Interactive Musical Partner: A Modular Human/Computer Duo Improvisation System

Jeffrey Albert

Loyola University New Orleans,
College of Music and Fine Arts
6363 St. Charles Ave., New Orleans, LA 70118 USA
jvalbert@loyno.edu
http://www.loyno.edu/~jvalbert

Abstract. The Interactive Musical Partner (IMP) is software designed for use in duo improvisations, with one human improviser and one instance of IMP, focusing on a freely improvised duo aesthetic. IMP has Musical Personality Settings (MPS) that can be set prior to performance, and these MPS guide the way IMP responds to musical input. The MPS also govern the probability of particular outcomes from IMPs creative algorithms. The IMP uses audio data feature extraction methods to listen to the human partner, and react to, or ignore, the human's musical input, based on the current MPS. This article presents the basic structure of the components of IMP: synthesis module, musical personality settings, listener system, creative algorithm, and machine learning implementation for timbral control.

Keywords: Improvisation, Interaction, Computer Generated Music

1 Introduction

The Interactive Musical Partner (IMP) is software designed for use in duo improvisations, with one human improviser and one instance of IMP, focusing on a freely improvised duo aesthetic. My goal in the creation of the IMP was to make a software system that could live up to all three parts of its name. It would have to be interactive, meaning that there is a two-way flow of information between the human performer and IMP. It would have to be musical, meaning that the results are musically rewarding both the performer and to listeners. And finally, it should be a partner, meaning that it is equal parts leader and follower, not always simply accompanying the improvising human, and at the same time, not always requiring the human to musically accommodate its output.

Two important concept/design decisions were made very early in the process. 1) IMP would be a monophonic participant in its musical settings, and that those settings would fall into the sphere of non-idiomatic improvisation (or free improvisation). 2) IMP would deal with the ways it hears, remembers, and creates musical content with the fewest possible levels of abstraction. Whenever possible pitches are dealt with in terms of frequency and durations in terms of milliseconds. By avoiding thinking in terms of note names and note vales, IMP can more easily navigate the spaces outside of tonality and tempo.[1]

IMP was programmed in the Max 6 Programming environment, using externals by Tristan Jehan[5] and Tap.Tools[6], and uses The Wekinator[4] for machine learning implementation. IMP consists of: a synthesis module, a Musical Personality Settings (MPS) module, a frequency decider, a duration decider, a global variation module, a listener module, and a number of smaller decider modules. I will use the term creative algorithm to refer to the aspects of the duration decider and frequency decider that control IMP's autonomous output.

2 Synthesis Module

The synthesis module is IMP's voice. This is the section of the software that makes the sounds, and manages the messages sent from the other modules. The frequency decider, duration decider, global variation module, and listener module are all sub-patches of the synthesis module.

The synthesis module uses frequency modulation (FM) synthesis to generate its sounds.[3] IMP also employs a second order FM synthesis, meaning that there is a second modulating oscillator, which modulates the product of the first order oscillators. Variations of the combination of these two FM pairs are how IMP controls timbral variety. The primary FM pair is usually set to a simple harmonicity ratio; 2 is the default. A more complex ratio on the second FM pair allows for IMP to make its sound more complex and strident by adjusting the amount of modulation from the second modulating oscillator. The modulation depth of this second order pair is controlled by a gain, which is controlled by the timbral noise analysis module. This means that at times the second order modulator is completely muted and does not affect the sound. This mechanism will be discussed in greater detail in the section on Machine Learning and Timbral Interaction.

The product of the FM module is multiplied by a line function (a `line~` object) that creates an amplitude envelope for each event.[2] Once this envelope reaches zero, a bang[3] is sent that triggers the next event.

The synthesis module also contains three smaller decision modules: the `density_decider`, the `volume_decider`, and the sub-patcher called `silent_event`.

[1] There are two exceptions to this principle in which musical data for the generative algorithm is stored in an abstracted form.

[2] I will use the term event to describe what is often called a note in musical discourse. This is in keeping with my desire to stay free from the note/rhythm abstractions, and it also allows for the existence of events that have no volume (rests) to be treated the same as events that do have volume, which is how they look to IMP.

[3] The bang message has a specific use within Max it's the message that tells many objects to *do that thing you do.*

The `density_decider` is the module that controls the density of events. Once the `line` $\sim$ object (amplitude envelope) completes its process, it sends out a bang that enters the `density_decider`. A decision is made as to whether the next event will make sound, or be silent. This decision is weighted by the values in the density parameter of the MPS. The densest setting will have every event make sound, and the least dense setting will have no events make sound. If the next event is to have sound, the `density_decider` sends a bang on the `next_now` channel, which cues the synthesis process, and if the event is to be silent, a bang is sent to the `silent_event` sub-patcher. The `silent_event` sub-patcher receives the next duration from the `duration_decider` and lets that amount of time pass in silence before sending a bang to the `density_decider` to cue the decision process for the next event. This system allows for the MPS to control the density of the texture without necessarily changing any of the other duration parameters, so it is possible for IMP to play sparsely in a setting that still has a relatively short duration of events, or densely in a setting that has relatively long event durations.

3 Musical Personality Settings

One of the original goals of this research was to design a system with variable sets of behavioral characteristics, or musical personalities. This is implemented in IMP through the Musical Personality Settings (MPS), which are seven separate parameters that influence various aspects of IMP's behavior. The parameters are: Density of Events, Length of Events, Rhythmic Regularity, Frequency Listenerness, Duration Listenerness, Melodicness, and Variation.

Each MPS parameter is controlled with a slider on the MPS interface. The interface also contains the mechanism for setting the length of an episode (or performance), and a visible timer to give the performer a reference for the amount of elapsed time since the beginning of the episode.

The Density of Events parameter controls the weighting of the `density _decider` 's decision algorithm, which decides whether an event will make sound or not. The higher this parameter is set the higher the sound to silence ratio will be. This parameter is also influenced by what is heard from the human, once an episode begins.

The Length of Events and Rhythmic Regularity parameters work together to control IMP's tempo and sense of pulse. I use these terms (tempo and sense of pulse) loosely in this context, since there is no abstraction of meter present, but there can be a sense of IMP playing faster or slower, and in more or less regular event lengths. The Rhythmic Regularity parameter controls a pool from which duration proportions are chosen in the creative algorithm, and the Length of Events parameter controls a factor that controls the speed at which these proportions are realized.

Listenerness is a term I have coined to describe the two parameters that control IMP's responsiveness to human input. The lower the listenerness value, the more independently IMP will behave, and the higher the value, the more IMP will derive its output from what it has heard from the human. There are two listenerness settings; one for frequency and one for duration. Frequency Listenerness controls the weighting of the frequency decider mechanisms and influences whether IMP's pitch output is derived from its creative algorithm or from the pool of pitches it remembers hearing from the human. Duration Listenerness controls the weighting of the duration decider and similarly influences IMP's output in terms of duration of events.

The Melodicness parameter sets a set of pitches from which the creative algorithm chooses when IMP is generating content on its own. As the value moves from low to high the pool of available pitches moves from pentatonic sets, through major scales, melodic minor (ascending) scales, diminished scales, whole tone scales, and finally to a fully chromatic set of pitches.

The final MPS parameter is Variation. This parameter weights the decisions made by the global variation module, which controls a mechanism that causes variation of the other MPS parameters. The most often the parameters will change is once per second, and the least often they will change is every 100 seconds, with the largest possible jump on any MPS scale being 10 units, on a 128 unit scale. This keeps IMP's output from seeming static in content, but helps avoid seemingly random huge shifts in musical space as well.

4 Listener System

The listener module receives in the incoming audio signal from the human via the ADC, performs the audio feature extractions, and sends that extracted feature data to other IMP modules or to The Wekinator. The central component in the listener module is the `analyzer~` object, which is a Max/MSP extension programmed by Tristan Jehan.[5] The `analyzer~` object outputs seven different audio features, and IMP uses five of those: pitch, loudness, brightness, noisiness, and attack.

Pitch is output as frequency in Hz, which is stored in the frequency decider module. Onset detection is done using a combination of pitch and amplitude analysis. A bang is sent out of the attack outlet whenever a new onset is detected. This onset bang serves two important functions. The first is that it cues the current pitch to be sent to the `heard_freq` channel. The second is that it is sent into the `onset_average` subpatcher, which is used to keep a running average of the time between the last ten onsets detected from the human input, as well as to send the elapsed time between each individual onset on the `heard_dur` channel, which goes to the duration decider list of heard durations. The loudness, brightness, and noisiness features are sent via OpenSoundControl[1] to the Wekinator, where they are used to control the timbral interaction system.

5 Creative Algorithm

Equal and interactive participation in an improvised duo musical situation requires that each participant has the ability to both hear and remember what the other has played, the ability the create and introduce new musical content, and the option to choose between the two. IMP's creative algorithm and musical memory systems are what give IMP these abilities. There are two primary modules in the creative algorithm: the frequency decider, and the duration decider. Each functions very similarly, but their processes do not affect each other. The frequency decisions are made independently of the duration decisions, and vice versa.

The beginning of the new event process works similarly on both the frequency and duration deciders. A bang on the `next_now` channel cues each new event. That bang causes a decision to be made as to whether the next event will come from IMPs generative algorithm or IMPs pool of heard events. This is the way in which IMP can play off of what it has heard, or introduce independent new material. The frequency and duration deciders each have two sides: one side is a list of heard data, and the other side is the generative algorithm, or the part that makes IMP's original output.

On the heard data side of the duration decider, each new heard duration is entered into a list that keeps the last ten heard durations. When the `next_now` bang is sent to the heard data side, one of these ten most recently heard durations is selected randomly, and output as the next duration on the `next_dur` channel, which is received in the synthesis module and stored in the amplitude envelope until the `next_now` triggers an event and a new duration is sent. This side of the decider also keeps a running average of the list of heard durations that can be used to change the length and density MPS. This average is scaled to fit the MPS values, and every 500 ms a decision is made to change or not change the MPS based on the current average of heard durations. The average is scaled differently for the length and duration MPS, and the decision to change each MPS is made independently. This system keeps IMP in a similar density and speed area as the human improviser, but does allows for some divergence in terms of these parameters as well. It has the effect of varying how much it seems like IMP is following or ignoring the human.

On the generative algorithm side of the duration decider, IMP chooses a duration proportion from a set of lists of proportions, and that proportion is multiplied by a length factor to get the next duration. There are 15 different files of proportion values, one of which is loaded into the duration `coll` object (`dur_coll`) based on the rhythmic regularity MPS. The lower the rhythmic regularity MPS value the more varied the proportions are. The more similar the duration proportions, the more of a sense of pulse one hears from IMP. There is a rhythmic variation decider that uses the variation MPS to change the choice of proportion `coll` file during the course of an episode. If the `next_now` bang is sent to the generative algorithm side of the duration decider, a proportion is output from the currently loaded duration `coll`.

Once a proportion is sent from the duration `coll`, it is multiplied by a length factor. This length factor is controlled by the length MPS. The shortest duration that IMP will create from its generative algorithm is 50 ms, and the longest is 2500 ms (2.5 seconds). This proportion/length factor system allows IMP to deal separately with the sense of pulse and the sense of speed. Rhythmic regularity with long lengths will feel slow but have pulse and little rhythmic regularity and short lengths will feel fast, but with little feeling of pulse. The length factor is also influenced by the input from the human, so IMP will follow the human's tempo, for the most part, although as was mentioned earlier there is a mechanism in place to keep the following from happening too closely.

The frequency decider has a very similar structure to the duration decider. On the heard data side there is a list of the last ten heard frequencies, and each new heard frequency is added to that list. If a `next_now` bang is routed to the heard data side of the frequency decider, a frequency from the list of the ten most recently heard frequencies is selected. This randomly selected frequency is output as the `next_freq`, and a loop is setup that will output the rest of the list as the next frequencies. For example if the initial `next_now` bang causes the heard frequency in index 7 on the list to be chosen, then the next three frequencies sent will be indexes 8, 9, and 10. After the end of the list is reached, the system resets to choose the next frequency from either the heard data side or the generative algorithm side. This loop system causes IMP to play not just one pitch that it has heard from the human, but a series of pitches, and in the same order that they were heard.

The generative algorithm side of the frequency decider is structured similarly to the generative side of the duration decider. There are 42 different files of sets of pitches, and one of those files is loaded into the frequency `coll` (`freq_coll`) based in the melodicness MPS. The lower numbered sets are major pentatonic scales, and as the numbers go up they cycle through major scales, ascending melodic minor scales, diminished scales, whole tone scales, and finally a chromatic scale. There is a `melody_decider` that changes the choice of frequency `coll` file, according to the variation MPS, during the course of an episode.

Once a frequency is sent out of the current frequency `coll`, a loop is enabled that will select the next 1-5 pitches in a stepwise relationship to the original pitch within the frequency `coll`. The steps may move up or down, or any combination of up and down. This feature gives IMP's output a little more melodic coherence. While it does not eliminate large melodic leaps, it does force at least occasional stepwise motion. Each frequency is sent out on the "`next_freq`" channel, which is received in the synthesis module and stored as the frequency of the carrier oscillator until a `next_now` bang triggers an event and a new frequency is generated.

It should be noted that these frequency `coll` are the one place in IMP where pitch information is stored using MIDI note numbers. Once the MIDI note number is sent out of the `coll`, it passes through a MIDI to frequency conversion, and is handled as a raw frequency from that point on.

6 Machine Learning and Timbral Interaction

Much of the communication between participants in non-idiomatic improvisations is timbral. Timbre is of equal importance as pitch and rhythm in many free improvisation encounters, so it is important that IMP has both a way to hear timbre, and a way to communicate timbre. IMP uses The Wekinator[4], which is a real-time machine learning application, to analyze incoming timbral information, and to send appropriate timbral output data to the synthesis module. While IMP is playing, the Wekinator is running as a separate application. IMPs listener module sends loudness, brightness, and noisiness data to the Wekinator via OSC. The Wekinator runs these three streams of data through a neural network that outputs a single value between 0 and 127, which is sent back to IMP via OSC where it controls the timbral elements of the synthesis module.

The Wekinator must first be trained by playing tones into the feature extractor (which is part of the listener module), and assigning a value between 0 and 127 to each sound played in. This is usually done with 0 being the most pure tone, and 127 being the noisiest tone. However, if one wanted IMP to respond differently in the timbral domain, one could train the Wekinator differently. When IMP gets a 0 from the Wekinator, IMP plays its most pure tone, and a 127 gives its noisiest tone, with the varying degrees in between. With that knowledge, the Wekinator could be trained for any given input to make pure tones or noisy tones, as long as that input is associated with that value in the training stage.

For most of IMPs testing I used a set of training data comprised of solo trombone and solo saxophone recordings. This was done in hope that one universally useful set of training data could be used for all performers with IMP. That may still be possible, but a much larger sample size will be needed, so individual instrument training sets have been devised which have proved to be more accurate with smaller amounts of training data. When a trombonist performs with IMP, the Wekinator is trained on trombone sounds, and when a flutist performs, the training data is all flute sounds, etc.

The value returned by the Wekinator is received in the timbral noise module. This incoming value is in constant flux, so the timbral noise module polls that value every 50 ms and keeps a running average of the ten most recent polled values, and this average is what is used to drive the timbral variations in IMPs sound. Using this running average smooths the data flow, creating a more organic, less scattered result.

The value from the Wekinator is tied to the gain on the second order modulation oscillator in the synthesis module. This means that when the human is playing pure tones, the second order modulation is turned off. As the human's sounds get noisier, the second order modulation depth is increased and IMP's tone gets more strident. After a certain threshold, the harmonicity ratio on the first order modulation begins to change to a non-harmonic ratio as well, which can get quite crunchy. This direct relationship between the timbre of the human input and the timbre of IMP is the way I prefer to play with IMP, but it is entirely dependent on how the Wekinator is trained. Different training data can produce very different results.

7 Conclusion

This outline of the general structure of IMP shows a system that can be expanded and varied with some ease. Each aspect of the analysis and decision making is compartmentalized so that existing aspects may be altered without having to change the entire system, and new features can also be plugged in. At this point in its development, IMP is really just out of the proof of concept stage. IMP has been used with success in public performance, but there are still many areas of planned further development. A system for analyzing the amplitude envelopes of events heard from the human, and incorporating that information into the synthesis module's enveloping system will be the next addition, followed by an expansion of the timbral variance capabilities.

Acknowledgements

I would like to thank my colleagues in the Loyola University New Orleans College of Music and Fine Arts. I'd also like to thank Dr. Stephen David Beck and Dr. Jesse Allison for their advice and guidance. IMP could not have come to be without the input of the musicians who helped me test the software: Janna Saslaw, Mark McGrain, Ray Moore, Rick Trolsen, and Brad Walker.

References

[1] "Open Sound Control" http://opensoundcontrol.org/ (accessed March 19, 2013).
[2] Albert, Jeffrey V. "Interactive Musical Partner: A System for Human/Computer Duo Improvisations." Dissertation, Louisiana State University, 2013.
[3] Chowning, John. "The Synthesis of Complex Audio Spectra by Means of Frequncey Modulation." In Foundations of Computer Music, edited by Curtis Roads and John Strawn, 6 - 29. Cambridge, MA: The MIT Press, 1985.
[4] Fiebrink, Rebecca. "Real-Time Human Interaction with Supervised Learning Algorithms for Music Composition and Performance." Dissertation, Princeton University, 2011.
[5] Jehan, Tristan, "Max/Msp Externals" http://web.media.mit.edu/ tristan/ (accessed January 27, 2013).
[6] Place, Tim and Jesse Allison, "Tap Tools" http://74objects.com/taptools/ (accessed March 19, 2013).

Appendix: The Software Archive

The IMP software package is archived at `http://research.jeffalbert.com/imp/`. The most recent version is available, along with any older variants and links to related publications.