer, & Scheier, C. (2001). Understanding Intelligence. MIT Press, Cambridge, MA,USA. Illustrator-Isabelle Follath.

[47] A. Tidemann, (2008). Using multiple models to imitate the YMCA. In Agent and Multi-Agent Systems: Technologies and Applications, Vol. 4953 of LNAI, pp. 783–792.

[48] A. Tidemann, & Ozturk, P. (2007). Self-organizing multiple models for imitation: Teaching a robot to dance the YMCA. In IEA/AIE, Vol. 4570 of Lecture Notes in Computer Science, pp. 291–302.

[49] Y. Demiris, & Khadhouri, B. (2006). Hierarchical attentive multiple models for execution and recognition of actions. Robotics and Autonomous Systems, 54, pp. 361–369.

[50] M. Kawato, (1990). Feedback-error-learning neural network for supervised motor learning. In Eckmiller, R. (Ed.), Advanced neural computers, pp. 365–372.

[51] R. C. Miall, (2003). Connecting mirror neurons and forward models.. Neuroreport, 14(17), 2135–2137.[53] Kawato, M. (1999). Internal models for motor control and trajectory planning. Current Opinion in Neurobiology, 9, pp. 718–727.

[52] J. Williams, Whiten, A., Suddendorf, T., & Perrett, D. (2001). Imitation, mirror neurons and autism. Neuroscience and Biobehavioral Reviews, 25(4), pp. 287–295.

[53] T. Sobh, K. Coble, B.Wang, 'Experimental Robot Musician', Mobile Robots, Towards New Applications, University of Bridgeport, USA, 2003, pp. 545-558.

[54] T. Rogers, S. Barton, S. Kemper, 2007, www.expressivemachines.org

[55] www.apple.com/logicpro

[56] www.midiworld.com/beethoven.htm

APPENDIX A

MUSIC TERMINOLOGY

beat:	a song’s underlying pulse.
blend/mashup:	a song or composition created by blending two or more pre-recorded songs, usually by overlaying the vocal track of one song seamlessly over the instrumental track of another.
chord:	three or more musical tones sounded simultaneously.
chromatic scale:	the pitches that result when an octave is broken up into twelve logarithmically equidistant parts.
chromaticism:	refers to the inclusion of tones outside of a particular key or scale.
consonance:	roughly defined as harmonies whose tones complement and increase each others' resonance, and dissonance as those which create more complex acoustical interactions (called 'beats'). Another manner of thinking about the relationship regards stability; dissonant harmonies are sometimes considered to be unstable and to "want to move" or "resolve" toward consonance. However, this is not to say that dissonance is undesirable. A composition made entirely of consonant harmonies may be pleasing to the ear and yet boring because there are no instabilities to be resolved.
duration:	refers to how much time a sound event occupies.
enharmonic distinction:	theoretically, there is a difference between a sharped and a flatted pitch,
harmony:	the structure of music with respect to the progression of its chords. Harmony is the study of vertical sonorities in music. Vertical sonority refers to considering the relationships between pitches that occur together; usually this means at the same time, although harmony can also be implied by a melody that outlines a harmonic structure.
heterophony:	a texture characterized by the simultaneous variation of a single melodic line. Such a texture can be regarded as a kind of complex monophony in which there is only one basic melody, but realized at the same time in multiple voices, each of which plays the melody differently.
homophony:	a texture in which two or more parts move together in harmony, the relationship between them creating chords. This is distinct from polyphony, in which parts move with rhythmic independence, and monophony, in which all parts (if there are multiple parts) move in parallel rhythm and pitch.
improvisation:	the creation of spontaneous music. Act of instantaneous composition by performers, where compositional techniques are employed with or without preparation.
interval:	the distance between any two adjacent pitches in a melody.

jazz:	Travis Jackson has proposed a definition: “it is music that includes qualities such as swinging, improvising, group interaction, developing an individual voice, and being open to different musical possibilities.”
key/modulation/tonality:	these terms are explained together because their definitions are interrelated. Key relates to the predominant Major or Minor Scale associated with a tonal piece of music. Modulation relates to the change from one key or ‘tonal center’ to another within a piece of music. Tonality refers to the organization of a melody and/or harmony in relation to a tonic, which affects which tones will seem more consonant/stable or dissonant/instable.
melody:	a series of notes sounding in succession. The notes of a melody are typically created with respect to pitch systems such as scales or modes.
meter:	a song’s predominant repetitive beat grouping (e.g., in rock-n-roll, every fourth beat typically receives the most emphasis).
MIDI:	Musical Instrument Digital Interface.
modal:	with respect to improvisation, this term usually indicates that the underlying harmony only contains several chords that do not change very quickly.
monophonic:	music composed of a single voice.
monophony:	the simplest of textures, consisting of melody without accompanying harmony. This may be realized as just one note at a time, or with the same note duplicated at the octave
musical notation:	symbolic representation of music. Historically, and in the narrow sense, this is achieved with graphic symbols. Nowadays computer file formats are becoming important in this area.
music perception:	the field of music cognition focuses on how the mind makes sense of music as it is heard. It also deals with the related question of the cognitive processes involved when musicians perform music. Like language, music is a uniquely human capacity that arguably played a central role in the origins of human cognition.
octave:	the relationship that occurs when the ratio between two pitches’ fundamental frequencies is two.
pitch:	a subjective sensation in which a listener assigns perceived tones to notes on a musical scale based mainly on the frequency of vibration, with a lesser relation to sound pressure level. The pitch of a tone typically rises as frequency increases.
pitch class:	by ignoring in which octave a tone resides, pitch class maps a tone onto one of the twelve semitones that are contained within each octave.
pitch tracking/rhythmic quantization/tempo tracking:	these terms are explained together because their definitions are interrelated. Pitch tracking is the conversion of an audio signal into a stream of discrete pitches and rests. Tempo tracking is the corresponding segmentation of this stream into beat locations. Rhythmic quantization describes the mapping from arbitrary inter-beat locations onto discrete, hierarchical rhythms.
polyphony:	a texture consisting of two or more independent melodic voices, as opposed to music with just one voice (monophony) or music with one dominant melodic voice accompanied by chords.
rest:	a silence (no pitch is sounding).
riff/lick/motive:	each of these terms to refer to short melody fragments.
rhythm:	refers to the arrangement of sound events hierarchically in time and relates to the discrete perception of rhythm. Rhythm is the arrangement of sounds

	in time. The time signature or meter signature specifies how many beats are in a measure and which value of written note is counted and felt as a single beat.
scale:	notes can be arranged into different scales and modes. Western music theory generally divides the octave into a series of 12 notes that might be included in a piece of music. This series of twelve notes is called a chromatic scale. In the chromatic scale, each note is called a half-step or semitone.
semitone:	the distance between any two adjacent pitches in the chromatic scale.
solo:	a consecutive sequence of bars of solo (a spontaneously improvised melody that occupies one metrical group of beats).
sound event:	a pitch or rest and its corresponding duration.
syncopation:	this term refers to rhythms whose durations span beat boundaries.
tempo:	the rate at which beats pass.
timbre:	the quality of a musical note or sound or tone that distinguishes different types of sound production, such as voices and musical instruments. The physical characteristics of sound that mediate the perception of timbre include spectrum and envelope.
tonality:	a system of music in which specific hierarchical pitch relationships are based on a key "center", or tonic.
tonic:	relating to or based on the first tone of a scale.
transcription:	amounts to converting an audio signal into a musical score.
velocity/loudness:	dB value of sound pressure of a note with regard to whispering noise.

APPENDIX B

IMPROVISATION SURVEY FORM

Dear Listener,

We thank you for your participation in our Improvisation Survey. Please fill out the following form by ticking your rating (circle a number between 1-10, 1 being very bad and 10 being very good) at the end of each musical piece. The musical pieces are 40 seconds long. Please do not rate the first two musical pieces. They are being played just to let you get accustomed with the procedure.

Musical Piece 1 (do not rate)

Musical Piece 2 (do not rate)

Musical Piece 3

Very Bad	1	2	3	4	5	6	7	8	9	10	Very Good

Musical Piece 4

Very Bad	1	2	3	4	5	6	7	8	9	10	Very Good

Musical Piece 5

Very Bad	1	2	3	4	5	6	7	8	9	10	Very Good

Musical Piece 6

Very Bad	1	2	3	4	5	6	7	8	9	10	Very Good

Musical Piece 7

Very Bad	1	2	3	4	5	6	7	8	9	10	Very Good

Musical Piece 8

Very Bad	1	2	3	4	5	6	7	8	9	10	Very Good

Musical Piece 9

Very Bad	1	2	3	4	5	6	7	8	9	10	Very Good

Do you have musical background: YES NO

CURRICULUM VITAE

PERSONAL INFORMATION
Surname, Name: Aydın, Kubilay Kaan
Nationality: Turkish (TC)
Date and Place of Birth: 1 August 1967 , Ankara
Marital Status: Single
Phone: +90 532 332 35 56
email: kaan@7s.com.tr

EDUCATION

Degree	Institution	Year of Graduation
MS	METU Electrical and Electronics Engineering	1992
BS	METU Electrical and Electronics Engineering	1989
High School	Atatürk Anadolu High School, Ankara	1985

WORK EXPERIENCE

Year	Place	Enrolment
2010- Present	Gelişmiş Video Teknolojileri Ltd Şti	Project Manager
2006-2010	7S İleri Teknoloji Ürünleri Ltd. Şti	Project Manager
2005-2006	Fujitsu-Siemens Computers	Regional Manager
1999-2005	Siemens Business Services	Project Manager
1998-1999	SEBIT	Director

FOREIGN LANGUAGES

Advanced English

PUBLICATIONS

Aydın K. K., Erkmen A. and Erkmen İ., ‘:Improvisation Based on Relaxation of Imitation Parameters by a Robotic Acoustic Musical Device’, Engineering Letters, Volume 20, Issue 1, March 2012, pp. 28-41.

Aydın K. K., Erkmen A. and Erkmen İ., “Improvisation Based on Imitating Human Players by a Robotic Acoustic Musical Device”, IAENG International Conference on Intelligent Automation and Robotics 2011, October 2011, pp. 331-336. (received Best Paper Award)

Aydın K. K. and Erkmen A., “Musical State Representation for Imitating Human Players by a Robotic Acoustic Musical Device”, IEEE International Conference on Mechatronics 2011, April 2011, pp. 126-131.

Aydın K. K., "Merging Self-Motion Topology Knowledge of Multiple Cooperating Manipulators", 13th IFAC World Congress, July 1996, pp. 391-396.

Aydın K. K., "Work-Space Design for Fault Tolerant Manipulators", 13th IFAC World Congress, July 1996, pp. 313-318.

Aydın K. K., "Fuzzy Representation of Grasping Modes", 8th Mediterranean Electrotechnical Conference, May 1996, pp. 214-219

Aydın K. K., "Implications of Faults in Manipulator Joints on the Work-Space", 8th Mediterranean Electrotechnical Conference, May 1996, pp. 267-272

Aydın K. K., "Fuzzy Logic, Grasp Pre-shaping for Robot Hands", The Joint Third International Symposium on Uncertainty Modeling and Analysis, and The Annual Conference of the North American Fuzzy Information Processing Society, September 1995, pp. 520-523

Aydın K. K. and Kocaoğlan E., "Genetic Algorithm Based Redundancy Resolution of Robot Manipulators", The Joint Third International Symposium on Uncertainty Modeling and Analysis, and The Annual Conference of the North American Fuzzy Information Processing Society, September 1995, pp. 322-327

Aydın K. K. and Kocaoğlan E., "A Knowledge-Based System for Redundancy Resolution and Path Planning, Using Self-Motion Topology of Redundant Manipulators", 7th IEEE International Conference on Tools with Artificial Intelligence, November 1995, pp.282-285

Aydın K. K., "Path Planning and Redundancy Resolution for Planar Redundant Manipulators, Using Artificial Neural Networks", IEEE International Conference on Neural Networks, November 1995, pp.2560-2565.

CERTIFICATES

» The Counsellor Salesperson EKSER - 09.09.2002
Establishing the Relationship. Discovery. Presentation-Defence. Maintaining the Relationship.

» Selling to Senior Executives SIEBEL - 18.11.2002
Gaining Access. Establishing Credibility. Positioning Your Solution. Managing Your Value.

» Percentage of Completion (US GAAP) SBS - 17.07.2000
Percentage of Completion (US GAAP) is a methodology to measure the resource usage, budget, profit.

» Effective Presentation Skills SBS - 18.10.1999
Presentation Goal. Effective Presentation Skills. Visuals. Voice Tone. Maintaining interest.

» Risk Management SBS - 14.06.1999
Risk Analysis. Risk Mitigation. Risk Assessment. Managing risks that take place.

» Proposal Management SBS - 22.03.1999
Properties of the tender process. Selecting and managing business partners. Negotiation techniques.

» Project Management and Coaching SBS - 01.03.1999
Project Planning. Resource Planning. Contract Management. Personnel Coaching

» Proposal & Project Management Methodology SBS - 15.02.1999
CHESTRA is a proposal and project management methodology. Cost, risk, partner, contract management.

» Marketing of Security Products RACAL - 17.11.1997
System Security. Data Security. Encryption. Authentication. Backup. Network Security.

» Sales Strategies ECI - 09.06.1997
Customer Relations Management. Product Management. New Project Development. Sales Focus.

» Advanced Management Topics ECI - 02.06.1997
Finance. Budgeting. Accounting. Reporting. Resource Management. Delegation.

» WAN/LAN Technologies BIMEL - 09.12.1996
WAN/LAN Topologies & Technologies. Routers. Switches. Gateways. Domains. TCP/IP. OSI Layers.

» Effective Managers Workshop ROTA - 11.11.1996
Team Forming. Motivation. Creating Synergy. Personality Prototypes

HOBBIES

SQUASH, VIDEO RECORDING & EDITING