Imitation Framework for Percussion

Özgür İzmirli
Connecticut College
New London, CT, 06320, USA
oizm@conncoll.edu

Jake Faris
Connecticut College
New London, CT, 06320, USA
jfaris@conncoll.edu

ABSTRACT

We present a framework for imitation of percussion performances with parameter-based learning for accurate reproduction. We constructed a robotic setup involving pull-solenoids attached to drum sticks which communicate with a computer through a microcontroller. The imitation framework allows for parameter adaptation to different mechanical constructions by learning the capabilities of the overall system being used. For the rhythmic vocabulary, we have considered regular stroke, flam and drag styles. A learning and calibration system was developed to perform grace notes for the drag rudiment as well as single strokes and the flam rudiment. A second pre-performance process was introduced to minimize the latency difference between individual drum sticks. We also developed an off-line onset detection method to recognize onsets from the microphone input. Once these pre-performance steps are taken, our setup will then listen to a human drummer's performance, analyze for onsets, loudness, and rudiment pattern, and then play back using the learned parameters for the particular system. We conducted three different evaluations of our constructed system.

Keywords

Imitation, robotic percussion, calibration

1. INTRODUCTION

Computer assisted musical applications have become indispensible in the lives of many performers. These applications are either in the form of software that responds to and interacts with the user through a display and speakers, or they can actually employ electromechanical devices to accomplish tasks that the former type cannot. We are interested in studying computer assisted music performance systems, actual percussion systems in particular, within this rapidly growing interdisciplinary field. Reasons for utilizing physical systems instead of audio output from a computer are that these systems can produce spatial sounds due to their sound radiation pattern, are perceived as having better presence when they are in the same room as the listener and usually have higher dynamic range compared to CD quality reproduction systems. Another advantage of these systems is that they can repeat drumming patterns with high accuracy enabling one to study the effects of different performance parameters such as tempo and loudness for the same pattern. Robotic percussion systems also make possible a more realistic rendition of the drumming sound compared to mixing isolated recorded sounds.

Our approach entails the analysis and imitation of human performances in order to gain an understanding toward building better real-time performance systems. With many drum machines on the market, metronomic performance is usually easy to achieve but there are discernible qualities present in musicians' performances that need to be captured to develop more compelling systems. Similar shortcomings are present in non-parametric systems and sound reproduction equipment in that they cannot respond to changing dynamics of a performance and hence their use in real performances remain limited. Our long term goal is to study these two aspects by first analyzing human performances and imitating them through mechanical actuation, and second by building an adaptive system that has the capability of remaining in pace with a performance.

In this work we have focused on the first part of this long-term goal. We are interested in studying percussion performance on a snare drum and we present an imitation framework that can cater to different mechanical constructions. The framework involves a learning stage by which relevant parameters are optimized for the given physical configuration of the electromechanical player. We have constructed two actuator controlled drum sticks which operate on the snare drum. In the reminder of the paper we describe the refinement to the control mechanism of these kinds of systems motivated by the shortcomings of simple trigger-based control methods.

Professional human performers are able to seamlessly track tempo, perform various patterns, and modify their individual strokes based on a variety of different factors. Aside from a basic style where drum sticks hit the drum head according to a simple rhythmic pattern, there are several basic skills a drum player usually employs. These can be classified into four parts: roll, diddle, flam and drag rudiments, each of which contains several variations. In this work, we are interested in classifying these rudiments to fit actuation capabilities of robotic drumming systems. For example, we have explored the ways a drag on a snare drum could be played to sound most like a human. We assume that mixtures such as the ratamacue can be reduced to a drag and stokes, and similarly the paradiddle and flam rudiments can be reduced to combinations of regular strokes. We have excluded roll variants from our study.

The ability to replicate actual drumming depends largely on the mechanical setup of the machine performer. Our system uses heavy duty pull-solenoids attached to drumsticks, which, when turned on, bring the drumstick down to hit the snare drum with the necessary speed. Large solenoids are needed to achieve loud sounds and good dynamic range. Several problems in computer controlled performance include the limited precision of force control that is available, the force being unidirectional, and inertia. At the same time, we are trying to test the feasibility of our system without making it overly complicated by employing positional feedback from the drumstick. Our goal for this paper is to integrate flam and drag rudiments into a playback performance system which otherwise tracks and recognizes regular stroke patterns autonomously.

2. RELATED WORK

Problems such as tempo tracking, onset detection, and pattern analysis have been researched extensively and continue to be researched. Researchers Hochenbaum and Kapur have discussed various options for onset detection [7]. They ran an experiment to differentiate onsets between performer's individual hands using accelerometers against a simple audio analysis similar to that of our experiment. Similarly, Han et al. discussed various other methods to find onsets directly from an audio input, using the Fourier Transform and Spectral Flux analysis [6]. Their research ultimately used the Spectral Flux analysis designed by Dixon [4]. Dixon explored five different onset detection methods, including Spectral Flux, Phase Deviation, Complex Domain, Weighted Phase Deviation, and Rectified Complex Domain. Many of these techniques share similar

NIME'14, June 30 – July 03, 2014, Goldsmiths, University of London, UK.

approaches to that of Bello, et al. [1], who developed onset detection systems that cater to a variety of different audio qualities. At the core of all of these concepts lie three components: post-processing, thresholding, and peak-picking. The use of phase and energy for onset detection is a complex process according to Bello [2]. Here, the researchers discuss phase-based onset detection as well as detection in the complex domain. Input signals have also been analyzed by Marchini and Purwins, who developed a system that recognizes rhythmic patterns and tracks the tempo of percussive audio, and replay variations using those parameters [8].

Derbinsky and Essl discuss many features of tempo tracking needed for their Mobile Percussion Collaboration project [3]. Uhle and Herre search for the tempo of a percussive performance through bar length estimation, from which they extract the tempo of the performance [10]. They use a combination of onset and tatum estimation along with periodicity estimation to approximate bar length and thus the time signal.

Researchers Murphy et al. elaborated on the challenges and possible solutions on creating a more efficient drum system through the use of calibration. They employed various search techniques in an attempt to create a system which produces an approximately linear force output based on a linear input. They used three different search methods, including a pattern search which would decrease step size the closer the calibration was to optimal. Their conclusion was that this pattern search was the most efficient among the searches they had considered [9].

The playback of a percussive performance relies on pattern recognition in addition to simply onset recognition and imitation. Gillet and Richard researched the patterns of drum loops specifically and approaches to categorize them [5]. They enact Hidden Markov Models and Support Vector Machines upon the drum audio entities in a database and determine what instrument or combination of instruments the audio file being analyzed contains.

3. THE SYSTEM

Control systems and the physical actuators that they control are usually co-developed making them difficult to replace or tune the mechanical components. This creates challenges in maintaining the quality of the performance when the actuators need to be replaced, components are changed, springs are replaced and even when the system is transported and set up. One way to retain the quality of the performance is through an automatic recalibration when the need arises. Below we present the details of a framework for percussion performance that allows for component replacements and tuning as well as changes in the relative positions of components and musical instruments.

3.1 Interactive Imitation Framework

The imitation framework employs an audio analysis subsystem, an actuation control subsystem and physical actuators as shown in Figure 1.

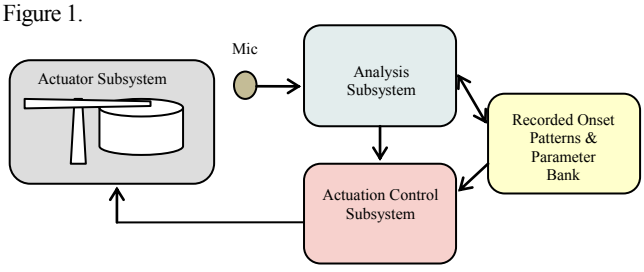

Figure 1. Components of the imitation framework.

There are three modes of operation: 1) Calibration and learning of performance parameters 2) Analysis and pattern recognition 3) Performance through imitation or database access. For example, auto-calibration using mode 1 can be employed for a particular setup configuration of the actuator subsystem with choice of springs, height, position and angle of the drum. This mode uses the analysis - actuation control - actuator subsystem loop. Mode 2 is specific to the analysis of the incoming sound from the microphone. Mode 3 is employed for the rendition of the drumming pattern produced by the analysis subsystem and optionally to render drum patterns from symbolic score-like representations.

3.2 Actuation Control & Actuation

We have constructed a two drumstick/snare system that uses heavy duty solenoids. Each drumstick is attached to a solenoid via a fulcrum dividing the drum stick with a particular leverage ratio. The rear tip of the drumstick is connected to the frame via a spring or elastic band which keeps the tip of the drum stick away from the drum head by providing force in the opposing direction of the solenoid actuation. Both solenoids are controlled with an Arduino microcontroller connected to a computer running MATLAB. The actuation system can be seen in Figure 2.

Figure 2. Drum stick/snare system constructed for this study.

The electrical signals, controlled by the Arduino, are sent to the solenoid which pulls the drumstick rapidly to hit the snare drum. Each solenoid's pull-strength is controlled using pulse-width modulation (PWM) from the Arduino. The modulated output is connected to a MOSFET transistor that in turn drives the solenoid. This facilitates time varying force control on the solenoid during operation. The solenoids are connected to independent power supplies so that a current surge for one solenoid does not affect the performance of the other.

3.3 Onsets & Analysis

A necessary component of a performance system is the ability to detect and predict the onsets of a performance – whether it is the original human performance or machine rendered performance. In our single snare drum setup, audio is input through a microphone into our processing system. The onset detection algorithm attempts to detect all events for which a drum stick hits the drum. This algorithm detects onsets based on the change of energies in a range of frequencies in spectral domain. In our implementation, the sampling rate is 22050 Hz. Heavily overlapped, Hanning-windowed 256 point FFTs are calculated for the input audio – hop length is 1.1 msecs. Only the FFT bins with center frequencies lower than 1.7 kHz. are retained. This cut-off frequency has been empirically determined and it corresponds to the range of frequencies that capture a significant portion of the energies necessary for onset detection. For each FFT output (frame), the amplitude spectrum is squared and summed over the frequency range to give the frame-by-frame energies in sequence ***b***. These points are then smoothed using a 40 point Gaussian window to form sequence ***c***. The resulting sequence of points is then differenced by subtracting the element at ***i*** from element at ***i***+1 to obtain the final sequence ***d***. Sequence ***d*** is then analyzed for peaks that exceed a certain threshold so that only prominent, non-noise peaks are marked. Another condition for a peak to be marked is for it to have significant energy in the corresponding audio at that point in time. The energies are estimated with an energy function that returns the sum of the squares of the input signal for a window of length 50 msecs. starting at the registration point. These are used as intensity estimates or a measure of how fast the stick has hit the drum head

which are correlated with the loudness of the particular event that led to the onset. Figure 3 shows an example of onset detection for three flams in rapid succession. The circles on the peaks of sequence ***d*** in the bottom plot represent the onset times. Note that the height of the peak indicates the confidence or clarity of the onset rather than the intensity, which is calculated separately with the energy function.

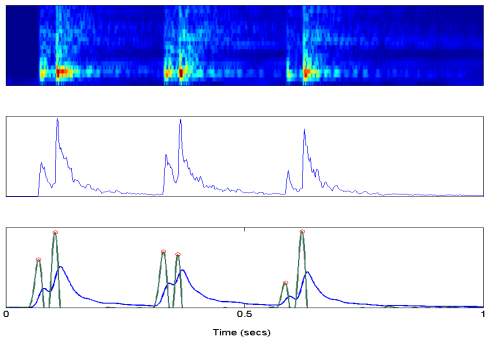

Figure 3. Example of onset analysis for an input with three consecutive flams. Spectrum (top), sequence *b* (middle), and sequences *c, d* and onsets shown as circles (bottom).

After all onsets in a given performance have been detected, the algorithm must make the distinction between drags, flams and strokes and label them accordingly. To do this, it examines the relationship between the onsets and tests for patterns in a particular order. For each onset, conditions for a drag are tested first. Drags, containing 2 grace notes and one regular stroke, are detected by a group of 3 onsets within a specific proximity to each other. An additional condition for drag detection is that the average of the energies of the two grace notes be lower than a percentage of the energy of the stroke that follows them – 50 percent has worked well. Next, if an onset does not pass the drag test it is checked against conditions for a flam which are treated the same way, except in a set of two and with a smaller time proximity. Finally, single strokes are detected when single onsets exist separately from their adjacent onsets along the timeline. The output of this stage is a sequence of onset times, intensity estimates and rudiment labels. For playback, this data is sent to the performance portion of our program governed by the actuation control and actuation subsystems.

Knowing the type of rudiment and the initial onset time, the system re-interprets the data in a format compatible with our physical setup via Arduino. The methods by which the parameters are learned are explained in the next section. It is worth noting that the grace notes for a drag are notably distinct than the flam and the single strokes. Therefore, a separate code is executed for drags. Now converted into a symbolic representation, the pattern is stored for future use and timed signals can be sent to the actuation subsystem at a later time. These onsets are offset by a fixed amount of time to account for the latency between the signal output on the computer and the actual performance as calibrated previously.

3.4 Drag & Stroke Calibration

In absence of position feedback it is hard to produce precisely controlled force outputs from the solenoid; hence, dynamic force control is possible but needs to be critically determined. These issues usually plague solenoid-based setups. In our case, this is especially problematic in the performance of drag rudiments because the grace notes in a drag are significantly lighter and have inter-onset times that may be smaller than the solenoid's full swing time. That is, since mainly the spring stiffness determines the recovery time of the drum stick it may not be possible to complete two full swings of the drum stick in the amount of time required to reproduce a drag (with the same stick). To counter this inefficiency, we have developed a calibration system which finds the optimal signal to send to the solenoid that will best imitate the sound of drag grace notes. We send a square wave (for force modulation – the PWM is at a much higher frequency) with varying parameters to the solenoid (via Arduino) and determine the effectiveness of the set of parameters through analysis of the sound produced by the actuation. We change the signal by altering three components: the duty cycle (***D***), the period length (***L***) of the square wave, and the average current channeled to the solenoid (***A***) essentially given by the amplitude written to the Arduino.

Due to the manual nature of constructing these kinds of performance machines, every machine must be calibrated individually. The small differences in each machine's construction can affect performance in a significant way. It is important to distinguish that each drum stick's latency must also be calculated individually. The exact relationship between the solenoid, drum stick, and spring may vary from setup to setup, and the signal's timing must account for this. We created a secondary calibration system which aims to find the exact latencies between two actuators such that it is possible to hit the drum with both sticks simultanously. The results of this calibration are then implemented in the performance of the system to increase the accuracy of the playback timing.

3.4.1 Parameters for a Single Stroke

The learning of parameters for regular stroke activation involves only loudness control in addition to the onset time. Our experiment mainly deals with the range of durations during which the full current is applied to control the speed of the hit. In other words, we are only controlling the duration of a single pulse applied to the solenoid. This duration is nonlinearly related to the intensity of the stroke. Hence, a mapping function is learned by individually activating the solenoids for a range of durations and measuring the energies with the same method described in the onset detection section. Once all values are recorded, we invert the function and interpolate to obtain the energy to amplitude mapping function ***f***. This mapping function is used to convert the calculated energy (***e***) of the input stroke sound to the actuation amplitude level. ***A***, for an imitation: ***A=f*** (***e***). The learning is done separately for each solenoid.

3.4.2 Flams

The flam rudiment can be thought of as two single strokes played with different drum sticks. In a flam, the first stick that hits the drum head is usually much quieter than the second and this can be reflected in the imitation through the stroke parameters. Unlike the drag that is described below, flams are played with two sticks in succession and therefore arbitrarily small time intervals between the hits can be achieved during actuation.

3.4.3 Learning a Single Drag Rudiment

In this section, we will outline the parameter learning method for a single drag and we will explain the generic drag calibration in the next section. The learning is based again on a parameter search. The input sound is analyzed for onset times and energy estimates. A parameter search is then performed by rendering the imitation and doing another analysis for onsets and energies on this rendition. The degree of fit is estimated by the following distance function that uses timing information as well as energy estimates:

$$dist = \frac{\beta}{K}\sum_{k=1}^{K}\left|r_k - n_k\right| + corr(R,N) \qquad (1)$$

Where r_k is the rendered onset time, n_k is the input onset time, R is the vector of energy estimates calculated from the rendition, N is the vector of energy estimates calculated from the input, β is a scaling parameter that controls the balance of timing vs. energy sequence similarity, and *corr* is the correlation function that calculates the Pearson correlation coefficient between R and N. The parameter search tries to minimize the distance function by systematically applying parameters within the search space. Since the error surface is not smooth we could not employ hill-climbing type of methods for more efficient searching. Figure 4 shows the time signal and corresponding spectra of the original input and its imitation rendition for a set of parameters during the search process.

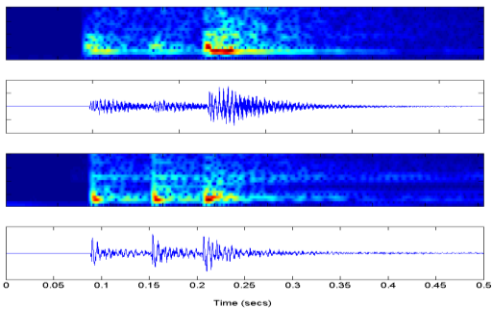

Figure 4. An example of spectra and time signals of the input drag (top two) and the rendered drag (bottom two).

3.4.4 Learning Generalized Drags

The previous section explained how the parameters for a single drag rudiment could be learned. However, the parameters from this experiment cannot be used in a general imitation system. The drag needs to be further parameterized so that the rate of the grace notes as well as the loudness can be controlled. Similar to the stroke and single drag parameter learning, a three parameter search is conducted. For each point in the search space the corresponding inter-onset interval and the average energy estimates of the rendition are calculated. After inverting the mapping and applying interpolation, the resulting mapping function, $\boldsymbol{g}$, produces a three-tuple in response to its arguments. The two inputs are the inter-onset time, $\boldsymbol{t}$, of the grace notes and the energy, $\boldsymbol{e}$ of the grace notes: $(\boldsymbol{D},\boldsymbol{L},\boldsymbol{A})=\boldsymbol{g}(\boldsymbol{t},\boldsymbol{e})$

4. EVALUATION

We conducted three types of evaluations. The first was a qualitative evaluation in which the drummer first played short rhythmic patterns and then we listened to the playback performances. This allowed for us to test the limits of the system and assess whether a reasonable amount of information was passing through the system end-to-end. While there were very few detection errors with slow rhythms, we realized that the onset detection could not keep up with very fast passages and started skipping some onsets.

The second evaluation was an automated comparative evaluation. Each test consisted of the drummer performing a short sequence of rudiments and then the system playing it back. Onset analysis was performed on the input and the playback audio and the mean absolute onset errors between corresponding events were recorded for each sequence. Since the performance by the drummer and the system rendition are back to back and not simultaneous, the onset times are calculated with respect to a time reference at the beginning of analysis recording, offset by the actuation latency as explained earlier. The maximum absolute errors were also calculated for each sequence. The drummer performed 15 sequences with the following rudiments: single stroke four, single ratamacue, flam paradiddle, single drag, drag and flamacue, each with 3 repetitions and at varying tempi. The average of the mean absolute onset errors over all pieces was 6.7 msecs. and the max absolute error was 18.3 msecs.

The third evaluation was based on comparing the original and imitation sounds in the spectral domain. The product spectra with the same resolution used for onset analysis were displayed along with the individual spectra of the two sounds. This method serves more as visual tool where the analyst can see the alignment through reinforcement of overlapping energies in the product spectra, as a result, bypassing the onset detection stage for evaluation. Figure 5 shows the visualization for a flam paradiddle routine. The spectra of the input and the output audio can be seen in the top two images. The third image shows the product of the two. The bottom image shows all three of them individually collapsed by summing over frequencies to obtain a single value for each time point. This provides a good form of visualization for evaluating the system's imitation performance.

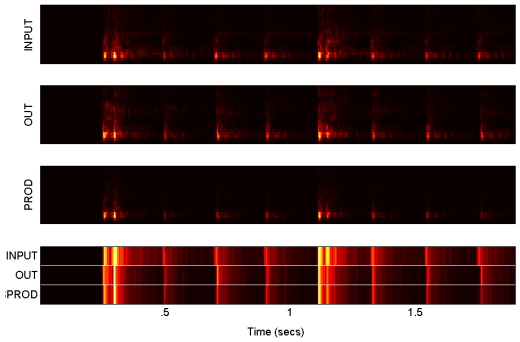

Figure 5. Visual evaluation using spectral displays.

5. CONCLUSIONS AND FUTURE WORK

We have presented a percussion imitation framework that can learn the necessary parameters to replicate a rhythmic pattern with controlled onsets, intensities and specific gestures. The finesse in obtaining parameters through learning which can most overcome the hindrances from using large solenoids, software processing lag and latency from a chain of digital circuitry has been discussed. The learning portion enables the system to be general and rather independent from the specific construction of the actuator setup. This makes the overall imitation system adaptive to changes in position, tuning, spring stiffness, drum stick type and other conditions.

Future work involves an extension to the current system to make it interactive. We are working on utilizing a two-dimensional control input to control high-level performance parameters such as tempo and dynamics. This will allow drummers to record their performances and experiment with performance parameters and styles on an actual drum.

6. REFERENCES

[1] Bello, J.P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M. and Sandler, M. A Tutorial on Onset Detection in Music Signals. IEEE Trans. on Speech and Audio Processing, 13(5):1035–1047, 2005.

[2] Bello, J.P., Duxbury, C., Davies, M. and Sandler, M. On the Use of Phase and Energy for Musical Onset Detection in the Complex Domain. IEEE Signal Processing Letters 11 (6), 553-556, 2004.

[3] Derbinsky, N. and Essl, G. Exploring Reinforcement Learning for Mobile Percussive Collaboration. Intl Conference on New Interfaces for Musical Expression (NIME), Ann Arbor, 2012.

[4] Dixon, S. Onset Detection Revisited. Intl Conference on Digital Audio Effects (DAFx), 2006.

[5] Gillet, O. and Richard, G. Automatic Transcription of Drum Loops. Intl Conference on Acoustics, Speech and Signal Processing (ICASSP), 2004.

[6] Han, Y., Kwon, S., Lee, K. and Lee K. A Musical Performance Evaluation System for Beginner Musician based on Real-time Score Following. NIME Demo, 2013.

[7] Hochenbaum, J. and Kapur, A. Drum Stroke Computing: Multimodal Signal Processing for Drum Stroke Identification and Performance Metrics. NIME, 2012.

[8] Marchini, M. Unsupervised Generation of Percussion Sound Sequences from a Sound Sample. MSc. Thesis, UPF, 2010.

[9] Murphy, J., Kapur, A. and Carnegie, D. Better Drumming Through Calibration: Techniques for Pre-Performance Robotic Percussion Optimization, NIME, 2012.

[10] Uhle, C. and Herre, J. Estimation of Tempo, Micro Time and Time Signature from Percussive Music, DAFx, 2003.