Real-time Rhythmic Pattern Detection from Audio via Template Matching

Ryan Maguire
Dartmouth College
Digital Musics
6242 Hallgarten Hall
Hanover, NH 03755
Ryan.P.Maguire.GR@dartmouth.edu

ABSTRACT

Real-time Rhythmic Pattern Detection from live audio signals is possible using an audio programming environment such as Pure Data, but has been largely neglected. Lower level features such as amplitude, frequency, and tempo are commonly used as data for live processing, however, specific rhythmic pattern detection has not yet attained this same proliferation. We seek to prototype and implement a system for real-time rhythmic pattern detection from a live audio signal via template matching. Our approach is discussed, evaluating several alternative methods of preprocessing to the input audio signal and their effects on obtaining a distance measure between template and input signal. The system is further evaluated for robustness against near matches, and an online rhythmic pattern detector is implemented. Using the results from these tests, an outline for creating a real-time pattern detector in Pure Data is presented and future directions for this project are highlighted.

Keywords

Rhythm, Pattern Recognition, Template Matching, Machine Listening

INTRODUCTION

Our goal in this work is to implement real-time rhythmic pattern detection for use in live musical performance, using only audio as input. The motivation for such work is a desire to control and/or interact via machine listening with an audio/visual computer music system. Ultimately our goal is an implementation of real-time rhythmic pattern detection via audio template matching in an environment such as the Pure Data programming language. Making such an object available for a programming language such as Pure Data will add to the repertory of currently available tools for computer listening and analysis in said environment. As a regular matter of course amplitude, frequency, tempo, and timbral information are extracted from live audio input in Pure Data, and this data is utilized in a myriad of ways to drive interaction between a live sound input and the computer. This data can be used to control anything from robotic accompanists to live signal processing. Much of the information currently accessible in Pure Data is low-level signal information (frequency, amplitude, etc.). Introducing a method for rhythmic pattern detection is a move towards higher-level abstraction, and begins to utilize the kind of perceptually relevant, musical-semantic information that human listeners readily perceive and react to in live music performance and listening.

As a starting point for this project, we have sought to prototype a system which can detect a specified rhythmic pattern when played by the same instrument at roughly the same tempo. We have begun by working with a monophonic signal in a low-noise environment, but would like to eventually scale up to recognize monophonic or polyphonic rhythmic patterns in a variety of sound environments. These are all approachable problems and this work lays a foundation for such future work to build upon.

BACKGROUND

Template Matching has been used as a method in machine listening to determine the beat from an audio signal. It has also been used in rhythm detection from MIDI input signals. Other methods have been used to attempt to detect rhythmic patterns from audio, but none of these are currently available in the public domain for use in the Pure Data programing language to be utilized in live performance situations. A previous approach involves the use of Neural Nets to detect rhythmic patterns.

AUDIO TEMPLATE CREATION

To begin, a variety of approaches were tested by which to create our audio templates against which we will match our input stream. The first step in processing the audio signal which we aimed to use as our template was taking the log frequency spectrum with 1 band per octave and a window size of 4208. For our first test pattern, which was approximately 4 seconds long, this resulted in a 43 by 8 matrix, comprised of 43 time bins and 8 frequency bands.

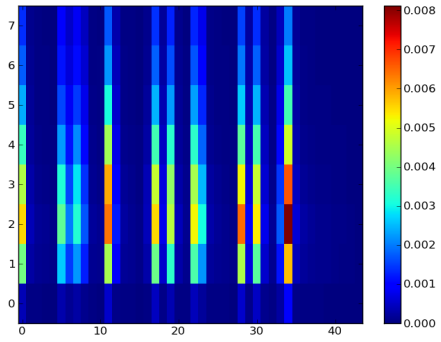


Figure 1. Log frequency 1 band per octave spectrum of snare template

The next step in our processing was taking the first difference along the time axis in each frequency band. The effect of this is highlighting the changes in amplitude in each band. A large increase in amplitude from one time window to the next, such as is present during the attack of a percussion instrument, for example, would result in a high, positive first difference value. A typical signal decay would result in a small negative first difference value.

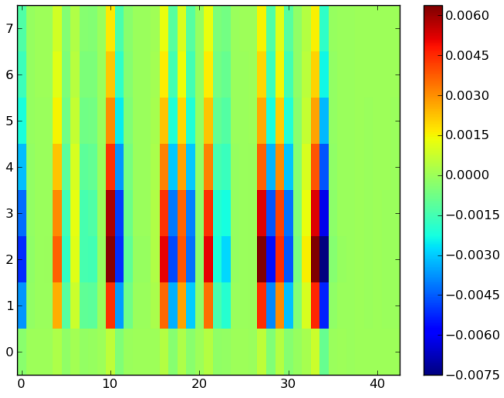
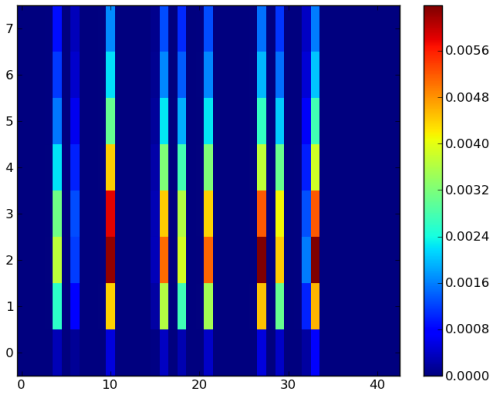


Figure 2. First difference of snare template

The third step in our processing was half-wave rectifying the first differentiated signal. By doing this, setting all negative difference values to zero, we eliminate all of the instrumental decays from our signal, and have a signal which could be used fairly easily for onset detection. The positive increases in amplitude in each frequency band have been distilled and amplified.



In the next phase we attempted to match signals at each of the three steps in our processing method, to see at which step we obtained the most accurate distance measure between different and similar rhythmic patterns.

RHYTMIC PATTERN MATCHING

During this phase of our work, we attempted to determine the best representation for our audio signal from which to calculate the distance. To calculate our distance, we used the Pearson's product-moment correlation coefficient to compute Pearson's distance. Alternatively, the Euclidean normed distance could be used. Here, we use the Pearson's product-moment correlation coefficient to compute the Pearson's distance between two matrices, the first representing our rhythm template, and the second the audio segment of equal length which we are testing as a possible rhythmic match to the first. Then taking the Pearson's distance, which returns a matrix, we take the mean of the values lying along the diagonal, giving us a single number from which to evaluate our "distance". This distance metric ranges from 0 for an identical match to 2 for the most dissimilar matrices possible.

In this test, we compared four different performances of the same rhythmic pattern played on a snare drum to the first of those four performances, which we took as our template. For each of the 4 possible comparisons (A to A, A to B, A to C, and A to D) we computed the Pearson's distance after either taking the Log Frequency Spectrum, the spectrum plus the first difference, or the spectrum and first difference half-wave rectified, as described in the previous section.

It should be mentioned here that in comparing A to A, we should expect a distance of 0, as A is exactly identical to A. B and C should also be fairly close to our template A, and D should be a little further away as it was dragged a little bit behind the original recording and was not a very accurate performance of the desired rhythmic pattern.

Beginning with our raw audio signals, we take the Log Frequency Spectrum of both, with 1 band per octave and a window size of 4208 samples, with a sample rate of 44100. In our test, we observed that the distance from A for A through D respectively was 0.022, 0.027, 0.056, and 0.033. Several things seem wrong with this distance measure. First, the distance between A and A, while it is the lowest of the four, should truly be zero. Second, D should have the largest distance, not C. These are serious problems and warrant further testing with these features and this distance metric. Until such testing is done, using the 1 band per octave log frequency spectrum representation of our signals to calculate the Pearson's distance can only be viewed as inappropriate for our needs.

Next, we took the first difference of each of our signals in the time domain and used this data to test the distance between the template and the four respective audio signals. The distance we measured between A and A through D respectively were: 0.0, 0.167, 0.152, 0.409. These distance measurements fit very well with our expectation from simply listening to the audio files.

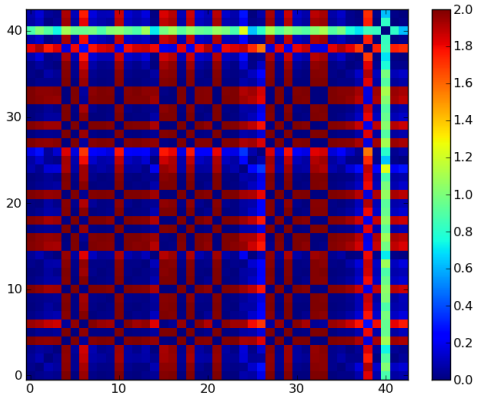


Figure 4. Pearson's distance of template to itself via first difference

The mean distance along the diagonal between A and A should, as it is, be zero, since this is the same signal compared to itself. B and C are both fairly close to A, though not perfectly, exactly the same, so their fairly low scores can be viewed as close matches to the original template. D which was a flawed performance and for our purposes should probably not be viewed as a match, has a higher distance value of 0.409, more than twice the score of B. This is good and we can

imagine setting a threshold between the scores for B and D that would identify those performances closer to B as matches with A and those approaching D as not being matches, as it strayed enough from the original that it could potentially be interpreted as different rhythmic pattern. Overall, using the first difference to calculate the Pearson's distance provides us with distance values that align well with the authors perceptual judgement of the similarity between the different performances.

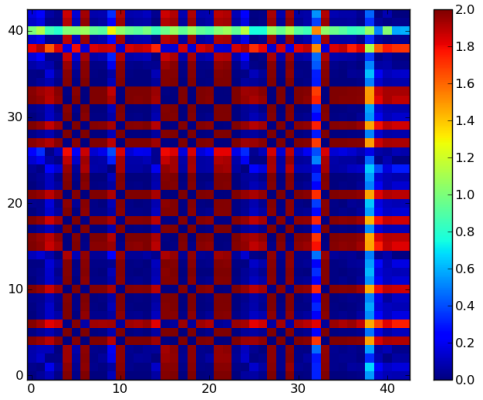


Figure 5. D vs. A. Note the divergence near the end.

Finally, we attempted to half-wave rectify the data from our previous step before calculating the Pearson's distance. This ran into numerical problems, and we had to add a small amount of noise to the data to get rid of zeros and achieve a more accurate result. Even doing this, however, the Pearson's distances we calculated were large, all above 0.7, and inaccurate. The distance to A of A through D respectively was calculated as: 0.737, 0.747, 0.710, 0.863. D was correctly identified as the most distant of the four, however C was identified as the closest, and A was not given a distance of zero to itself.

Given these results, we have decided to use the first difference representation of our audio signals to calculate the Pearson's product-moment correlation and Pearson's distance, for use in our real-time audio rhythmic pattern detection system.

NEAR-MATCHES

The goal of our rhythm detection system is to identify rhythmic patterns played on a specific instrument, at a specific tempo. Thus, the question of how our system responds to variations in instrumental timbre, tempo, and pattern variations is an important one. Ideally, our system will reject the same pattern played on a different instrument, will reject the same pattern played on the same instrument but at a different tempo, and will reject alternative patterns that do not match our specified template, despite similarities in tempo and identical timbres.

In order to test our system, we first compared the recording of our snare drum test template A to recordings of the same pattern played on a ride cymbal and on a tom-tom. In each case the distance was calculated to be approximately 1.15. You can see that despite the patterns being identical, the differences in timbre accounted for the large distance between our template and these other performances.

The next task was to see how the system would perform when presented with faster and slower performances of the original pattern on the same instrument. In both cases, a large distance was returned, greater than 0.7, demonstrating that the system will not match the same pattern played at a different tempo. This is a desirable quality for our purposes, but one can imagine a situation in which a live performer plays the specified pattern plus or minus a few beats per minute despite which it might be desirable to identify the performance as a match. In order to make the system robust to such deviations without sacrificing it's tempo stringency, we can resample the input audio selection to slightly stretch and compress the new pattern so that, in the case that a performance of a pattern is accurate, but deviates only slightly in tempo, it will still be identified as a match.

Finally, we tested our system using a similar but distinct rhythmic pattern, to see how the system would respond to a different pattern, despite similarities in tempo and timbre. As hoped, the distance for our alternative pattern was 0.565, significantly higher than the distance scores for accurate performances of the same rhythm.

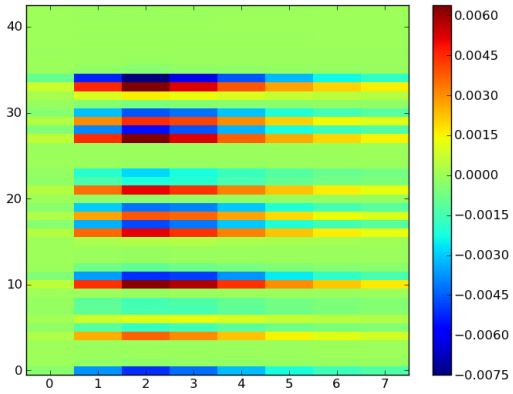


Figure 6. Snare Template First Difference.

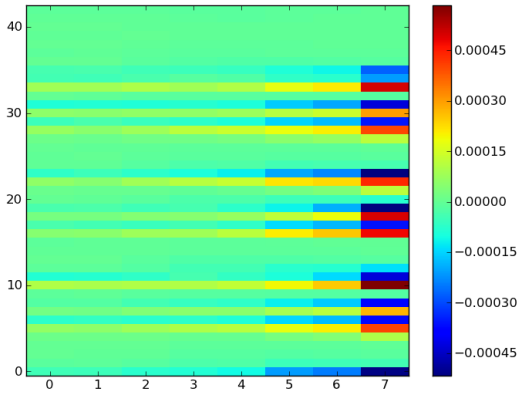


Figure 7. Ride Cymbal First Difference. Note high frequency content.

ONLINE PATTERN DETECTION

As the next step in our march towards real-time rhythmic pattern detection, we implemented online pattern detection detecting our 4 second long rhythmic template from a longer sound file containing other rhythms at the beginning, the pattern emerging in the middle, and then other rhythms at the end.

To implement this step, we took sub-sections of the longer audio stream file equal in length to the shorter template file, starting at the first window and proceeding through the file one step at a time, taking a 43 window long segment, measuring the distance between the segment and the template, and then incrementing ahead one window and taking a new 43 window

long segment, until reaching the end of the file. Thus, we took segment [0:43], then [1:44], then [2:45], and so on until teaching the last 43 window long segment we could take from the stream file. Calculating the distance between each of these segments and our template was fast and efficient.

In our experiment, the correct rhythm emerged at approximately window [8:51] in the longer stream audio file. Our distance metric correctly scored this point as the nearest distance match in the stream, with a value of 0.275.

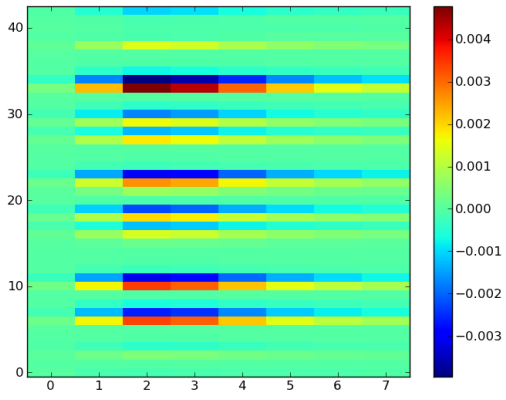


Figure 8. Stream segment [8:51]. Compare to Figure 6.

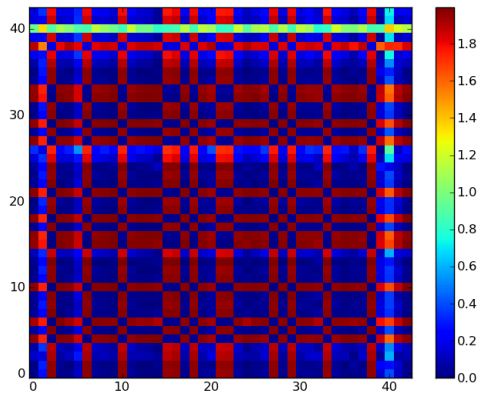


Figure 9. Stream segment [8:51] vs. Snare Template. Note near perfect matching along the diagonal in the middle, but divergence near the beginning and end.

Most of the other values calculated where above 0.8 and this was a clear match. Two other points had relatively low values, 0.39 and 0.46, but there is enough space between these numbers and our actual match that one could set an appropriate threshold , say around 0.3, and it should perform fairly well. This warrants further testing to determine the best threshold value. One issue that arises here is how tightly bound your rhythmic pattern should be within its respective template. In our case, the 43 window long matrix contains a rhythmic pattern that is somewhat shorter than the template as a whole. Thus, in the context of an auditory stream, rhythmic events occurring just before and just after the rhythmic pattern "leak" into our new audio segment which we are trying to match against the template. This explains the distance value of 0.275, despite the actual rhythm being performed being a nearly identical match to the template rhythm taken on its own. Needless to say, the tighter the template can be made to fit around the rhythm, the closer the distance measure will be when a close match emerges from an audio stream.

REAL-TIME IMPLEMENTATION

As the final step, we would like to create real-time implementation of this system in Pure Data. Our modeling and testing have been done in Python, utilizing the Bregman Music and Audio Toolkit for feature extraction. Creating a version in Pure Data will involve much the same processes. We will use 8 sharp band pass filters to create 1 octave frequency bands and then take the first difference between subsequent analysis windows. Using this data we will have a template creation mode and a matching mode. After creating a template. We will search, frame by frame, in real time for matches to our template which achieve a distance below a given threshold. This will be done in the same manner as our online matching. Finally we will need to implement the distance function in Pure Data, and, when a distance value arises which is below our threshold a "bang" will be output, signifying that we have found a match to our template. This can be used for anything with in Pure Data, from call and response, to triggering changes in spectral processing, etc. Future work will involve making this process more robust to background signal noise, as might be present in polyphonic signal mixtures.

CONCLUSION

We have sought to prototype a system for the real-time detection of rhythmic patterns from audio signals using template matching. We have implemented a successful online rhythmic pattern detector in Python and have an outline for doing so in Pure Data with which to move forward. The system is capable of detecting near matches to a specified rhythmic pattern played with a specified musical instrument / timbre at a given tempo. The system can differentiate between accurate and inaccurate renditions of the rhythmic pattern, can differentiate between the same pattern played with different timbres, and the same pattern played at different tempos. It is capable of detecting a rhythmic pattern from within a longer auditory stream containing rhythmic onsets not related to the pattern, from which the pattern then emerges.

ACKNOWLEDGEMENTS

Special Thanks to Michael Casey for his guidance in realizing this project.

REFERENCES

Blostein, D., & Haken, L. (1990, June). Template matching for rhythmic analysis of music keyboard input. In *Pattern Recognition, 1990. Proceedings., 10th International Conference on* (Vol. 1, pp. 767-770). IEEE.

Figueiró, C. (2009) Detection of rhythmic patterns in real time with Pd. *3rd pd International Convention.*

Klapuri, A., and Paulus, J.. (2002, October) Measuring the Similarity of Rhythmic Patterns. In *Proc. ISMIR* (Vol. 2).

Nikolaidis, R., & Weinberg, G. (2010, September). Playing with the masters: A model for improvisatory musical interaction between robots and humans. In*RO-MAN, 2010 IEEE* (pp. 712-717). IEEE.

Rowe, R. (1992) *Interactive Music Systems: machine listening and composing.* MIT Press.

Scheirer, E.D. (1998) Tempo and beat analysis of acoustic musical signals. *The Journal of the Acoustical Society of America, 103,* 588.

Toussaint, G. T. (2004, October). A comparison of rhythmic similarity measures. In Proc. 5th International Conference on Music Information Retrieval(pp. 242-245).