

Gesture-based Human-Robot Jazz Improvisation

Guy Hoffman and Gil Weinberg

Abstract— We present *Shimon*, an interactive improvisational robotic marimba player, developed for research in Robotic Musicianship. The robot listens to a human musician and continuously adapts its improvisation and choreography, while playing simultaneously with the human. We discuss the robot’s mechanism and motion-control, which uses physics simulation and animation principles to achieve both expressivity and safety. We then present a novel interactive improvisation system based on the notion of *gestures* for both musical and visual expression. The system also uses anticipatory beat-matched action to enable real-time synchronization with the human player.

Our system was implemented on a full-length human-robot Jazz duet, displaying highly coordinated melodic and rhythmic human-robot joint improvisation. We have performed with the system in front of a live public audience.

I. INTRODUCTION

This paper describes *Shimon*, an interactive robotic marimba player. *Shimon* improvises in real-time while listening to, and building upon, a human pianist’s performance. The robot’s improvisation uses a *gesture* framework, based on the belief that musicianship is not merely a sequence of notes, but a choreography of movements. Movements which result not only in musical sounds, but also perform visually and communicatively with other band-members and the audience.

A. Context and Prior Work

Shimon is a new research platform for Robotic Musicianship (RM) [1]. We define RM to extend both the tradition of computer-supported interactive music systems, and that of music-playing robotics:

Most computer-supported interactive music systems are hampered by not providing players and audiences with physical cues that are essential for creating expressive musical interactions. For example, in humans, motion size often corresponds to loudness, and gesture location to pitch. These movements are not just used for tone production, but also provide visual feedback and help players anticipate and coordinate their playing. In addition, they create a more engaging experience for the audience by providing a visual connection to the sound. Most computer-supported interactive music systems are also limited by the electronic reproduction and amplification of sound through speakers, which cannot fully capture the richness of acoustic sound [3].

On the other hand, most research in musical robotics focuses mostly on sound production alone, and rarely addresses perceptual and interactive aspects of musicianship, such as

listening, analysis, improvisation, or interaction. Most such devices can be classified in one of two ways: robotic musical instruments, which are mechanical constructions that can be played by live musicians or triggered by pre-recorded sequences [4], [5]; or anthropomorphic musical robots that attempt to imitate the action of human musicians [6], [7]. Some systems use the human’s performance as a user-interface to the robot’s performance [8]; and only a few attempts have been made to develop perceptual, interactive robots that are controlled by autonomous methods [9].

In contrast, in previous work, we have developed a perceptual and improvisatory robotic musician in the form of *Haile*, a robotic drummer [1]. However, *Haile*’s instrumental range was percussive and not melodic, and its motion range was limited to a small space relative to the robot’s body. We have addressed these limitations with *Shimon*, a robot that plays a melodic instrument—a marimba—and does so by covering a larger range of movement [10]. We build on these traits, developing an expressive motion-control system as well as a gesture-based improvisation framework, as described in this paper.

II. ROBOTIC PLATFORM

Several considerations informed the physical design of *Shimon*: we wanted large movements for visibility, as well as fast movements for virtuosity. In addition our goal was to allow for a wide range of sequential and simultaneous note combinations. The resulting design was a combination of fast, long-range, linear actuators, and two sets of rapid parallel solenoids, split over both registers of the instrument.

The physical robot is comprised of four arms, each actuated by a voice-coil linear actuator at its base, and running along a shared rail, in parallel to the marimba’s long side. The robot’s trajectory covers the marimba’s full 4 octaves (Figure 1). The linear actuators are based on a commercial product by IAI and are controlled by a SCON trajectory controller. They can reach an acceleration of $3g$, and—at top speed—move at approximately one octave per 0.25 seconds.

The arms are custom-made aluminum shells housing two rotational solenoids each. The solenoids control mallets, chosen with an appropriate softness to fit the area of the marimba that they are most likely to hit. Each arm contains one mallet for the bottom-row (“white”) keys, and one for the top-row (“black”) keys. *Shimon* was designed in collaboration with Roberto Aimi of *Alium Labs*.

III. MOTOR CONTROL

A standard approach for musical robots is to handle a stream of MIDI notes and translate them into actuator move-



Fig. 1. Overall view and detail view of the robotic marimba player *Shimon*

ments that produce those notes. In *Shimon*'s case, this would mean a note being converted into a slider movement and a subsequent mallet strike. Two drawbacks of this method are (a) an inevitable delay between activation and note production, hampering truly synchronous joint musicianship, and (b) not allowing for expressive control of gesture-choreography, including tonal and silent gestures.

We have therefore separated the control for the mallets and the sliders to enable more artistic freedom in the generation of musical and choreographic gestures, without compromising immediacy and safety. This section describes the two control systems designed for safe artistic expression.

A. Mallets

The mallets are struck using rotational solenoids responding to on/off control through a MIDI interface. Eight MIDI notes are mapped to the eight mallets, and the MIDI NOTE_ON and NOTE_OFF messages are used to activate and deactivate the solenoid.

Given this electro-mechanical setup, we want to be able to achieve a large dynamic range of striking intensities. We also want to be able to strike repeatedly at a high note rate.

Since we can only control the solenoids in an on/off fashion, the striking intensity is a function of two parameters: (a) the velocity gained from the distance traveled; and (b) the length of time the mallet is held on the marimba key.

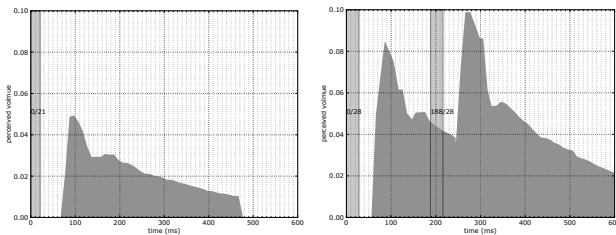


Fig. 2. Empirical strike/sound measurements used to build mallet models. We show one example each for single strike measurement to estimate d_{\downarrow} , d_{\rightarrow} , and i_m (left), and dual strike measurements used to estimate d_{\uparrow} (right).

We therefore need to maintain a model of the mallet position for each striker. In order to do so, we have empirically sampled sound intensity profiles for different solenoid activation lengths, and used those to build a model for each striker (Figure 2). This model includes four parameters:

- d_{\downarrow} — the mean travel time from the rest position to contact with the key,
- d_{\uparrow} — the mean travel time from the down position back to the rest position,
- d_{\rightarrow} — the hold duration that results in the highest intensity note for that particular mallet, and
- i_m — the duty cycle that results in the highest intensity note for that mallet, when it starts from the resting position.

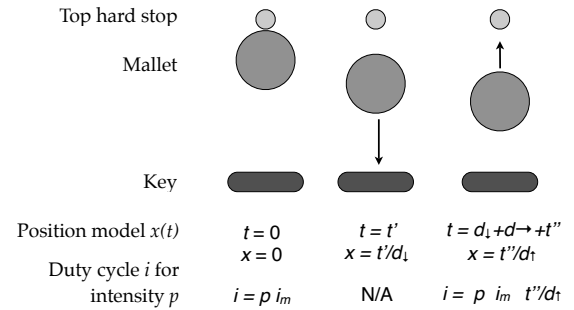


Fig. 3. Duty-cycle computation based on mallet position model

Using this model, each of the eight mallet control modules translates a combination of desired auditory intensity and time of impact into a solenoid duty cycle. Intuitively—the lower a mallet is at request time, the shorter the duty cycle needs to be to achieve impact, and to prevent muting of the key through a prolonged holding time.

An estimated position x is thus dynamically maintained based on the triggered solenoid commands, and the empirical mallet model (Figure 3). During up-travel, $x(t)$, with t being the time since the last mallet activation start, is estimated as

$$x(t) = \frac{t - d_{\downarrow} - d_{\rightarrow}}{d_{\uparrow}}$$

As a result, the updated duty cycle i as a function of the desired intensity p , is then

$$i = p \times i_m \times x(t)$$

In the above equation, we approximate the mallet position as a linear function of travel time. Obviously, a more realistic model would be to take into account the acceleration of the mallet from the resting position to the key impact. Also,

bounce-back should be accounted for, for short hold times. We leave these improvements for future work.

The described system results a high level of musical expressivity, since it (a) maintains a finely adjustable dynamic striking range, and (b) allows for high-frequency repetitions for the same mallet, during which the mallet does not travel all the way up to the resting position.

B. Sliders

The horizontally moving sliders are four linear carriages sharing a rail and actuated through voice coil actuators under acceleration- and velocity-limited trapezoid control.

There are two issues with this control approach. (a) a mechanical (“robotic”—so to speak) movement quality associated with the standard fire-and-forget motion control approach, and (b) collision-avoidance, since all four arms share one rail.

1) *Animation Approach*: To tackle these issues, we chose to take an *animation* approach to the gesture control. Based on our experience with previous robots (e.g. [11], [12]) we use a high-frequency controller that updates the absolute position of each slider at a given frame rate. This controller is fed position data for all four arms at a lower frequency, based on higher-level movement considerations.

This approach has two main advantages: (a) for each of the robotic arms, we are able to generate a more expressive spatio-temporal trajectory than just a trapezoid, and we can add animation principles such as ease-in, ease-out, anticipation, and follow-through [13]; and (b) since the position of the sliders is continuously controlled, collisions can be avoided at the position request level.

2) *Slider Manager: PID and simulated springs*: The intermediate layer that handles the slider position requests and generates the positions for each of the four sliders, while maintaining collision safety, is called the *Slider Manager*. It enables higher-level modules to “lock” a slider, and thus control it for the duration of the locking period.

The Slider Manager then uses a combination of PID control for each slider, with a simulated spring system between sliders, to update the position of all four sliders during each update cycle (Figure 4). For each position request of a locked slider at position x , we first calculate the required PID force using the standard PID formula:

$$F_{PID} = K_p e + K_i \int_0^t e d\tau + K_d \frac{de}{dt}(t)$$

In addition to the PID force, the Slider Manager models “virtual springs”, which help prevent collisions and move unlocked sliders out of the way in a naturally-seeming fashion. Based on the current position of the carriages, the length of the virtual springs, and thus their current simulated compression, we add the spring component kx to the force. The force update for each carriage is then

$$F_{PID} - kx$$

where the sign of kx is determined by the side of the simulated spring.

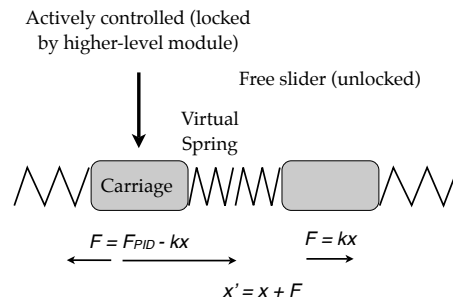


Fig. 4. Interaction between PID control and simulated spring model

The result of this control approach is a system that is both safe—carriages will never collide and push each other out of the way—and expressive.

While it is expected that higher-level modules will not cross over carriages, and be generally coordinated, adding this middle-layer control system allows more freedom of expressivity on a higher-level, for example inserting pre-scripted animations, and parametric gestures, while keeping the system safe at all times.

IV. GESTURES AND ANTICIPATION

A main contribution of this paper is modeling interactive musical improvisation as *gestures* instead of as a sequence of notes. Using gestures as the building blocks of musical expression is particularly appropriate for robotic musicianship, as it puts the emphasis on physical movement and not on symbolic note data. Gestures have been the focus of much research in human musicianship, often distinguishing between tone-generating and non-tone-generating gestures [2]. A physical-gestural approach is also in line with our embodied view of human-robot interaction [14].

Our definition of gesture deviates from the common use of the word in human musicianship. In this paper, a “gesture” is a physical behavior of the robot which may or may not activate the instrument. A gesture could be a cyclical motion of one arm, a simple “go-to and play” gesture for a single note, or a rhythmic striking of a combination of mallets. Since gestures are not determined by notes, but by the robot’s physical structure, *Shimon*’s gestures separately control the timing of the mallet strikers and the movement of the sliders.

A. Anticipatory Action

To allow for real-time synchronous non-scripted playing with a human, we also take an anticipatory approach, dividing each gesture into *preparation* and *follow-through*. This principle is based on a long tradition of performance, such as ensemble acting [15], and has been explored in our recent work, both in the context of human-robot teamwork [16], and for human-robot joint theater performance [11].

By separating the—potentially lengthy—preparatory movement (in our case: the horizontal movement) from the almost instant follow-through (in our case: the mallet action), we can achieve a high level of synchronization and beat keeping without relying on a complete-musical-bar delay of the system.

V. IMPROVISATION

Implementing this gesture-based approach, we have developed a Jazz improvisation system, which we employed in a human-robot joint performance. In our system, a performance is made out of *interaction modules* each of which is an independently controlled phase in the performance. It is continuously updated until the part’s end condition is met. This is usually a perceptual condition, but can also be a pre-set amount of bars to play.

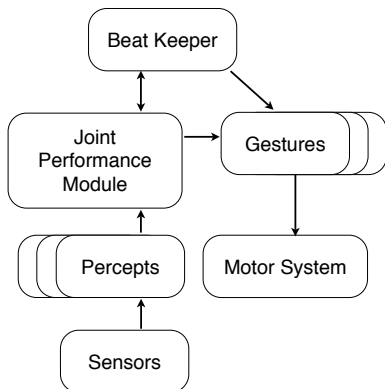


Fig. 5. Schematic interaction module for each phase of the performance

Figure 5 shows the general structure of an interaction module. It contains a number of gestures which are either triggered directly, or registered to play based on the current beat, as managed by the beat keeper.

Gestures are selected and affected by information coming in from percepts, which analyze input from the robot’s sensory system. These percepts can include, for example, a certain note density, or the triggering of a particular phrase or rhythm.

A. Infrastructure

1) *MIDI Listener*: While there are a number of sensory modules possible, we are currently using a MIDI sensory input, responding to the notes from a MIDI-enabled electric piano. On top of this sensor, we developed several perceptual modules described later in this section.

2) *Beat Keeper*: Common to all parts, and continuously running is the *Beat Keeper* module, which serves as an adjustable metronome that can be dynamically set and reset during play, and calls registered callback functions in the parts and the gestures making up the performance.

To provide a simple example: a “one-three-random-note” gesture could register to get called on every “one” and “three” of a bar. In between calls it would “prepare” into a certain random position, and then would activate the appropriate striker on the callback-registered beats.

3) *Chord Representation*: We use three kinds of representations for Jazz chords in our system. The simple representation is that of a fixed set of notes in the robot’s playing range. The second representation is octave-agnostic and includes a set of notes and all their octave harmonics. Finally, we also represent chords as a base note with a set of

set-octave or octave agnostic harmonics. We can parse these from string representation in traditional Jazz notation (e.g. “Cm7”, “BbM6”).

B. Module I: Call-and Response

The first interaction module is the phrase-call and chord-response module. In this module, the system responds to a musical phrase with a pre-set chord sequence, arranged in a particular rhythmic pattern. The challenge here is not to select the right notes, but to be able to respond in time and play on a seamlessly synchronized beat and onset to that of the human player, who can vary the tempo at will.

This module makes use of the anticipatory structure of gestures. During the sequence detection phase, the robot prepares the chord gesture. When the phrase is detected, the robot can strike the response almost instantly, resulting in a highly meshed musical interaction.

This module includes two kinds of gestures:

Simple chord gestures —

select an arm configuration based on a given chord during the preparation stage, and strike the prepared chord in the follow-through stage. If the chord is a set chord, the configuration is set. If it is a flexible chord, the gesture will pick different configurations satisfying the chord at different times.

Rhythmic chord gestures —

are similar to the simple chord gestures in preparation, but during follow-through will strike the mallets in a pre-set pattern. This can be an arpeggiated sequence, or any other rhythmic structure.

The robot adapts to the call phrase using a simultaneous *sequence spotter* and *beat estimator* percept. Using an on-beat representation of the sequences that are to be detected, we use a Levenshtein distance metric [17] with an allowed distance $d = 1$ to consider a phrase detected¹.

At that stage, the beat estimator will estimate both the played beat based on the duration of the sequence, and the beat synchronization based on the time of the last note played. These are transmitted to the beat keeper, which will execute a sequence of simple and rhythmic chords, as beat callbacks. The result is an on-sync, beat-matched call-and-response pattern, a common interaction in a Jazz ensemble.

C. Module II: Opportunistic Overlay Improvisation

A second type of interaction module is the *opportunistic overlay improvisation*. This interaction is centered around the choreographic aspect of movement with the notes appearing as a side-effect of the performance. The intention of this module is to play a relatively sparse improvisation that is beat-matched, synchronized, and chord-adaptive to the human’s playing.

The central gesture in this module is a rhythmic cyclical movement gesture that takes its synchronization from the currently active beat in the beat keeper module. This beat

¹Naturally, we do not allow the last note in the phrase to be deleted for the purposes of comparison, as this would invalidate the synchronization

is updated through a beat detection percept tracking the beat of the bass line in the human playing, using a simple bass-note temporal interval difference, modeled as either a full, a half, or a quarter bar based on the previous beat. In parallel, a chord classification percept is running, classifying the currently played chord by the human player, by finding a best fit from the chords that are part of the current piece.

Without interrupting the choreographic gesture, this interaction module attempts to opportunistically play notes that belong to the currently detected chord, based on a preset rhythmic pattern. If the rhythmic pattern is in a “beat” state, and one or more of the mallets happen to be in a position to play a note from the detected chord, those mallets strike.

Since both the choreographic gesture and the rhythmic strike pattern are activated through a shared beat keeper, the result is a confluence of two rhythms and one chord structure, resulting in a novel improvisational gesture which is highly choreographic, can only be conceived by a machine, and is tightly synchronized to the human’s playing.

D. Module III: Rhythmic Phrase-Matching Improvisation

The third interaction module that we implemented is a *rhythmic phrase-matching improvisation* module. As in the previous section, this module supports improvisation that is beat- and chord-synchronized to the human player. In addition, it attempts to match the style and density of the human player, and generate improvisational phrases inspired by the human playing.

Beat tracking and chord classification is done in a similar fashion as the in the opportunistic overlay improvisation: The timing and pitch of the bass notes are used for detecting the beat, for synchronizing the downbeats of the human’s playing, as well as for chord classification.

In addition, this module uses a decaying-history probability distribution to generate improvisational phrases that are rhythm-similar to phrases played by the human. The main gesture of this part selects—in each bar—one of the arm positions that correspond to the currently classified chord. This is the gesture’s anticipatory phase.

When in position, the gesture then plays a rhythmic phrase tempo- and sync-matched to the human’s performance. Each arm plays a different phrase. Arm i plays a phrase based on a probabilistic striking pattern, which can be described as a vector of probabilities

$$\vec{p}_i = \{ p_0^i \quad p_1^i \quad \cdots \quad p_k^i \}$$

where k is the number of quantizations made. E.g.—on a 4/4 beat with 1/32 note quantization, $k = 32$. Thus, within each bar, arm i will play at time j with a probability of p_j^i .

This probability is calculated based on the decayed history of the human player’s quantized playing patterns of the human player. The system listens to the human player’s last beat’s improvisation, quantizes the playing into k bins, and then attempts to cluster the notes in the phrase into the number of arms which the robot will use. This clustering is done on a one-dimensional linear model, using only the note pitch as the clustering variable.

Once the clusters have been assigned, we create a human play vector

$$\vec{h}_i = \{ h_k^i \} = \begin{cases} 1 & \text{if a note in cluster } i \text{ was played at time } k \\ 0 & \text{otherwise} \end{cases}$$

The probability p_j^i is then updated inductively as follows:

$$p_j^i = h_j^i \lambda + p_j^i (1 - \lambda)$$

where λ is the decay parameter.

The result is an improvisation system which plays phrases influenced by the human playing’s rhythm, phrases, and density. For example, if the human plays a chord rhythm, then the vectors \vec{h}_i would be identical or near-identical for all clusters, resulting in a robot improvisation that will be close to a chord rhythm. However, there is variance in the robot’s playing since it is using the human phrases as a probability basis, therefore changing the pattern that the human plays. Also, since the arm positions change according to the current harmonic lead of the human, and the robot’s exploration of the chord space, the phrases will never be a precise copy of the human improvisation but only rhythmically inspired.

Moreover, as the probability vectors mix with data from earlier history, the current playing of the robot is always a combination of all the previous human plays. The precise structure of the robot’s memory depends on the value of λ .

Another example would be the human playing a 1–3–5 arpeggio twice in one bar. This would be clustered into three clusters, each of which would be assigned to one of the arms of the robot, resulting in a similar arpeggio in the robot’s improvisation.

An interesting variation on this system is to re-assign clusters not according to their original note-pitch order. This results in the maintenance of the rhythmic structure of the phrase but not the melodic structure. In the performance described below, we have actually used only two clusters and assigned them to cross-over arms, i.e. cluster 0 to arms 0 and 2 and cluster 1 to arms 1 and 3.

Note that this approach maintains our focus on *gestures* as opposed to note sequences, as the clustering records the human’s rhythmic gestures, matching different spatial activity regions to probabilities, which are in turn used by the robot to generate its own improvisation. Importantly—in both improvisation modules—the robot never maintains a note-based representation of the keys it is about to play.

VI. LIVE PERFORMANCE

We have used the described robot and gesture-based improvisation system in a live performance before a public audience. The show occurred on April 17 2009 in Atlanta, GA, USA. The performance was part of an evening of computer music and was sold-out to an audience of approximately 160 attendants.

The performance was structured around a “Jordu”, a Jazz standard by Duke Jordan. The first part was an adaptive and synchronized call-and-response, in which the pianist would prompt the robot with a number of renditions of the piece’s opening phrase. The robot detected the correct phrase



Fig. 6. A live performance of the robot *Shimon* using the gesture-based improvisation system was held on April 17th 2009 in Atlanta, GA, USA.

and, using preparatory gesture responded on beat. A shorter version of this interaction was repeated between each of the subsequent performance segments.

The second phase used the introduction's last detected tempo to play a fixed-progression accompaniment to the human's improvisation. Then the robot started playing in *opportunistic overlay improvisation* taking tempo and chord cues from the human player while repeating an "opening-and-closing" breathing-like gesture, over which the rhythmic improvisation was structured.

The next segment employed *rhythmic phrase-matching improvisation*, in which the robot adapted to the human's tempo, density, style, chord progression, and rhythmic phrases. Our gesture-based approach enabled the robot to adapt without noticeable delay while maintaining an overall uninterrupted visual motion arc, and seem to be playing in interactive synchrony with the human player.

An interesting result of this improvisation was a constant back-and-forth inspiration between the human and the robotic player. Since the robot's phrases were similar, but not identical to the human's phrases, the human picked up the variations, in return influencing the robot's next iteration of rhythms.

Finally, a pre-programmed crescendo finale led to the end-chord, which was an anticipatory call-and-response, resulting in a perceived synchronous end of the performance.

The overall performance lasted just under seven minutes. Video recordings of the performance ([18], [19]; also attached to this paper) were widely covered to acclaim by the press and viewed by an additional audience of approximately 40,000 online viewers.

VII. CONCLUSION AND FUTURE WORK

We presented *Shimon*, an interactive improvisational robotic marimba player developed for research in Robotic Musicianship. We discussed the musically and visually expressive motor-control system, and a gesture-based improvisation system. The design of these systems stems from our belief that musical performance is as much about visual choreography and communication, as it is about tonal music generation.

We have implemented our system on a full human-robot Jazz performance, and performed live with a human pianist

in front of a public audience.

We are currently using *Shimon* to empirically study some of our hypotheses brought forth in this paper. These include an evaluation of the gesture system vis-a-vis a note-generation motor control; as well as the effects of robotic presence on both band members and audience.

Additionally, we are developing a more predictive anticipatory system to allow the robot to use past interactions to generate preparatory gestures, based on our findings on anticipatory human-robot interaction in [20], [16].

REFERENCES

- [1] G. Weinberg and S. Driscoll, "Toward robotic musicianship," *Computer Music Journal*, vol. 30, no. 4, pp. 28–45, 2006.
- [2] C. Cadoz and M. M. Wanderley, "Gesture - music," in *Trends in Gestural Control of Music*, M. M. Wanderley and M. Battier, Eds. Paris, France: Ircam - Centre Pompidou, 2000, pp. 71–94.
- [3] R. Rowe, *Machine musicianship*. Cambridge, MA: MIT Press, 2001.
- [4] E. Singer, K. Larke, and D. Bianciardi, "Lemur guitarbot: Midi robotic string instrument," in *NIME '03: Proceedings of the 2003 conference on New interfaces for musical expression*. Singapore, Singapore: National University of Singapore, 2003, pp. 188–191.
- [5] R. B. Dannenberg, B. Brown, G. Zeglin, and R. Lupish, "Mcblare: a robotic bagpipe player," in *NIME '05: Proceedings of the 2005 conference on New interfaces for musical expression*. Singapore, Singapore: National University of Singapore, 2005, pp. 80–84.
- [6] J. Solis, K. Taniguchi, T. Ninomiya, T. Yamamoto, and A. Takanishi, "The waseda flutist robot no 4 refined IV: enhancing the sound clarity and the articulation between notes by improving the design of the lips and tonguing mechanisms," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007)*. IEEE, 2007, pp. 2041–2046.
- [7] Toyota, "Trumpet robot," <http://www.toyota.co.jp/en/special/robot/>, 2004.
- [8] K. Petersen, J. Solis, and A. Takanishi, "Toward enabling a natural interaction between human musicians and musical performance robots: Implementation of a real-time gestural interface," in *Proceedings of the 17th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN 2008)*, 2008.
- [9] N. Baginsky, "The three sirens: a self-learning robotic rock band," <http://www.the-three-sirens.info/>, 2004.
- [10] G. Weinberg and S. Driscoll, "The design of a perceptual and improvisational robotic marimba player," in *Proceedings of the 18th IEEE Symposium on Robot and Human Interactive Communication (RO-MAN 2007)*, Jeju, Korea, August 2007, pp. 769–774.
- [11] G. Hoffman, R. Kubat, and C. Breazeal, "A hybrid control system for puppeteering a live robotic stage actor," in *Proceedings of the 17th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN 2008)*, 2008.
- [12] G. Hoffman and C. Breazeal, "Collaboration in human-robot teams," in *Proc. of the AIAA 1st Intelligent Systems Technical Conference*. Chicago, IL, USA: AIAA, September 2004.
- [13] J. Lasseter, "Principles of traditional animation applied to 3d computer animation," *Computer Graphics*, vol. 21, no. 4, pp. 35–44, July 1987.
- [14] G. Hoffman and C. Breazeal, "Robotic partners' bodies and minds: An embodied approach to fluid human-robot collaboration," in *Fifth International Workshop on Cognitive Robotics, AAAI'06*, 2006.
- [15] S. Meisner and D. Longwell, *Sanford Meisner on Acting*, 1st ed. Vintage, August 1987.
- [16] G. Hoffman and C. Breazeal, "Anticipatory perceptual simulation for human-robot joint practice: Theory and application study," in *Proceedings of the 23rd AAAI Conference for Artificial Intelligence (AAAI'08)*, July 2008.
- [17] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Soviet Physics Doklady*, vol. 10, p. 707, 1966.
- [18] G. Hoffman, "Human-robot jazz improvisation (highlights)," <http://www.youtube.com/watch?v=jqcoDECGde8>, April 2009.
- [19] —, "Human-robot jazz improvisation (full performance)," <http://www.youtube.com/watch?v=qy02lwwGv3U>, April 2009.
- [20] G. Hoffman and C. Breazeal, "Cost-based anticipatory action-selection for human-robot fluency," *IEEE Transactions on Robotics and Automation*, vol. 23, no. 5, pp. 952–961, October 2007.