Improvisation Based on Relaxation of Imitation Parameters by a Robotic Acoustic Musical Device

Kubilay K. Aydın, Aydan M. Erkmen, and Ismet Erkmen

***Abstract*—A new approach to musical improvisation based on controlled relaxation of imitation parameters by a robotic acoustic musical device, is presented in this paper. The presented RObotic Musical Instrument "ROMI", is aimed at jointly playing two instruments that belong to two different classes of acoustic instruments and improvises while assisting others in an orchestral performance. ROMI's intelligent control architecture also has the ability to provide player identification and performance training. In this paper we introduce the robotic device ROMI together with its control architecture, the Musical State Representation "MSR" and focus on parameter estimation for imitation of duo players by ROMI. The MSR we have developed for controlling ROMI is a feature extractor for learning to imitate human players in a duet. The control architecture has an automatic parameter estimation process that is usually employed for optimization of the imitation stage. Improvisation can be achieved by programming this process to function at non-optimal values, thereby sounding notes that are definitely different than the music piece being imitated. If this programming is not done without constraints then the resultant sound will deviate too much form the original music piece being imitated and can not be classified as an improvisation. The constraints of this programming are also introduced in this paper.**

***Index Terms*—Control, Imitation, Improvisation, Musical Representation, State Representation**

I. Introduction

The objective in our work is to develop a robotic acoustic musical device that will be jointly playing at least two acoustic instruments while assisting others in an orchestral performance and that will also learn to imitate other players using the same instruments. This intelligence in imitation also provides player identification and performance training. In this study we introduce the robotic device together with its control architecture and focus on our musical state representation and automatic parameter estimation. We also provide the initial steps of ROMI's improvisation realized by relaxation of imitation parameters.

Building an all new acoustic musical instrument which plays itself by learning from human players and being capable of improvisation is 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.

Initial results of our work on improvisation based on imitation of human players by a robotic acoustic musical device has been presented in [1]. The MSR that is used in controlling the imitation process has been presented in [2]. This paper gives the final results and sensitivity analysis of our work on improvisation by relaxation of imitation parameters.

Existing work on improvisation includes grammars, genetic algorithms and neural networks. Grammars have been developed for automated jazz improvisation in which non terminals can have a counter associated with them to indicate when to stop expanding. The key idea in this system is manifested in the terminals. The 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 [3], [4].

Genetic algorithms, inspired by the principles of natural selection, have been used as heuristics in optimization of imitation parameters for possible improvisations [5]. 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, that are improvisations meeting some convergence criteria [6]. Improvisation which is a difficult topic in robotic imitation has been investigated in the well defined musical improvisation domain of jazz [7]-[9]. Many of the studies on imitation and improvisation of musical instrument playing has been facilitated through the use of computer generated music [10] and various new electric musical instruments, including a wearable one a PDA based one and a graphical one have been proposed [11]-[13]. Research for imitation of playing musical instruments and reproduction of acoustic music has also been investigated as a pattern recognition problem [14]. The data used from imitation of an acoustic musical instrument playing technique has been applied to musical instrument teaching practice as a new area of application [15]. Artificial Neural Networks are systems inspired by neuron connections in the brain. CONCERT [16] is an Artificial Neural Network trained to generate melodies in the style of

Manuscript received January 31, 2012; revised February 14, 2012.

Kubilay K. Aydın is with Advanced Video Technologies Ltd, Ankara, TURKEY (phone: +90-532-332-3556; e-mail: e092170@ metu.edu.tr).

Aydan M. Erkmen is with the Electrical and Electronics Engineering Department, Middle East Technical University, Ankara, TURKEY (e-mail: aydan@ metu.edu.tr).

Ismet Erkmen is with the Electrical and Electronics Engineering Department, Middle East Technical University, Ankara, TURKEY (e-mail: erkmen@ metu.edu.tr).

Bach. CONCERT looks to learn information about note-to-note transitions as well as higher-level structure of songs.

More complex models, such as 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 [17], [18].

Some research works have focused on the presence of a single model which is always detectable in the scene and which is always performing the task that the observer is programmed to learn, [19], [20]. A fixed-function mapping based imitation supporting system has also been proposed [21].

The idea of a rule based algorithm is to model 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 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 [22].

Gesture based musical accompaniment systems have been developed [23]. Some have simplified robotic imitation by using only simple perceptions which are matched to relevant aspects of the task [24]. Some have simplified the problem of action selection by having limited observable behaviors and limited responses [25], by assuming that it is always an appropriate time and place to imitate [26], and by fixing the mapping between observed behaviors and response actions [27]. A survey of AI methods in composing music is given in [28] and melody extraction as the basis of improvisation by AI methods has been investigated in [29].

Our approach to improvisation is based on an imitation process. The control architecture of ROMI has been designed and the internal state representation MSR has been formulated from the start with improvisation in mind. The result is a system which produces additional notes, including silences, and deleting notes form the original music piece being imitated. Since the underlying process that is doing these changes is a parameter estimator for the imitation, the resultant samples are coherent with the existing melody, rhythm, and note range of the original musical piece. The system patches the improvisation parts into the imitation parts which further enhances the musical quality of the improvisation for the listener. We also introduce constraints on how this relaxation of imitation parameters must be governed in order to keep the deviations under control.

In our work, we concentrate upon imitating by ROMI to reproduce acoustic melodies from human teachers playing two types of instruments. In the second section, ROMI will be introduced together with its control architecture. In the third section, we will summarize our musical state representation that is used for controlling ROMI. The imitation process will be demonstrated by an example in the same section as well. Our parameter estimation process will be presented and discussed in the fourth section. Section five demonstrates the results of our proposed improvisation approach based on relaxation of imitation parameters. Section six concludes the paper with sensitivity analysis.

II. Design of ROMI

Two acoustic musical instruments from two different domains have been selected, namely Clavichord and Tubular bells in building ROMI where its main components of the tubular bells and playing subsystem are shown in Figure 1. ROMI utilizes a 2 octave string section with "note A" frequencies of 110 and 220 Hz and a tubular bells section having a 1 octave percussion with "note A" frequency of 55 Hz. The sound of the tubular bells section is chromatic only in room temperatures since the sound production properties of the copper tubes being used are sensitive to temperature changes. Sound is generated by solenoids hitting the copper tubes and the harp strings as shown in Figure 1. The string sections' loudness is low compared to the tubes, so we utilize an amplifier for the string sections sound.

Sample sound recordings have been collected from two musicians playing a piano. These recordings are then utilized for the development of the imitation algorithms after they are converted to MIDI format by a commercial software. We developed a software converter which converts these MIDI representations, which are incomprehensible to the human eye, into recognizable note sequences by ROMI enabling it to gain insight to the musical notes being played.

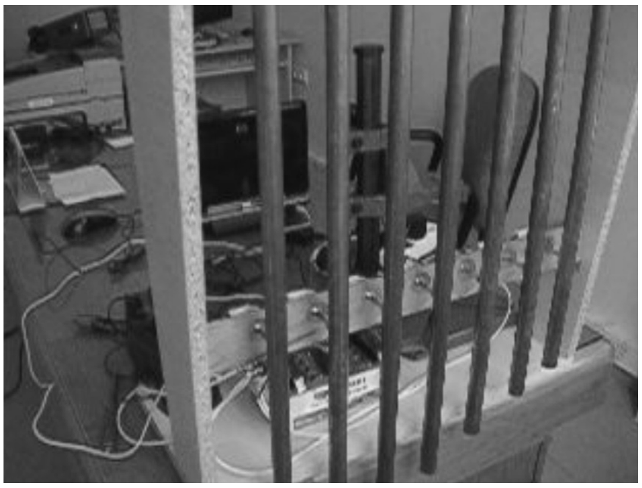


Fig. 1 Tubular Bells, and Solenoids of ROMI

A relay control card has been designed and used as shown in Figure 2. This card is connected to a PC via the USB port and can be programmed to control the 220VAC 7A relays. These relays control the current of the solenoids in an ON/OFF configuration. The block diagram of this card is given in Figure 3. The resultant control can simulate note ON/OFF commands of a sequencer. Velocity control for ROMI is implemented by pulse width adjustment. Velocity control is incorporated in the software control architecture of ROMI enabling future D/A implementations.

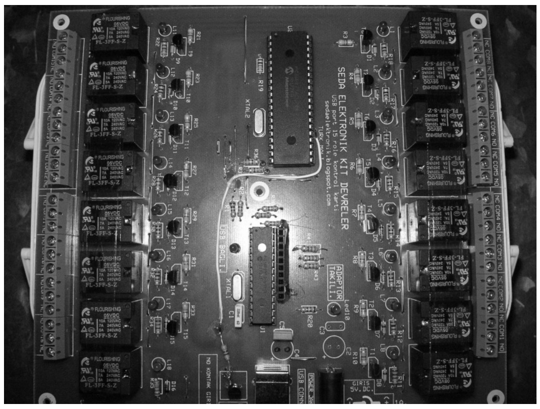


Fig. 2 Solenoid Control Card

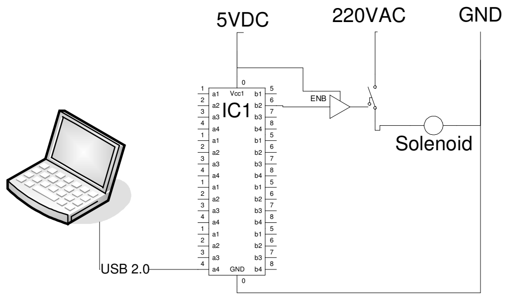


Fig. 3 Block Diagram of Solenoid Control Card

Figure 4, shows how velocity control is implemented by pulse width adjustment. 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_1 supplied to the solenoid which is smaller than t_{hmax}, will result in a sound in lower loudness. If a pulse supplied to the solenoid has a wider width $(t+1)_1$ than t_1 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 tubular bell while it is at rest. The width of the pulse can be used with limited precision to implement velocity control of a note. Any pulse width larger than t_{hmax} has the potential of causing an unwanted secondary sound on the tubular bell, therefore t_{h100} must be smaller than $t_1 + t_2$ which is the minimum time duration between two consecutive notes played on the same tubular bell.

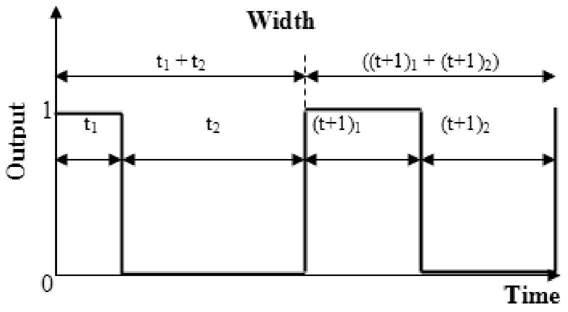
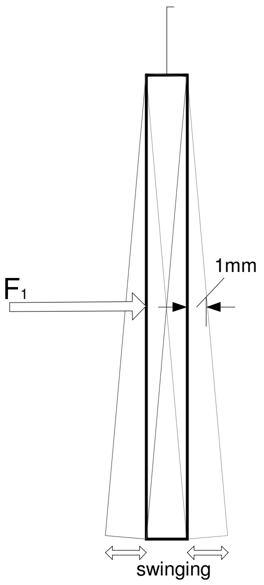


Fig. 4 Pulse Width Adjustment for Velocity Control

The string section is currently using simple hammers to hit the strings. Small microphones are employed to amplify the sound. Trials showed that bandbass filters with very sharp and narrow frequency responses are necessary to avoid crosstalk between these microphones.

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. Our test showed that the tubular bells section is chromatic for a temperature range of 22-26C. There is no way to tune the cooper tubes for differences in ambient temperature since their frequency response is a function of their geometry.



There is a noise due to the operation of the relays and solenoids which is inaudible when the control card with the relays are placed in a sound proof box.

The operation of tubular bells present a swinging problem as shown in the figure to the left. Once a solenoid hits a tubular bell, a momentum is induced to the tubular bell which is proportional with the bells weight. This swinging motion is like a pendulum if the solenoid is very well aligned with the center of the tubular bell. If not, than 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. After a tubular bell is hit by the solenoid the swinging motion fades away in a finite time. If a second note on the same tubular bell is to be played before the pendulum motion has become negligible, than the solenoid to hit the tubular bell at a location other than the standard location 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 can become unpredictable. Our solution to reduce the swinging problem to a negligible level, is to hit each tubular as close to its hinge as possible.

The control architecture of ROMI is given in Figure 5. Here two sets of musical signals are separately processed, one for the clavichord and the other for tubular bells. The processing of these signals is never mixed in any of the application blocks. In learning mode, the human teachers play the respective musical instrument in an acoustically noise free environment. 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.

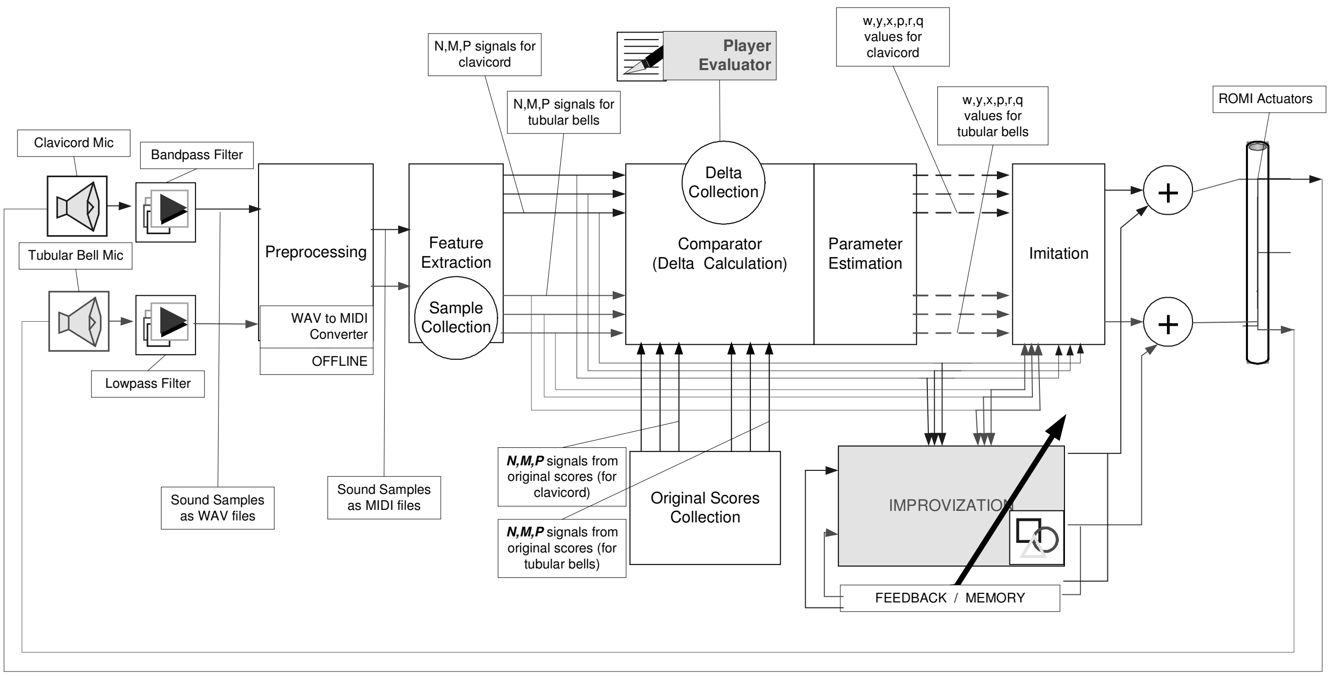

Clavicord Mic
Bandpass Filter
Tubular Bell Mic
Lowpass Filter
Preprocessing
WAV to MIDI Converter
OFFLINE
Feature Extraction
Sample Collection
N,M,P signals for clavicord
N,M,P signals for tubular bells
Player Evaluator
Delta Collection
Comparator (Delta Calculation)
Parameter Estimation
w,y,x,p,r,q values for clavicord
w,y,x,p,r,q values for tubular bells
Imitation
ROMI Actuators
Sound Samples as WAV files
Sound Samples as MIDI files
N,M,P signals from original scores (for clavicord)
N,M,P signals from original scores (for tubular bells)
Original Scores Collection
IMPROVIZATION
FEEDBACK / MEMORY

Fig. 5 Control Arcithecture of ROMI

This process is shown in Figure 5, as "WAV to MIDI Conversion". 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 N, M and P. Original musical recordings representing models to be used for player identification and parameter estimation, are also converted into MSR and is stored in the "Original Scores Collection". 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 the reproduced sound is fedback to the system via microphones and control is achieved by minimizing the difference between the musical information stored in the MSR with the music generated.

III. Musical State Representation

In the "Musical State Representation" (MSR) that we have developed as a feature extractor for controlling ROMI, time (t) is slotted into 1/64th note duration. 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 (these numbers are based on the fact that the control algorithms are set for 120bpm).

At the moment our MSR 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" (MPg) are stored at the "Feature Extraction" processes' "Sample Collection" as shown in Figure 5. Our reason to chose 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.

Each monophonic voice is represented by two number streams "N" and "M", where the number values are whole 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.

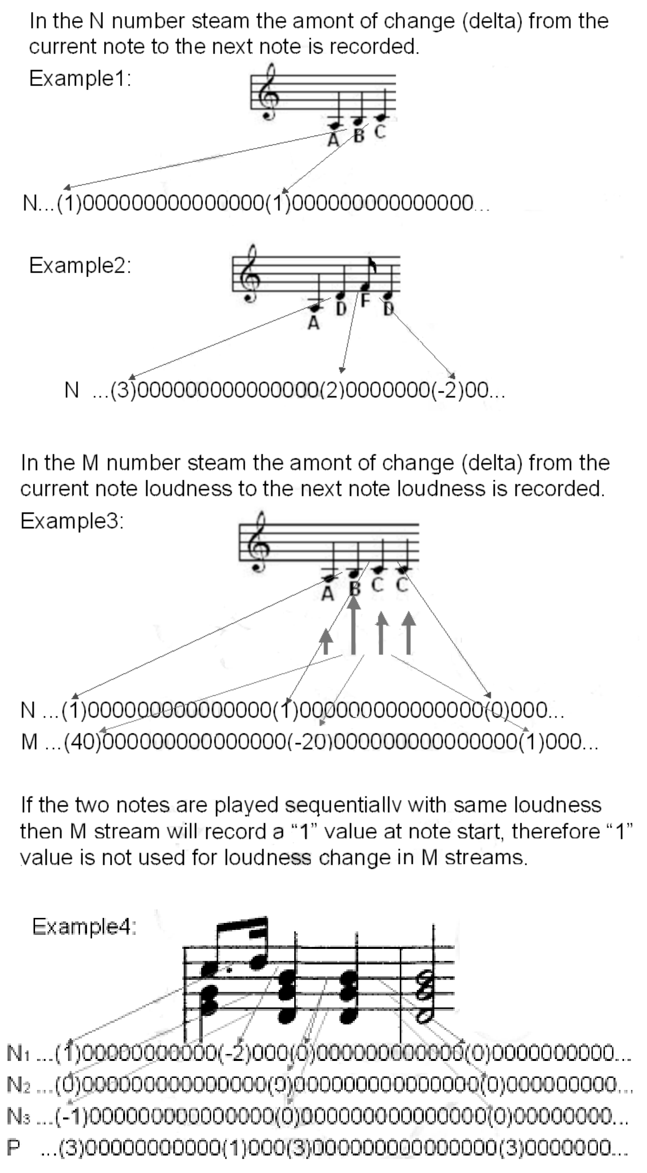
The number streams N and M consists of "0" values as long as there is no change in the current note. Each number in these streams are equivalent to a 1/64th note duration. Note that for most people a 1/64th note is incomprehensibly short.

Number stream "P" is an event indicator similar to a token state change in a Petri Net, where P values can assume any rational number. The event indicator P number stream is important in our improvisation algorithms. The addition of the event indicator P to the MSR has eased the detection of tempo in musical parts.

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.

Four examples will demonstrate the process of generating the number streams N, M, and P next:


In the N number steam the amont of change (delta) from the current note to the next note is recorded.

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

In the M number steam the amont of change (delta) from the current note loudness to the next note loudness is recorded.

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

If the two notes are played sequentiallv with same loudness then M stream will record a “1” value at note start, therefore “1” value is not used for loudness change in M streams.

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

Starting note value and velocity (loudness) 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.

The following figures present a visualization of our MSR. Here the opening part of Lugwig von Beethoven`s Ecossais has been used as the sample. Figure 6, 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 array has been continued from one line below for each 64 consecutive array elements. The numbers in the first row and column represent 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 Figure 6 & 7, 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. Note that the changes in M streams has a larger scale. Pure black array elements represent a “silence” in M streams.

Figure 7, shows the MSR for one of the performances of the same musical part that ROMI “heard” 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 “heard” recording from human teacher has small deviations from the original score.

Figure 8, shows the difference (Delta Vector) between the original score and the heard sample 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 “heard”. This information can be used for parameter estimation and player evaluation as will be presented in the next section.

Using the MSR made of N, M and P streams, ROMI imitates a musical piece based on the following algorithm. This algorithm uses six imitation parameters named as “w, y, x, p, r, q” which affect the reproduction quality of the imitated musical part.

1. Play all notes where Nij(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 Mij(t) values.

2. Play all notes, not already played by step 1, where Mij(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 Nij(t) values.

3. Play all notes, not already played by step 1 or 2, where Pj(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 Nij(t) values and with average value of all available Mij(t) values.

4. If Pj(t) lengths are different, select longest available length as music piece length with gradually decreased loudness, starting the decrease with the shortest available length.

The imitation parameters and their effect to imitation performance are explained next:

w: This parameter is the main note generator. It uses the note change information, which is stored in the N streams. When a sufficient number, “w” percent of all 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. The effects of this parameter on imitation performance is discussed in the next section.

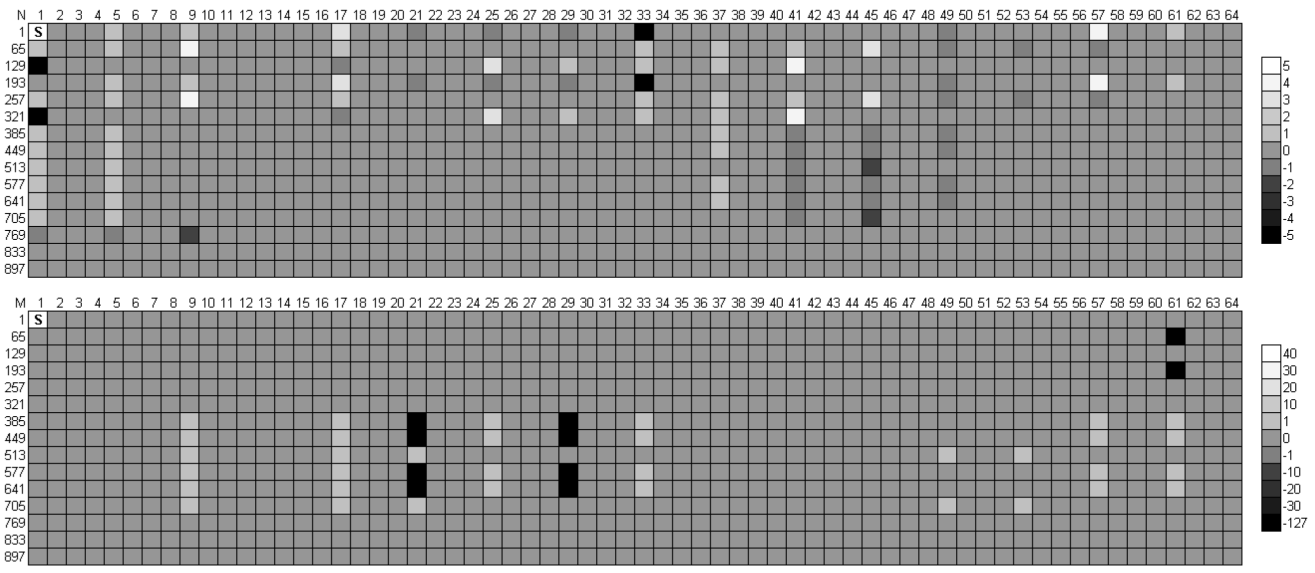


Fig. 6 N & M Number Streams for Original Score in MSR

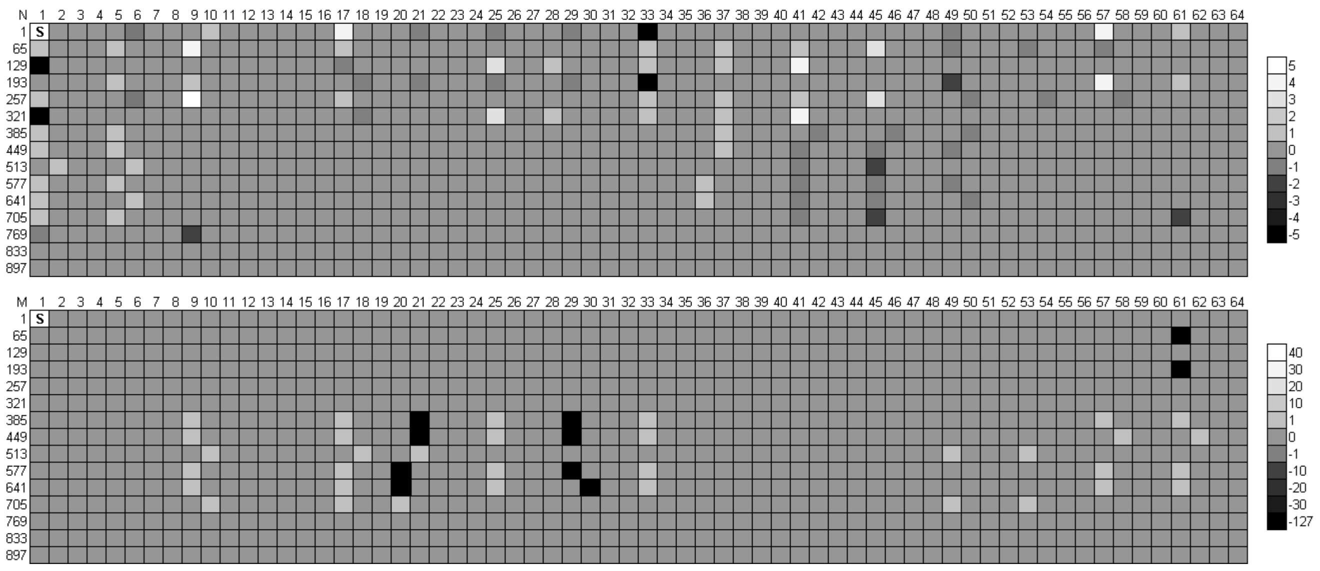


Fig. 7 N & M Number Streams for Played Sample in MSR

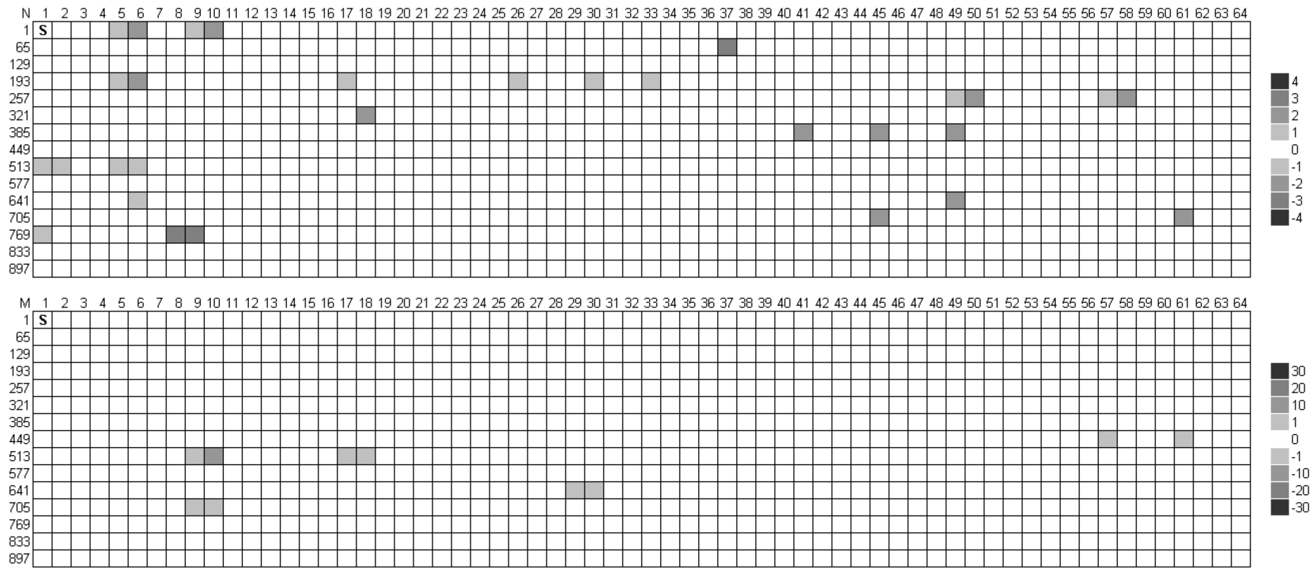


Fig. 8 Delta in N & M Number Streams Between the Original Score and the Played Sample in MSR

y: This parameter is the secondary note generator. It uses loudness change information, which is stored in the M streams in the same mechanism as explained for “w”. A new time window parameter “r” had been defined other than “p” in order to gain more control on imitation performance.

x: This parameter is used for generating notes that are in the original score but were not produced by the note generators explained above. This 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” had been defined other than “p” and “r” in order to gain more control on imitation performance.

p: Due to slight tempo variations or less than perfect teacher performances, some notes are sounded about 1/64th of a note before or after they are in the original score. This parameter controls the width of a time window to group such note values together. 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 is chosen.

r: Same as “p” but used for loudness variations. Is effective on M streams.

q: Same as “p” but used for event changes.

IV. Parameter Estimation

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 P 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, it is possible to track the performances of each human musician; if not, this process becomes that of player 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, is assumed to have the highest “quality” if imitation mode is used 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 imitation process uses the six imitation 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 imitation algorithm given in the previous section. 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.

At the end of studies, the parameter estimation step showed us that there is no unique value set for minimizing the delta for these parameters but a range of parameter values has to be generated for very close Delta values. Our studies also showed that choices for p, q, r parameters are limited, since their value is in fact connected to the time granularity, or resolution, of the MSR. The w, y, x parameters can attain larger ranges.

Due to the structure of the imitation process these parameters are not independent. The choice for one effects the plausible values for the others. For the parameter estimation process, samples for piano part Ecossais from Ludwig von Beethoven has been recorded by ROMI from different human teachers. This has been separately done for the tubular bells and clavicord sections.

The effects of different values for the imitation parameters are shown in the following figures. Each graph in these figures have been generated using the imitated piano parts musical reproduction, being compared with the original score. Some imitation parameters are set to fixed values to show the effects of changing others. Delta$_k$ values have been used for one of the human teachers, total number of samples processed is six.

Figures 9 and 10, show how Delta$_k$ values are effected by changes in the main note generator parameters “y” and “w”. Parameters “y” and “w” 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. If either one of these parameters is 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.

Figure 11 shows the effects of parameter “x” on imitation performance. This parameter has less impact on imitation performance compared to “w” and “y” parameters. This is understandable since the imitation algorithm generates notes based on N, M and P streams in this order. This results in most of the notes already being produced by the N and M streams with P 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. For values above 95 it generates less notes than the original score. Figure 12 shows the effects of parameter “p”. Graphs for parameter “r” and “q” have the same shape and effect the imitation performance in a similar way as explained here for parameter “p”. The parameter “p” used by the first note generator using N streams and has the greatest impact on note production.

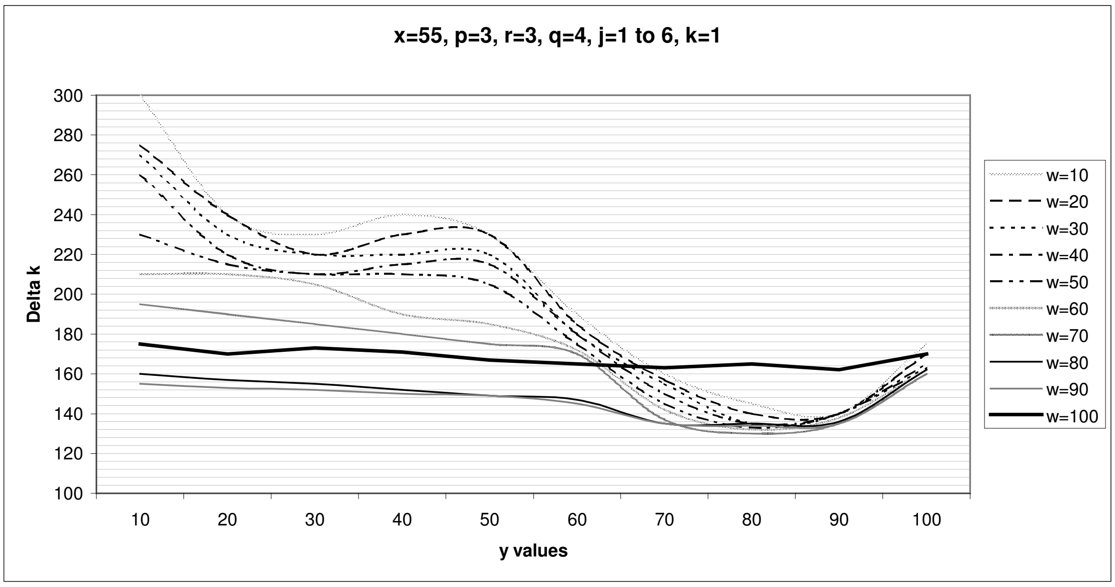


Fig. 9 Delta$_k$ for Varying y Values with 10 Different w Values

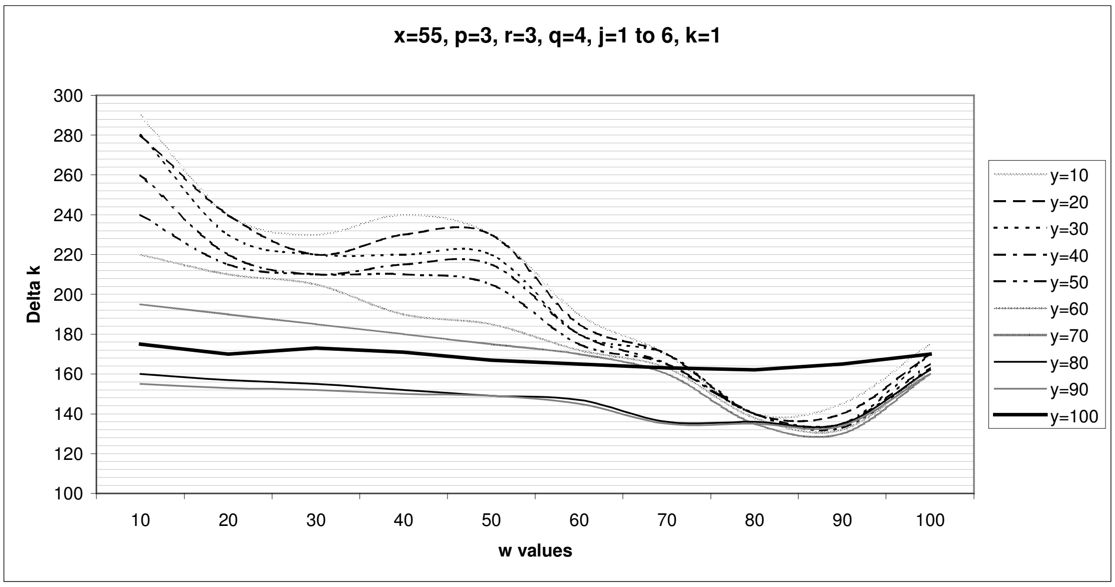


Fig. 10 Delta$_k$ for Varying w Values with 10 Different y Values

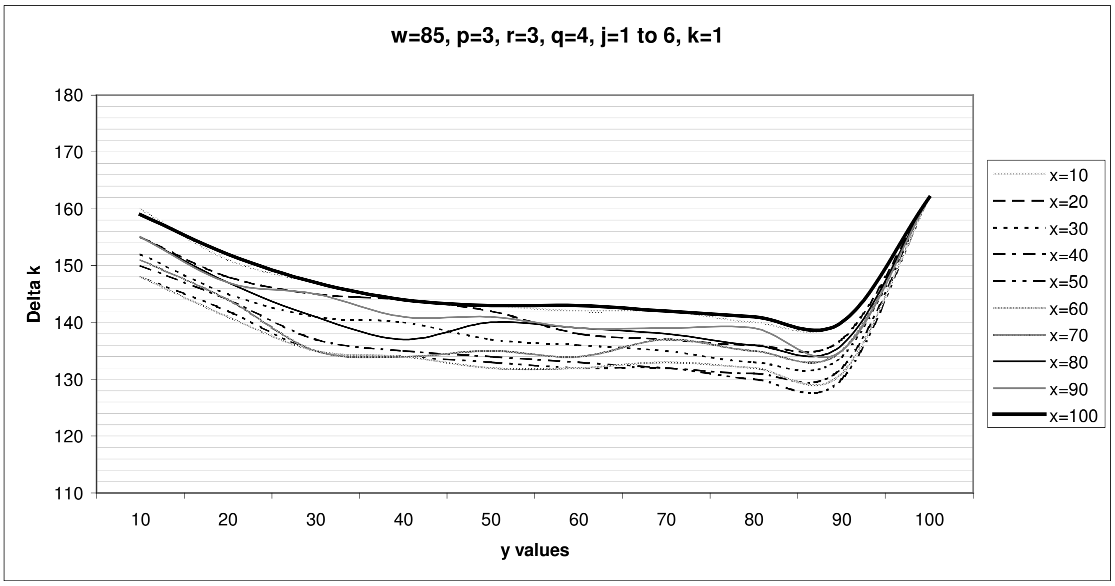


Fig. 11 Delta$_k$ for Varying y Values with 10 Different x Values

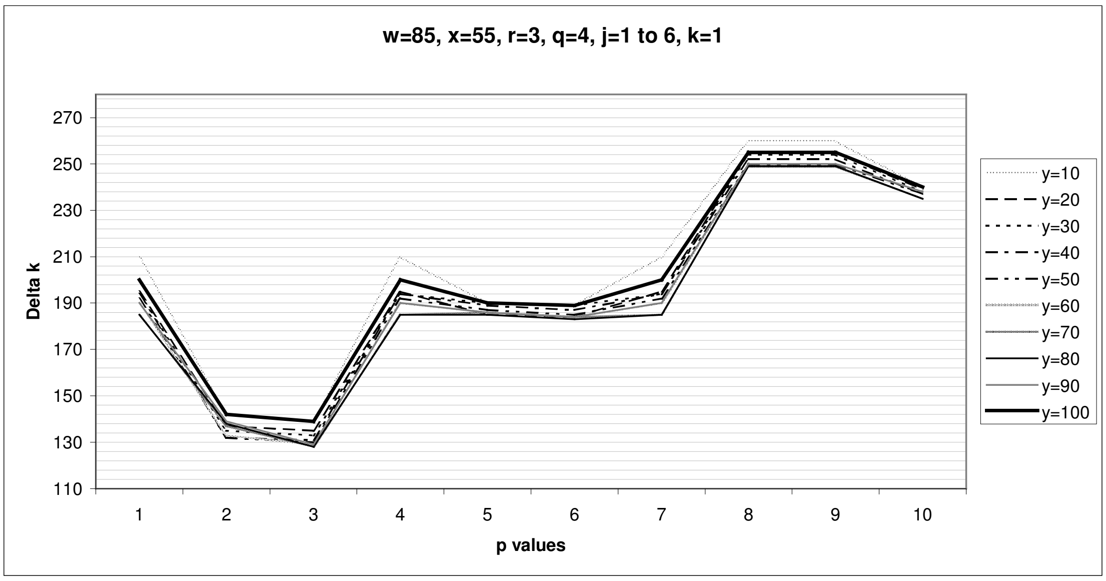


Fig. 12 Delta$_k$ for Varying p Values with 10 Different y Values

A value of 1 will produce less notes than the original score. Values of 2 and 3 are optimal. Values above 3 produce more notes than the original score by combining two consecutive notes into one, increasing Delta$_k$. The second jump in Delta$_k$ at "p" value of 8 is due to the fact that more notes that is 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 of 12 and 16 for this parameter.

V. Improvisation by Relaxation of Imitation Parameters

In our studies we have seen that it is possible to use Improvisation by Relaxation of Imitation Parameters (IRIP) as a low level improvisation tool; whose parameters are defined within time intervals controlled by a higher level improvisation algorithm. The final results are presented in this section. The values of imitation parameters that minimize Delta$_k$ produce an output very similar to the original score, or the median of the samples. Improvisation can be achieve with limited success, by relaxation of the imitation parameters that result in non-minimum Delta$_k$. Most of the imitation parameters give higher Delta$_k$ if used below or above certain values.

Our studies showed that the values that produce less notes than the original score are less suitable for improvisation. The following example illustrates this. Figure 13 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 at their near optimal imitation values of x=55, p=3, r=3, q=4. As seen from Figure 13, there are less notes than the original score given in Figure 6. The Delta$_k$ is however higher than that of the sample given in Figure 7. This can be seen if the Delta Vector for Improvisation Sample 1, given in Figure 14, is compared with the lower value Delta$_k$ sample given in Figure 8.

Please note that Improvisation Sample 1, is one of the random samples available. There are many distinct output samples resulting if imitation parameters are relaxed. Some produce even higher Delta$_k$ values. The sample given in this example is one with an average Delta$_k$ for the imitation parameter set given. On its own it can not be classified as an improvisation but as a bad imitation sample.

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 non-minimum Delta$_k$ with values that produce more notes than the original score are more suitable. The following example illustrates this. Figure 15, 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 at their near optimal imitation values of x=55, p=3, r=3, q=4. As seen from Figure 15, there are more notes than the original score given in Figure 6. The Delta$_k$ is similar to the sample given in Figure 7. This can be seen if the Delta Vector for Improvisation Sample 2, given in Figure 16, is compared with the lower value Delta$_k$ sample given in Figure 8. Again, Improvisation Sample 2 is one of the random samples available.

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.

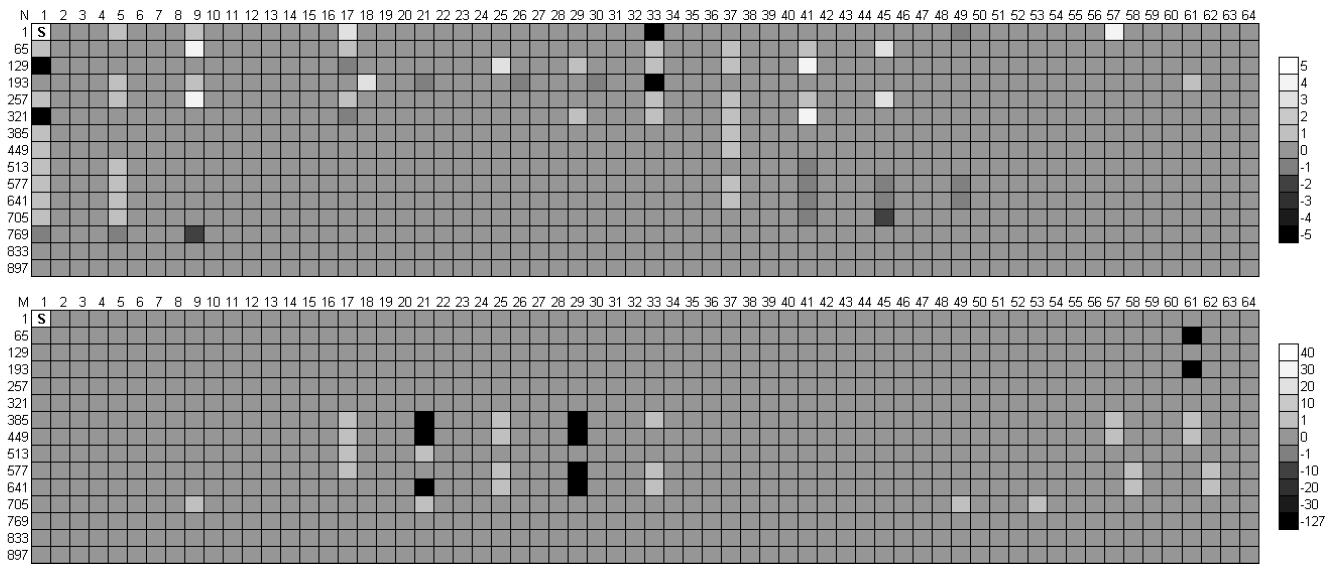


Fig. 13 N & M Number Streams for Improvisation Sample 1

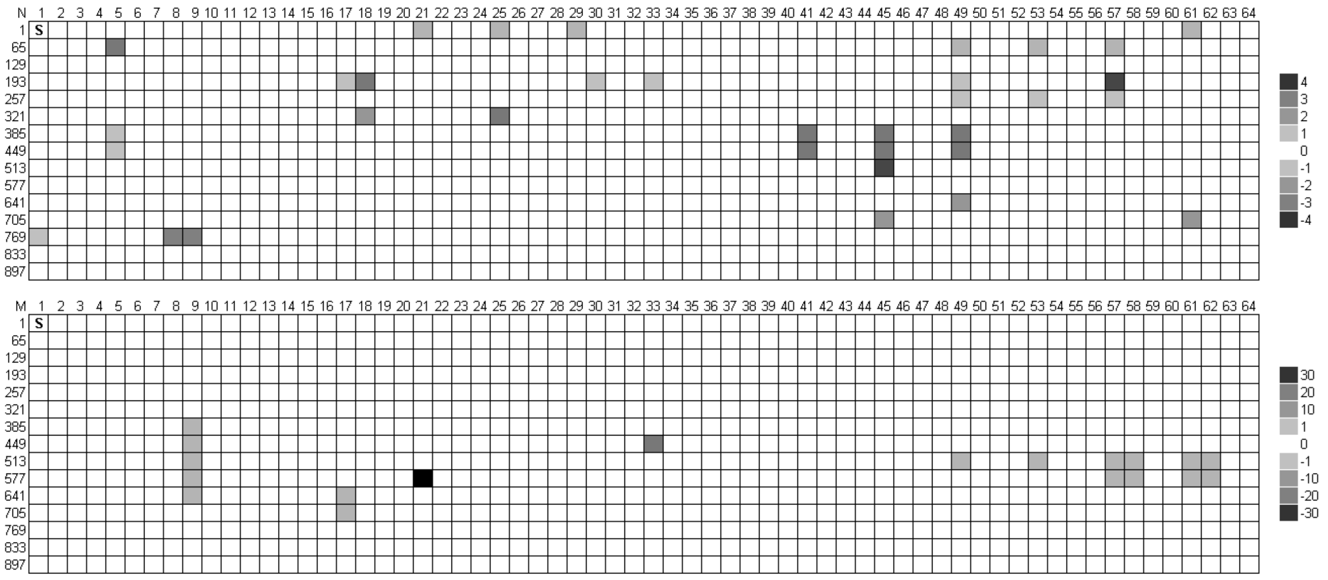


Fig. 14 Delta in N & M Number Streams Between the Original Score and Improvisation Sample 1

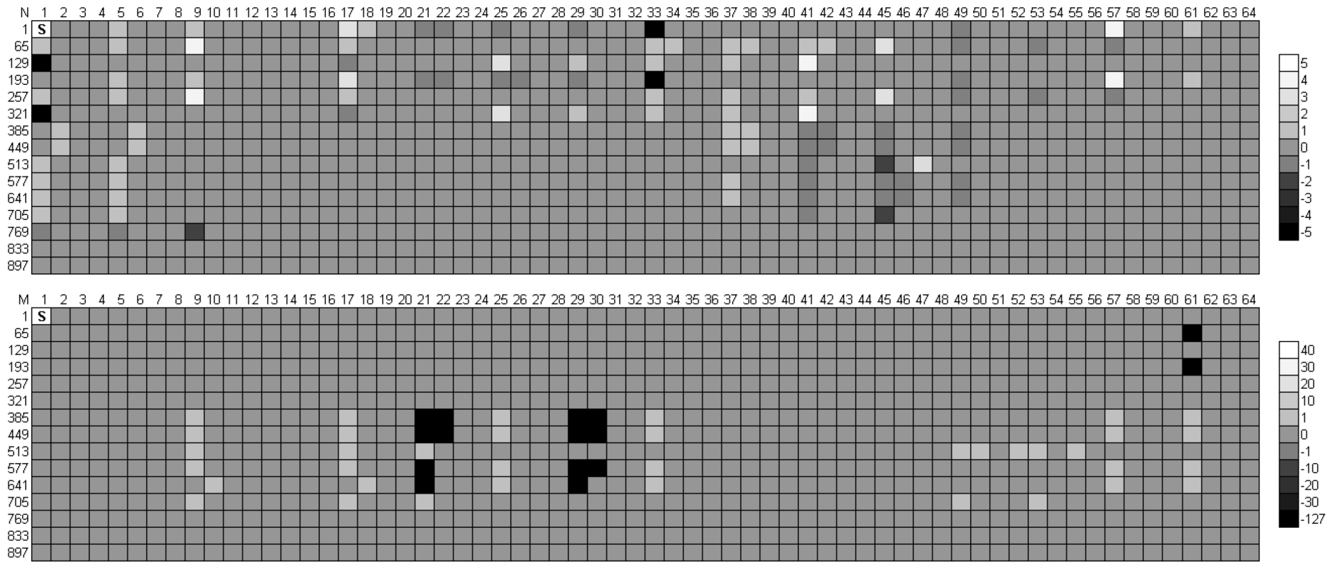


Fig. 15 N & M Number Streams for Improvisation Sample 2

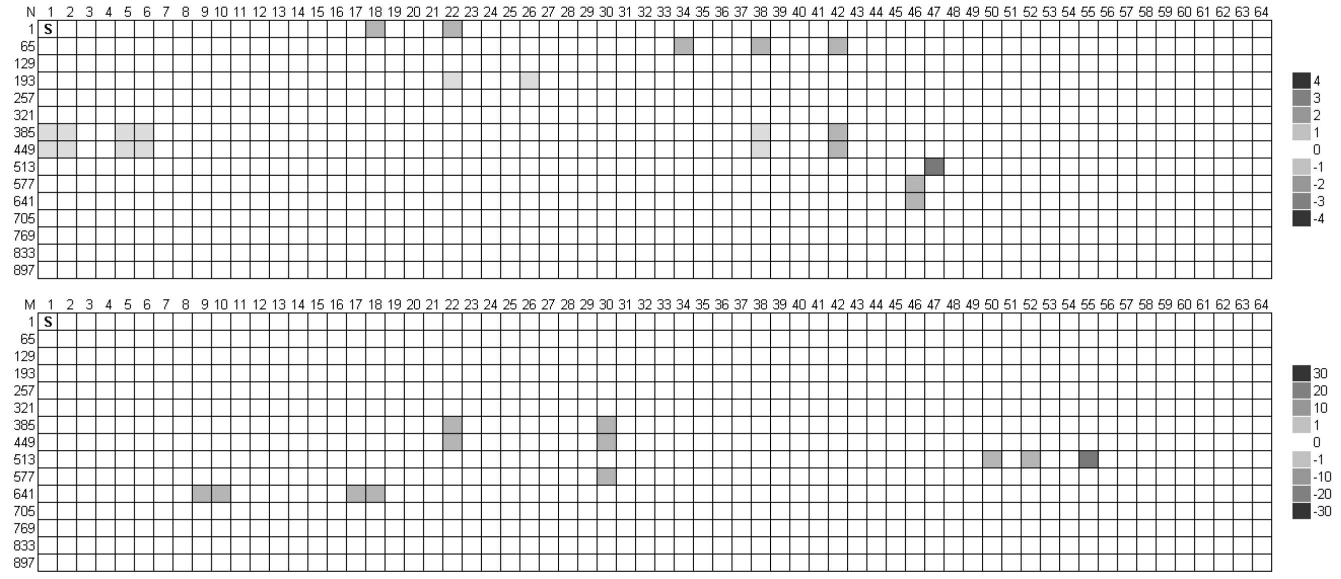


Fig. 16 Delta in N & M Number Streams Between the Original Score and Improvisation Sample 2

Our studies have showed that if imitation parameters are further relaxed the system tends to drift too much out of the original scores' note frequency range. The resultant improvisation is more exiting due to higher variations but the overall musical part becomes fuzzy. We propose to use a partial application for the improvisation if the imitation parameters are further relaxed. For example, a mask as shown in Figure 17, 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. For example the "black" slots set imitation parameters at y=50, w=50, x=50, p=2, r=2, q=3; and imitation parameters for the "white" slots set at y=85, w=85, x=55, p=3, r=3, q=4.

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 is; a higher level of improvisation algorithm. There is a rule set that we have formulated based on our studies. These rules are:

1. Short periods of IRIP does sound like a wrong note has been played. Therefore we suggest that the minimum duration for an IRIP part must be at least 1 seconds. This value depends on the bpm of the musical piece.
2. Long periods of IRIP tend to drift out of the scale of the musical piece being played 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 drift the note sequences out of the scale of the musical piece. To limit this effect, these intervals should not be larger than 2 seconds and at certain intervals the musical piece could be returned to one absolute note value.
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 at what the modal of the musical piece is.

Figure 18, helps to visualize these ground rules. In this mask 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.

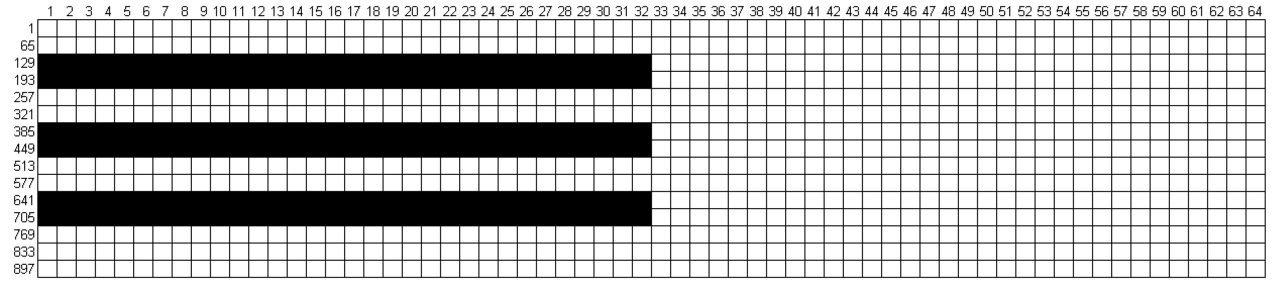


Fig. 17 Mask for Improvisation Intervals, Black Slots Represent Where Imitation Parameters are Very Relaxed

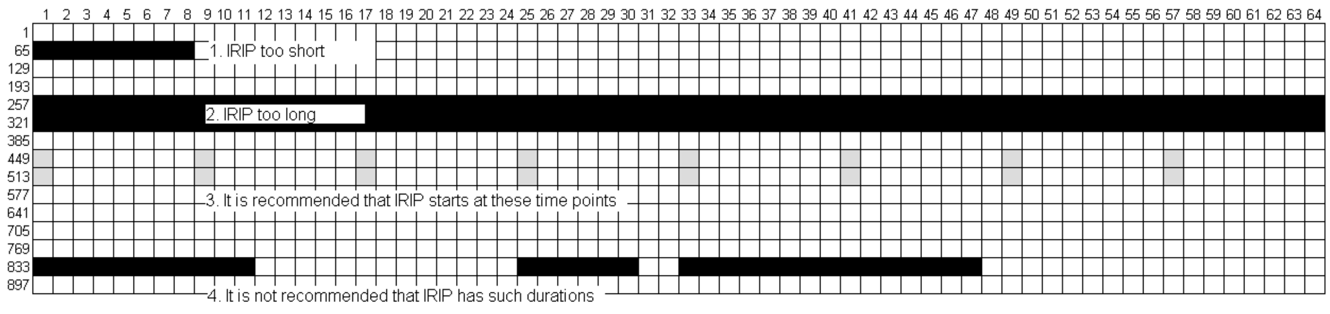

1. IRIP too short
2. IRIP too long
3. It is recommended that IRIP starts at these time points
4. It is not recommended that IRIP has such durations

Fig. 18 Visualization of ground rules for successful implementation of IRIP

VI. CONCLUSION

We experimented with our MSR to increase its performance in fast notes. The idea was to increase the time granularity of the system by defining a smaller time window; for example each time slot is a 1/128th note. However, if granularity of the discrete time model is increased, in order to pinpoint notes, our system shows a strong tendency to produce extra notes that were not intended. To limit this tendency p, r, q values can be increased. However this essentially reduces the system to one with lower granularity, not resulting in any gain in fast note performance.

Our system uses the note and loudness change information rather than the absolute values of notes. This enables our system to trace the melody and rhythm in a musical part implicitly. If the imitation algorithm is changed to evaluating the loudness changes before the note changes, then the system becomes more sensitive to filter performance. If such a change is made, then the bandpass and lowpass filters in Figure 5, used for acoustic signal conditioning, must have gain controls. If the original loudness for different samples are not very close without any adjustment, then our system tends to create additional notes in the opening parts of the imitation that are not present in the original score.

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.

Since improvisation is a subjective topic, a tool for evaluating results is necessary. We followed a similar approach that other studies have followed so far; gathering a listener group and asking them to rate 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 form 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 improvisations.

Applying the IRIP rules with a higher level of AI is the next step in our studies. We have two areas of investigation. One is to develop an improvisation algorithm based on n-grams including velocity information. The second one is to develop a patching algorithm which will analyze the current imitation and the generated improvisation and decide where to patch the improvisation. In this way our work will have a new approach to improvisation. Our current idea of how this patching could be implemented is to analyze the imitation and improvisation as a signal and match the slopes of imitation and improvisation signals at the entry and exit points of the improvisation.

APPENDIX

Relevant Musical Information

After many trial and errors we arrived at the conclusion that a musical state representation where only note and loudness delta values are stored in a discrete time model will suit our improvisation needs best. With the addition of the starting note and loudness values, the absolute note and loudness values can also be obtained from the delta values, but our model does not make use of the absolute values. If the goal was to playback music, then perhaps the absolute value stream would have been a better candidate.

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. In fact a musical attribute can make this more clear: an average listener 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 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 the 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 will typically be 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 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 has emotional results on human listeners.

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

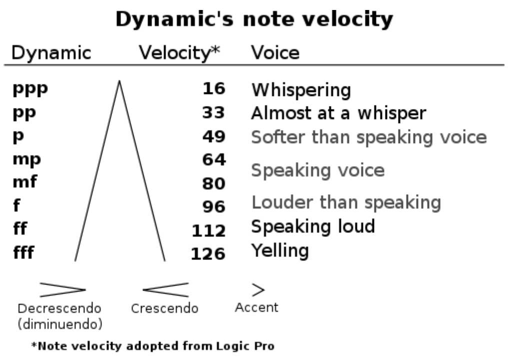
Another example of the lesser importance of note values is as follows: two notes with fundamental frequencies in a ratio of any power of two (e.g. half, twice, or four times) are perceived as very similar. Because of that, all notes with these kinds of relations can be grouped under the same pitch class. 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).

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".
mp, standing for mezzo-piano, meaning "moderately soft"
m*f*, standing for mezzo-forte, meaning "moderately loud".

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



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

REFERENCES

[1] 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.

[2] 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.

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

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

[5] 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.

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

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

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

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

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

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

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

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

[14] 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.

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

[16] 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.

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

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

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

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

[28] 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.

[29] 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.