Towards Note-Level Prediction for Networked Music Performance

Reid Oda
Department of Computer Science
Princeton University
roda@princeton.edu

Adam Finkelstein
Department of Computer Science
Princeton University
af@cs.princeton.edu

Rebecca Fiebrink
Department of Computer Science (also Music)
Princeton University
fiebrink@princeton.edu

ABSTRACT

The Internet allows musicians and other artists to collaborate remotely. However, network latency presents a fundamental challenge for remote collaborators who need to coordinate and respond to each other's performance in real time. In this paper, we investigate the viability of predicting percussion hits *before* they have occurred, so that information about the predicted drum hit can be sent over a network, and the sound can be synthesized at a receiver's location at approximately the *same moment* the hit occurs at the sender's location. Such a system would allow two percussionists to play in perfect synchrony despite the delays caused by computer networks. To investigate the feasibility of such an approach, we record vibraphone mallet strikes with a high-speed camera and track the mallet head position. We show that 30 ms before the strike occurs, it is possible to predict strike time and velocity with acceptable accuracy. Our method fits a second-order polynomial to the data to produce a strike-time prediction that is within 10 ms of the actual strike, and a velocity estimate that will enable the sound pressure level of the synthesized strike to be accurate within 3 dB.

Keywords

Networked performance, prediction, computer vision

1. INTRODUCTION

The Internet has allowed people who are separated by distance to more easily collaborate with one another. Email, cloud storage, and video chat have all increased the speed with which we can share and exchange information. However, certain forms of real-time collaboration have still not gained widespread use. Among these are playing live music with remotely located musicians.

There are a number of reasons why one might want to play with another musician over the Internet. First, a musician might want to work with other artists who have the same ideals, a complementary aesthetic sense, or valuable skills. Second, a group of musicians might want to practice pieces for performance while traveling. Third, they might want to perform pieces to an audience without having to travel to a common location. Finally, limits on travel might make it difficult for an artist to leave or enter a country.

NIME'13, May 27 – 30, 2013, KAIST, Daejeon, Korea.

Playing live with remote collaborators is difficult because of network latency [3, 6]. Therefore, researchers have devised a variety of approaches for dealing with latency in musical performance. For example, one strategy is to create a low-latency, streamlined client designed to deliver audio data from one computer to another as quickly as possible [5]. Another approach that addresses latency involves having the computer learn typical music patterns, predict which pattern a musician is playing, and have that same pattern synthesized on a remote machine. With this approach, patterns on local and remote machines can be synchronized [11]. However, the approach is limited by the range of patterns that can be learned by an algorithm, and it offers responsiveness only at the granularity of an entire sequence of notes.

Inspired by this work, we propose to take the approach one step further. Instead of predicting patterns, we investigate the viability of predicting individual notes. If we can predict the onset time and velocity of the next note a musician will play, we can send the time and velocity prediction over the network *before* the note is played, and synthesize the note sound at the remote machine at approximately the *same moment* it is played (Figure 1). We hypothesize that percussion instruments are a viable class of instruments for this type of prediction. In this paper, we analyze the travel of a vibraphone mallet head just before striking a surface, and we predict the timing and velocity of the oncoming strike. We study two different factors that contribute to the accuracy of the prediction: the amount of desired prediction time (in ms), and the sampling frequency of the sensing device (in Hz).

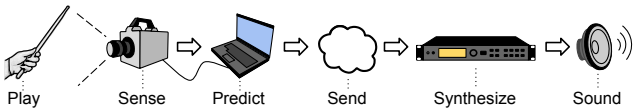

Figure 1: The proposed method: Predict a note event before it is played. Send it over a network to the receiver and synthesize it at the same moment it is played by the sender.

2. BACKGROUND

The effects of network lag on musical performance have been studied by Chafe, Cáceres, and Gurevich [6] as well as Bartlette and Bocko [3]. They found that latencies of 60–100 ms can significantly impede pairs of musicians' ability to play with one another. High latencies cause *tempo drag* as each musician slows down to remain synchronized with the other.

The current state of the art in low-latency Internet music performance clients is JackTrip [5]. JackTrip uses the UDP protocol for packet delivery, so it does not have the congestion control and packet verification overhead of TCP. Packets that arrive late are dropped. Other optimizations

in this client reduce network and host latency as far as possible. However, Internet path—and therefore the latency resulting from the path—is largely out of the control of the user. Overlay networks offer one of the few approaches to reducing travel time within the network [2], they come with a cost: slow paths may be avoided, but faster paths may become slightly slower, due to processing overhead.

Recently, Sarkar and Vercoe proposed a system that uses pattern recognition to eliminate network latency [11]. In their approach, two remote tabla players perform together. Each end of the connection employs a software system that has learned a collection of canonical tabla beats. The system classifies the pattern that the local musician is playing and sends this information to the remote host, which then synthesizes a version of the pattern to the remote player. Because tabla patterns are highly standardized, it is likely that the receiver hears a close approximation to what is actually being performed. The spirit of the players' intention is communicated and the artists experience playing with one another in synchrony. Our method builds on this idea of prediction. However, instead of predicting with *pattern* granularity, we propose to predict each individual *note*.

While not all notes can be easily predicted, the overt gestures of a percussionist provide fair warning as to when a note will be played. In a study of drumstick motion, Dahl shows that the drumstick travels a repetitive, somewhat regular path [7]. Onset velocity correlates tightly with the height to which the drumstick is raised before the strike. In our paper, we investigate the viability of modeling such percussion strokes and predicting the timing and velocity of each strike, with the intention of sending these predictions over the Internet before the strike occurs, and synthesizing the strike sound at the receiver's end.

3. RECORDING SETUP

We study the vibraphone mallet because the head can be easily tracked using computer vision. To capture mallet motion, we used a Photron Fastcam SA3 grayscale camera running at 500 frames per second. The resolution is 512×512 pixels. The wooden striking surface was oriented such that mallet contact with the surface could be easily identified by examining mallet height, as seen in Figure 2.

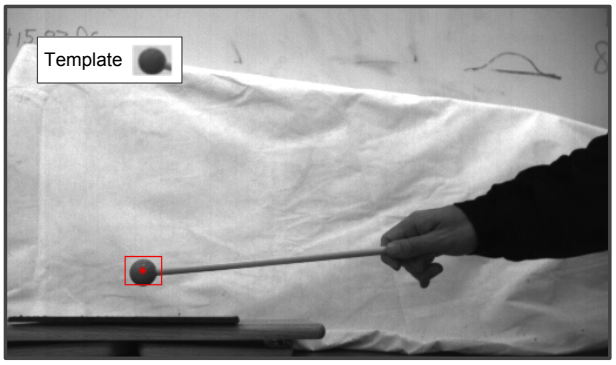

Figure 2: A single image from the high speed camera. Inset is the template used to track the mallet head. Best match is indicated in red.

The mallet head was tracked using a template-matching algorithm, which takes as input a *template* image of a mallet head, and a *source* image within which to locate the mallet head [12]. The algorithm computes the normalized cross-correlation distance between the template and each position on the source image.

We recorded two patterns, a 4/4 quarter note pattern at 120 bpm with an accent on the first beat of each measure (consisting of 24 notes), and a syncopated rhythm at 180 bpm (consisting of 34 notes). The syncopated rhythm included quarter and eighth notes. This yielded 11329 images. Examples of the tracking algorithm's raw, unsmoothed height output appear in Figure 3.

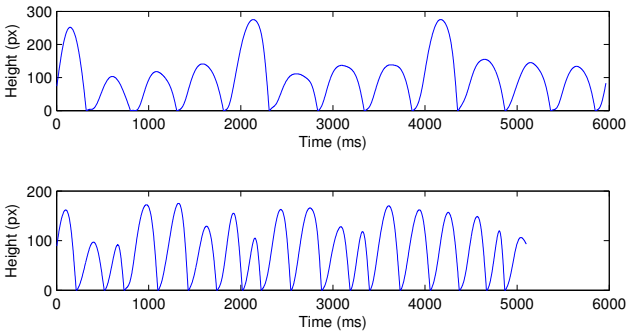

Figure 3: Mallet height over time, as output by our computer vision tracking algorithm, for quarter notes at 120 bpm (top) and syncopated notes at 180 bpm (bottom).

4. EVALUATION

We wish to test the hypothesis that the timing and velocity of a mallet strike can be predicted accurately and early enough to be useful in reducing network lag.

We divided the recording into 58 individual strikes, called *cycles* in this paper. We chose a simple prediction algorithm, described shortly, that predicts strike time by fitting a polynomial to a cycle during the descent. Figure 4 illustrates one such single cycle in blue: the dashed red line is the predicted mallet path, and yellow circles represent the data fed to the polynomial-fitting algorithm. Here, the algorithm predicts the strike time to be 8 ms before the strike actually occurred.

We conducted a sequence of tests evaluating the accuracy of predicted *strike time* and *strike velocity*, as well as the number of missed notes, as we varied (1) the *anticipation time* (i.e., how far in advance of the strike to commit to a prediction, and send the prediction over the network to a receiver), and (2) the *sample rate* of the camera data. Tests employed anticipation times of 10, 30, 40, and 50 ms. The sampling rates tested were 60, 100, 250, and 500 Hz. We chose these sampling rates because they correspond roughly to different cost levels of commercial cameras. The dataset for each sample rate was constructed by downsampling from the original 500 Hz dataset.

Our prediction algorithm works as follows: For each new mallet height sample, test to see whether the peak of the cycle has passed and the mallet is descending. If so, fit a quadratic using the samples from the previous 30 ms. The predicted strike time is simply the largest root (zero-crossing) of this polynomial. Next, we determine if there is

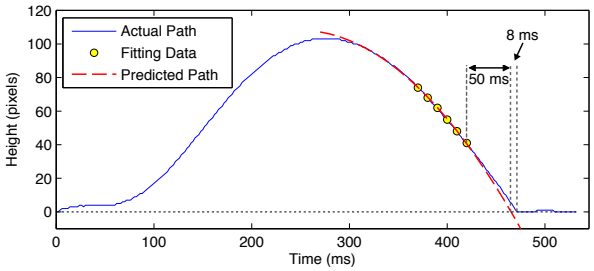

Figure 4: Mallet cycle fitted with path prediction. The final prediction is computed 50 ms before the predicted strike, based on the data (yellow, 100 Hz). Prediction error is 8 ms.

sufficient time between the current time and the predicted strike time to take another sample (i.e., wait for another frame of video to arrive and be processed), to produce an improved prediction. If so, repeat the above process. If not, we "send" the event over the network to the receiver. (In our prototype, the system does not actually send the event; the prediction is simply recorded for later analysis.)

In our experiments we found empirically that quadratic polynomials perform better in this context than either linear or cubic predictors. To produce a ground-truth estimate of the velocity at the time of contact, we also fit a quadratic to samples from the 40 ms preceding the strike. We estimate the ground-truth velocity using the first derivative of the polynomial at the estimated strike time.

Alternate Prediction Strategy (Data-Driven). We also experimented with a different prediction strategy that compares the partially-completed cycle to every previously recorded note. The previous cycles are each scaled to match peak height, and slightly stretched in time to match partial width, and then the best-matching shape (by L_2-norm) is used to predict the strike time. This approach did not perform any better than the basic quadratic fitting approach described above, so for the remainder of the paper we return to that approach. We believe that this data-driven approach might perform better in a real performance setting where thousands of notes are captured, and every new note provides training data. Moreover, in such a setting, the basic approach might be less robust due to noise.

Desired Accuracy. We would like to gain at least 30 ms of *anticipation time* (time between the sending of the prediction and the actual strike). This is equal to the average network latency between New York City and Chicago, or Denver and San Diego [1].

We selected a target window for the *predicted* strike such that it is within 20 ms of the *actual* strike, because perceptual studies have shown that humans cannot perceive ordering in sounds that occur within 20 ms of each other [4]. The studies show that while a listener may be able to perceive the existence of two sound onsets, he or she cannot tell which one occurred first.

Likewise, we aim for the sound pressure level (SPL) of the synthesized strike to be within 3 dB of the actual strike. This amount of error is perceivable, but loudness is not as vital a characteristic as timing. We use a simplified model for estimating the discrepancy between loudness of the synthesized and actual strikes, based on the Hertz Contact Model [10], and on empirical testing of how commercial synthesizers respond to MIDI velocity changes [8]. Specifically, as in [8], we assume peak RMS varies with the square of impact velocity. We can then express a perceptually-relevant, if overly simple, estimate of the error in decibels:

$$L_{error} = 20 \log_{10} \left[\left(\frac{V_p}{V_m} \right)^2 \right] \quad (1)$$

V_p is the predicted velocity, and V_m is the measured velocity. L_{error} is the difference in SPL between the predicted and actual note, in decibels.

Missed Notes. Occasionally, the prediction algorithm determines that there is enough time to collect another sample before making its final prediction, only to find that when the next prediction is made, the desired anticipation time cannot be satisfied. The cutoff time has passed. For our evaluation, we record these as *missed notes*.

5. RESULTS AND DISCUSSION

This section evaluates the effectiveness of this approach under varying conditions.

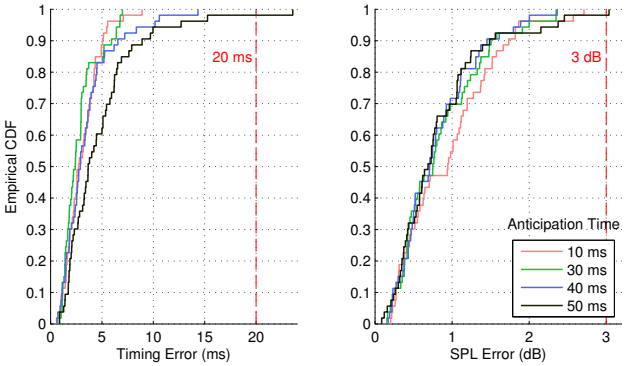

Figure 5: Varying Anticipation Time. Empirical cumulative distribution function of timing and sound pressure level errors while varying anticipation time. Sampling: 500 Hz.

Varying Anticipation Time. Figure 5 shows the empirical cumulative distribution of the errors in strike time prediction for various levels of anticipation times, at a sampling rate of 500 Hz. For 10, 30, and 40 ms, 100% of the measured errors fall within the desired 20 ms. Additionally, over 80% of the errors for these times are under 5 ms. The SPL errors also fall within the desired range of error. For anticipation times of 10, 30, and 40 ms, 100% of the error is less than the target of 3 dB. Additionally, over 90% of the errors for these times are under 2 dB.

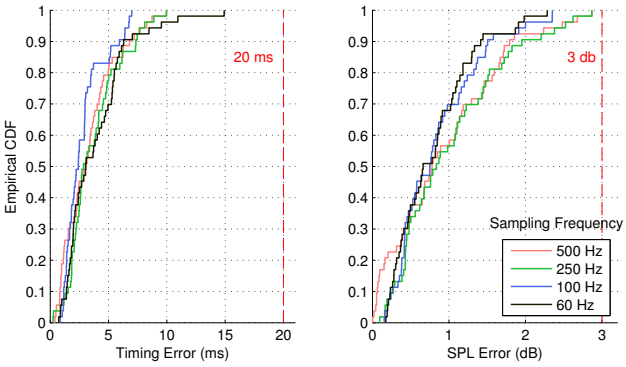

Figure 6: Varying Camera Sampling Rate. Empirical cumulative distribution function of timing and SPL errors while varying sampling frequency. Anticipation time: 30 ms.

Varying Camera Sampling Rate. Figure 6 shows the effect of sampling rate on prediction accuracy using an anticipation time of 30 ms. All frequencies performed roughly equally well. Occasionally the lower frequencies exhibit lower error than the higher frequencies. This is because troublesome cycles that are difficult to predict at the higher frequency ranges become impossible to predict at lower frequencies. They are recorded as missed notes, and are not included in the CDF.

Missed Notes. Figure 7 shows the result of varying the sampling rate and the anticipation time as they impact number of missed notes. At a sampling rate of 500 Hz, there are no missed notes at any anticipation time. Lower sampling rates increase the percentage of missed notes, as do higher anticipation times. We see a sweet spot at 250 Hz (shown in green) and 30 ms where the error rate is 0.97%. This means that for roughly every 100 notes played, 1 note must either be dropped or transmitted knowing it is possible that it will arrive late enough to be perceived as late.

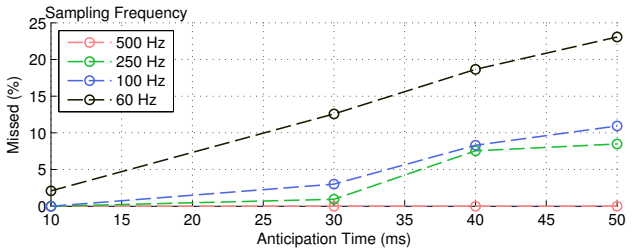

Figure 7: Missed notes. Fraction of notes missed for various required anticipation times and camera sampling rates.

Limits, Network Latency and Timing. Faster tempos, shorter note lengths, and lower sampling rates all have a similar effect of reducing the number of samples in a cycle. Our experiments with the number of sample points used in curve fitting suggested that at least 5 points were necessary to produce an accurate prediction. With these limitations in mind, at a tempo of 120 bpm, using a sampling rate of 250 Hz, the fastest note that can be played is a 32nd note. Additionally, the time from the vertical peak of the swing to the strike time must be less than the sum of the anticipation time and the prediction computation time. If it is longer, the algorithm can't possibly predict a strike. Furthermore, if an artist stops a downward swing after the predicted note has been sent to the receiver, then a false note will sound.

A real-world system will have to contend with changes in network latency. There is very little literature which studies changes in latency over time, so we conducted an informal pilot study, pinging 7 different popular websites over a period of 10 minutes. We found that in general 95% of ping times were within 5 milliseconds of each other. However, some ping times reached as high as 10 times the baseline. We intend to investigate latency behavior in a future study. Finally, our system will have to synchronize timing between sender and receiver. To address this, messages can be time stamped and independent clocks on each end of the connection can be synchronized via GPS time synchronization [9].

Computation Time. In order to further determine the plausibility of our proposed approach, we analyzed the computation time taken by the various steps of the prediction process. The computationally significant operations that must be performed on each image are (1) locating the mallet head via template matching and (2) making a new path prediction via least squares regression.

The complexity of the template matching algorithm is $O(mn)$ where m and n are the number of pixels in the template and source image, respectively. Using the 512×512 pixel source image, the operation takes roughly 112 ms (in Matlab, using a 2.3 GHz Core i5 Duo processor), which is too long. Therefore, we constrained the search area to a smaller region around the last known location of the mallet head. Searching a 64×64 area takes roughly 1.4 ms. This approach works extremely well for higher sampling rates where the mallet head moves no more than 8 pixels per sample. However, at 60 Hz, the mallet head in our dataset moved a maximum of 75 pixels. The size of the template image is 24×31 pixels. This implies that at 60 Hz, we need a search area of at least 160×160 pixels, which takes roughly 9 ms to search.

The least-squares algorithm used to predict the path algorithm is quite fast, taking roughly 1 ms. We varied the number of example points between 5 and 100 and found the fitting time to be empirically constant with regards to the number of input points. To this we add a 7 ms estimate of latency for a high performance motion-capture camera.

In summary, for each image captured, our Matlab-based motion tracking and prediction system requires 17 ms of computation time. We have included this estimate in our prediction model. This means that in figures 7, 5, and 6, when the anticipation time of 10, 30, 40, or 50 ms is listed, the prediction was made 27, 47, 57 and 67 ms before the predicted strike time, to allow 17 ms before "sending" the prediction across the network.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we explore the viability of predicting the timing and velocity of percussion strikes with the intention of reducing latency in networked musical performance. We show that a simple algorithm, second-degree polynomial fitting, is sufficient to predict notes 30 ms before they happen, and that the timing of the predictions are nearly indistinguishable from the original strikes. We also show that predicted and actual sound pressure levels are within 3 dB, and that the frequency of missed notes is below 1%. Because of the promising nature of these results, our next step will be to implement a real-time version of this prediction system and test the results with human performers. We are also eager to explore a number of other avenues for future work, including adapting to the style of a performer in real-time, incorporating musically-aware analysis by leveraging tempo tracking and/or score following, devising a robust treatment of false positive strikes, and tracking multiple mallets and/or other instruments.

7. ACKNOWLEDGMENTS

This work was supported by the Project X fund at Princeton University.

8. REFERENCES

[1] Global IP Network Latency, Jan. 2013. http://ipnetwork.bgtmo.ip.att.net/pws/network_delay.html.

[2] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proc. ACM Symposium on Operating Systems Principles*, 2001.

[3] C. Bartlette and M. Bocko. Effect of Network Latency on Interactive Musical Performance. *Music Perception*, 24(1), 2006.

[4] D. E. Broadbent and P. Ladefoged. Auditory perception of temporal order. *The Journal of the Acoustical Society of America*, 31(11), 1959.

[5] J. Cáceres and C. Chafe. JackTrip: Under the hood of an engine for network audio. *Journal of New Music Research*, 39(3), 2010.

[6] C. Chafe, J. Cáceres, and M. Gurevich. Effect of temporal separation on synchronization in rhythmic performance. *Perception*, 39(7), 2010.

[7] S. Dahl. Striking movements: A survey of motion analysis of percussionists. *Acoustical Science and Technology*, 32(5), 2011.

[8] R. Dannenberg. The Interpretation of MIDI Velocity. In *Proc. ICMC*, 2006.

[9] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Operating Systems Review*, 36(SI):147–163, 2002.

[10] H. Hertz. On the contact of elastic solids. *J. reine angew. Math*, 92(156-171), 1881.

[11] M. Sarkar and B. Vercoe. Recognition and prediction in a network music performance system for Indian percussion. *Proc. NIME*, 2007.

[12] R. Szeliski. *Computer vision: Algorithms and applications*. Springer, 2010.