Real-Time Dance Generation to Music for a Legged Robot

Conference Paper · October 2018

DOI: 10.1109/IROS.2018.8593983

4 authors:

Thomas Bi
ETH Zurich
1 PUBLICATION **1** CITATION
SEE PROFILE

Péter Fankhauser
ANYbotics
45 PUBLICATIONS **992** CITATIONS
SEE PROFILE

Dario Bellicoso
Boston Dynamics
33 PUBLICATIONS **763** CITATIONS
SEE PROFILE

Marco Hutter
ETH Zurich
142 PUBLICATIONS **2,798** CITATIONS
SEE PROFILE

Real-Time Dance Generation to Music for a Legged Robot

Thomas Bi, Péter Fankhauser, Dario Bellicoso, Marco Hutter

***Abstract*— The development of robots that can dance has received considerable attention. However, they are often either limited to a pre-defined set of movements and music or demonstrate little variance when reacting to external stimuli, such as microphone or camera input. In this paper, we contribute with a novel approach allowing a legged robot to listen to live music while dancing in synchronization with the music in a diverse fashion. This is achieved by extracting the beat from an onboard microphone in real-time, and subsequently creating a dance choreography by picking from a user-generated dance motion library at every new beat. Dance motions include various stepping and base motions. The process of picking from the library is defined by a probabilistic model, namely a Markov chain, that depends on the previously picked dance motion and the current music tempo. Finally, delays are determined online by time-shifting a measured signal and a reference signal, and minimizing the least squares error with the time-shift as parameter. Delays are then compensated for by using a combined feedforward and feedback delay controller which shifts the robot whole-body controller reference input in time. Results from experiments on a quadrupedal robot demonstrate the fast convergence and synchrony to the perceived music.**

I. Introduction

Traditionally, robots are designed to fulfill tasks with little to no social interaction, e.g., in fields such as manufacturing, logistics, or inspection. However, the emergence of humanoid and animal-like robots has lead to many instances of robots interacting with humans, such as robots that can hold conversations and understand human orders from speech [1], or even robotic toys that react to their users' emotions [2]. Not surprisingly then, dancing robots have gone from being a small gimmick to an actual field of research [3]. Dance, as a performance art form, has been part of human social interaction for multiple millennia. It helps us express emotion, communicate feelings and is often used as a form of entertainment. Consequently, this form of interaction has often been attempted to be imitated by robots.

The field of dancing robots has seen various contributions from numerous sources. SONY presented a humanoid robot called QRIO [4], that can dance with multiple units in a highly coordinated fashion, closely imitating a human dance performance. Nakaoka et al. [5] used a motion capture system to teach their HRP-2 robot a traditional Japanese folk dance. While these robots demonstrate impressive dancing capabilities, the sequence and timing of all their movements are precisely pre-programmed to fit a set of chosen musical pieces. As a result, they are unable to react to external stimuli such as musical or visual cues.

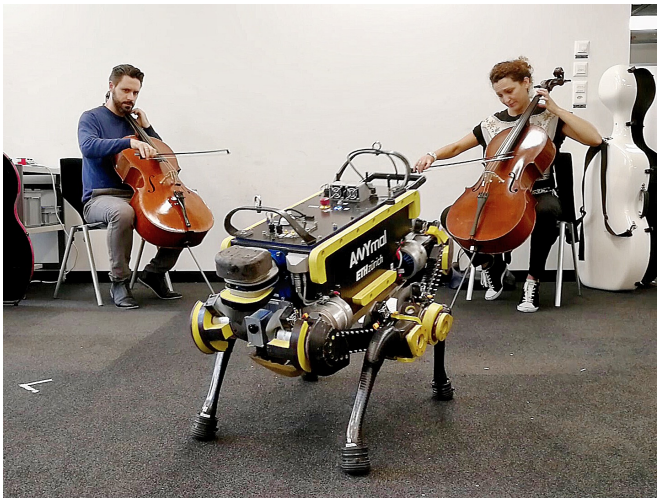

Fig. 1. The legged robot ANYmal [14] listens to live music with an onboard microphone, extracts the current beat, and generates varying dance motions with its legs and base. Online delay compensation ensures synchronous motions to the perceived music. A video is available at `https://youtu.be/kHBLaw5nfzk` (Musicians: Mr. & Mrs. Cello).

On the other hand, several robots were designed to dance in reaction to both auditory and visual stimuli. Examples include robots that drum along a human performer [6][7], a robot that steps in time with external music [8], and a quadrocopter performing along to a set of pre-tracked music [9]. So far, most robots that dance along with external stimuli are limited in their dancing capabilities. Often, they are limited to a single, periodic dance motion.

System hard- and software delays present another difficulty in robotic dance, as the robot output will usually lag behind its controller reference input. To prevent such asynchrony, [10] and [11] compared the point in time a robot has finished executing a dance motion and measured the timing difference with the beat. This timing difference was then used to adjust the velocity of the next motion. The authors of [12], [9] and [13] fed oscillatory motions to their dancing robot. The phase shift between the robot and desired trajectory is determined, and the phase of the robot control reference input is shifted accordingly.

Our goal with this work is to bridge the gap between the ability to react to external stimuli (in this case music), and the execution of dance motions that are both synchronized to the beat, and visually pleasing and varied. We focus on legged robots, as they can move in various ways with base and leg motions and strongly resemble the natural morphology of

The authors are with the Robotics Systems Lab (RSL), ETH Zurich, Switzerland, (thomas@novobit.ch)

This work has been conducted as part of ANYmal Research, a community to advance legged robotics and was supported by the National Centre of Competence in Research Robotics.

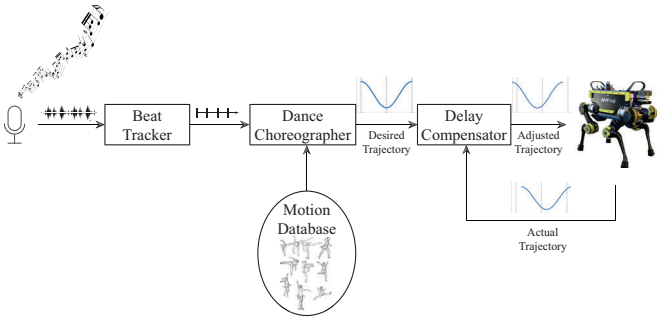

Fig. 2. An overview of the three modules, the beat tracker, the dance choreographer and the delay compensator, and their interactions.

humans and animals. The complex dynamics of legged robots aggravate precise timing of robot motion; contact switches of the end effectors are challenging to control as the exact timing of the switch is hard to predict. Additionally, delays in robot motion can vary greatly depending on the motion to be executed; slow, simple motions can be tracked much more precisely than highly dynamic maneuvers. As such, a robust synchronization algorithm ensuring continuous synchronization between the robot and the music is imperative for a successful legged robot dance performance. As a result, building on an existing real-time beat tracker, we contribute a probabilistic model to create a dance choreography based on dance motion primitives. Additionally, we present a robot motion delay estimator and controller which can synchronize arbitrary base and leg motions (including contact switches) to the extracted musical beat. An overview of our approach is shown in Fig. 2.

This paper is structured as follows: First, the real-time beat tracker in Section II, second the dance choreographer in Section III, and lastly the delay compensator in Section IV are described. In Section V, the effectiveness of our method will be evaluated on the quadrupedal robot ANYmal [14] (Fig. 1). A conclusion with potential improvements for the future finishes this work in Section VI.

II. Real-Time Beat Tracker

The beat is a salient periodicity in a musical signal and provides the fundamental measure of time and temporal structure of a musical piece. Not only does this structure provide the beat, but it also builds the foundations for motion timing in dance, making it a necessity for dance. Hence, our goal is to extract the beat from a captured audio signal in real-time.

Our real-time beat tracker is based on the method proposed in [15], where we used the full implementation provided by the *madmom* [16] library. As is common with most beat trackers, the madmom beat tracker processes the audio input framewise using 10 ms long audio frames, which results in an update rate of 100 Hz. This is short enough to not violate the real-time constraint, but still long enough to have sufficient data for meaningful frequency analysis.

The beat-tracker can be summarized in 4 parts:

1) *Frequency Domain:* Frequency information is extracted from the audio frame using the Fast Fourier Transform (FFT).
2) *Onset Detection Function:* Using the frequency spectrum as an input, the musical onset likelihoods for different frequency bands are calculated.
3) *Beat Likelihood Detection:* From the onset detection function output, a Recurrent Neural Network (RNN) infers the likelihood of the current frame being a beat.
4) *Beat Picking:* Lastly a Dynamic Bayesian Network infers both the tempo and the phase of a beat sequence based on the probabilities given by the RNN.

This method outputs a sequence of beat times

$$\tilde{b_1}, \tilde{b}_2, \ldots \tilde{b}_n \,. \tag{1}$$

which is updated in real-time as soon as a beat is detected in the current audio frame. The time it takes from changing/starting a new song to the beat tracker correctly identifying the beat and tempo, varies significantly for different songs and rhythms. Experiences from our experiments suggest that the convergence time can range anywhere from 1 to 8 s, much depending on how dominant a beat is in a song. For a more detailed description of the beat tracker, refer to [15].

We extended the capabilities of the madmom beat-tracker to be able to extrapolate one beat in the future. The current beat duration (the inverse of the tempo) is estimated using the two most recent beat times: $\tilde{d}_n = \tilde{b}_n - \tilde{b}_{n-1}$. Further, this beat duration is filtered using a median filter of range three: $d_n = \text{med}(\tilde{d}_n, \tilde{d}_{n-1}, \tilde{d}_{n-2})$. Using this duration, one beat in the future is extrapolated as

$$b_{n+1} = \tilde{b}_n + d_n \,, \tag{2}$$

leaving us with the sequence of filtered and extrapolated beats

$$b_2, b_3, \ldots b_{n+1} \,. \tag{3}$$

It is essential to our system to have a beat time at a point in the future, as we are expecting the robot to dance proactively, rather than reactively.

III. Dance Motion Design and Choreography

After extracting the beat from the microphone input, a fitting dance motion has to be chosen by the dance choreographer to create dance through a succession of dance motions. This dance choreographer picks from a dance motion library, which is discussed first. Then, the probabilistic model on which our dance choreographer is based is introduced.

A. Dance Motion Primitives

We chose to build our dance motion library by user-defining each dance motion separately. In contrast to other methods, e.g., motion capturing, this approach allows for greater artistic freedom which is only limited by the robotic capabilities. Furthermore, ensuring stability throughout the motion is facilitated.

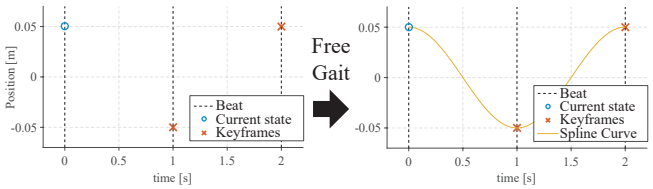

Fig. 3. Dance motions are defined by keyframe poses, which are fitted using quintic polynomial splines through the Free Gait interface. The starting keyframe is given by the current robot state.

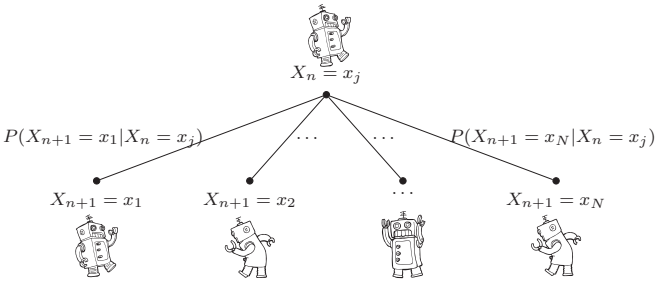

Fig. 4. The Markov chain as model for choosing an appropriate dance motion. The probability of our random variable X_{n+1}, representing the dance motion picked at beat $n+1$, to be the dance motion x_i, depends only on the previously picked dance motion $X_n = x_j$, and the current tempo in bpm. (With graphics from [18])

When designing dance motion primitives to build up our motion library, several considerations had to be done. The motion should ideally express the beat through its movement and also provide information for the dance choreographer to gauge how fitting a dance motion is to the current music. As a result, the following characteristics and design parameters are specified for each dance motion x_i.

1) Dance Motion Duration: Each dance motion primitive's duration is continuously set equal to an integer multiple of the current beat duration d_n.

2) Trajectory Amplitudes: By scaling certain dance motions' amplitude, e.g., the stomping height, linearly with the beat duration, we can express the music's temper.

3) Tempo Range: The tempo range $[\text{bpm}_{min,i}, \text{bpm}_{max,i}]$ defines the tempo limits within which the dance motion x_i is allowed to be executed.

4) Repeatability: Each dance motion is assigned a *repeatability* value r_i. This value can range from 0 to ∞, where a value of 0 indicates that a dance motion will never repeat itself and ∞ indicates that a dance motion, once it has started, will never stop repeating itself.

5) Characteristic Signal: To synchronize the motion with the music, a *characteristic signal* is defined for each dance motion. This signal should be representative for music-synchronous, or asynchronous behavior respectively. For example, the synchrony of a stomping motion with the beat is best characterized by the moment that the foot gains contact with the ground. Thus, the contact signal of the stomping foot is chosen as the characteristic signal. In case of a side-to-side motion of the base in y-direction, both the relative base position and velocity in y-direction, represent suitable characteristic signals.

6) Transition Behavior: Optimally, a transition connects two dance motions seamlessly without disturbing the flow of the dance, while remaining stable throughout the transition. This was achieved by grouping dance motions with similar start and end poses. Whenever a switch occurs from one group to another, a trajectory starting from the end pose of the previous dance motion and ending at the start pose of the next dance motion is sent to the robot. The duration of this trajectory is also scaled with the current beat tempo.

7) Trajectory Generation: Lastly, each dance motion has to be defined as a continuous trajectory that can be passed on to the whole-body controller. To generate such trajectories, we follow a *keyframe*-based design with the aid of the *Free Gait* [17] interface. Free Gait allows us to design motions by defining keyframe poses and their relative duration within the motion. They can be defined for the base, end effectors, and joints. In a final step, these keyframe poses are fitted using quintic polynomial splines, using the current robot state as starting keyframe, resulting in a continuous desired whole-body state. Fig. 3 illustrates this concept. Using this methodology, scaling a dance motion to be an integer multiple of the current beat duration, and scaling its amplitude according to the current tempo, is simplified to moving our keyframe poses in time and space.

B. Dance Choreography

Since our beat tracking stage does not receive any previous information about the music, we will try to emulate *improvised* dance. We concluded that a probabilistic model would best represent an improvised dance while taking musical information, theme and variation, and repetition into account. Specifically, this process was designed as a Markov chain. A Markov chain is a sequence of discrete random variables $X_1, X_2, X_3, \ldots$ with realization probabilities of the next state given by

$$\begin{aligned} &\Pr(X_{n+1} \mid X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) \\ &\quad = \Pr(X_{n+1} = x \mid X_n = x_n) \,, \end{aligned} \tag{4}$$

i.e. the probabilities of the next realization only depend on the current state. The possible values $x_1, x_2, \ldots, x_N$ of X_i form a countable set S called the *state space* of the chain.

Formulating such a Markov chain for creating a dance choreography, yields following: Our random variable X_n represents the dance motion picked at beat n. The state space S is given by our N user-defined dance-motion primitives $x_1, x_2, \ldots, x_N$.

Next, the probabilities $\Pr(X_{n+1} = x_i \mid X_n = x_j)$ for our N dance motions must be defined. These should be chosen such that an interesting and varied choreography can take place. This is done by first assigning each dance motion (denoted by i) a relative weight, which depends on the design parameters defined in Section III-A, i.e., the tempo range $[\text{bpm}_{\min,i}, \text{bpm}_{\max,i}]$ and repeatability r_i:

$$w_{i,n+1} = R_i B_{i,n+1} \,, \tag{5}$$

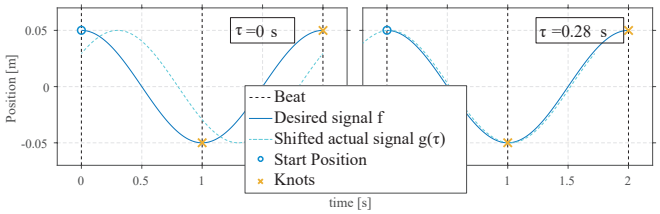

Fig. 5. We acquire both a beat-aligned reference signal and the actual robot signal. The delay is found by shifting the actual signal w.r.t. the reference signal and identifying the time shift which minimizes the least squares error.

where the *repeatability factor* is given by

$$R_i = \begin{cases} r_i, & \text{if } i = j \\ 1, & \text{else}\,, \end{cases} \tag{6}$$

and the *tempo range factor* by

$$B_{i,n+1} = \begin{cases} 1, & \text{if } \mathrm{bpm}_{n+1} \in [\mathrm{bpm}_{\min,i}, \mathrm{bpm}_{\max,i}] \\ 0 & \text{else}\,. \end{cases} \tag{7}$$

The repeatability factor takes into account how probable a dance motion is to be repeated, and the tempo range factor makes sure that only dance motions designed for the current music tempo are executed.

Finally, the probability of the next state being dance motion x_i is given by normalizing the weights $w_{i,n+1}$

$$\Pr(X_{n+1} = x_i \mid X_n = x_j) = \frac{w_{i,n+1}}{\sum_{i=1}^{N} w_{i,n+1}}\,. \tag{8}$$

An illustrative outline of our model can be seen in Fig. 4.

IV. Dance-Music Synchronization

Correctly timing dance movements to fit the music is indispensable for a successful dance. Our algorithm shall be robust against changes in tempo, external disturbances and long-term changes in robot dynamics. These requirements are met by applying time delay analysis on our desired and actual robot motion to determine the delay, and controlling the delay using a combined feedback and feedforward controller.

A. Delay Estimation

In a first step, the delay is estimated by shifting a measured signal w.r.t. a beat-synchronous reference signal and finding the time shift which minimizes the least squares error (LSE) between the reference and measured signal. The process of finding the delay between motion and music can then be divided into following three steps (see Fig. 5):

1) *Reference signal:* At first, a beat-synchronized motion has to be defined to act as the reference signal, which is achieved through the Free Gait interface. Using our keyframe poses (which are aligned to the current beat duration) from the dance motion to be executed next, the desired robot state is simulated for the length of the dance motion duration, resulting in a beat-synchronized robot state reference signal. As mentioned before, our measure of synchrony is based on a characteristic signal, defined for each dance motion separately. This characteristic signal is extracted from the simulated robot state and will act as the reference signal.
2) *Actual measured signal:* The actual signal to be synchronized is given by the robot state estimator. Just as for the reference signal, the measured characteristic signal for the duration of the motion will act as the actual signal which will be compared to the reference signal.
3) *Delay Estimation:* The last and most vital step lies in finding the time delay between the reference and actual signal. For explanatory purposes, the reference signal will be denoted by f and the actual signal by g, respectively. The basic idea of our algorithm is to shift f and g in time w.r.t. each other, while for each time shift τ the identicalness of the two signals is defined by the LSE. So, the LSE is given as a function of the time shift τ

$$||f-g||_2^2(\tau) = \frac{1}{T-|\tau|}\int_{\max(0,-\tau)}^{\min(T,T-\tau)} (f(t)-g(t+\tau))^2\,\mathrm{d}t\,, \tag{9}$$

where $\frac{1}{T-|\tau|}$ is a normalization factor, that accounts for the finite length T of the signal.
For each executed dance motion, the delay δ_n can then be found at the time shift which displays the best alignment, namely at the time shift which minimizes the LSE:

$$\delta_n = \arg\min_{\tau}\left(||f-g||_2^2(\tau)\right)\,. \tag{10}$$

B. Delay Controller

Having determined the delay, we are presented with two options compensate for it: We can either adjust the duration of the next motion or shift the entire trajectory in time. From previous experiments, we concluded that shortening dance motion primitives could lead to visually displeasing behavior. Thus, the latter option was chosen, as it does not distort, but only shift the motion in time. That is, the starting time of the next dance motion is set to

$$t_{\mathrm{start},n+1} = b_{n+1} - u_n \tag{11}$$

and the duration of the motion remains an integer multiple of d_n. Here, b_{n+1} is the extrapolated beat time (see Section II) and u_n the control action.

We chose to control the delay using a combined PID feedback and feedforward controller. This choice was taken based on the desire for a fast convergence time, a zero steady-state error and robust disturbance rejection. The control action at beat n, i.e., the time by which we shift the next dance motion, is then given by

$$u_n = u_{n,\mathrm{feedback}} + u_{n,\mathrm{feedforward}}\,. \tag{12}$$

1) Feedback Controller: The feedback control action is

$$u_{n,\mathrm{feedback}} = K_p\delta_n + K_i\sum_{m=1}^{n}\delta_m d_m + K_d\frac{\delta_n-\delta_{n-1}}{d_n}\,, \tag{13}$$

where K_p, K_i and K_d are controller parameters, and d_n the current duration of the beat.

2) Feedforward Controller: After a change in the reference signal, e.g., due to a change in music tempo, a pure feedback controller will usually take a certain amount of time to converge. Therefore we introduce a feedforward term, given by a lookup table, which shall ideally zero the convergence time.

Delays are assumed to be different across different tempi and dance motions, as both tempo and type of dance motion influence the robot velocity and its inertia, which in turn affect robot tracking performance. The lookup table's dimensions are therefore (1) the dance motion and (2) the bpm, where the bpm are divided into a grid of a user-defined resolution of B bpm. Moreover, we chose to implement a *dynamic* lookup table , as opposed to a *static* one. This design choice allows for changes in robot dynamics, e.g., new robot controller parameters, motors, etc.

The dynamic lookup table values $l_{i,[\text{bpm}]}$ are updated whenever the current control action successfully synchronizes the specific dance motion and the beat, i.e., when a delay below a user-defined threshold $\delta_n < \theta$ is detected. The lookup table values are then given by the moving average of the previous M control actions for which the delay for dance motion x_i was below the aforementioned threshold:

$$l_{i,[\text{bpm}]} = \frac{1}{\min(M,m)} \sum_{j=\max(1,m-M+1)}^{m} u_{0,i,j} \tag{14}$$

$$u_{0,i,m} = u_n, \quad \text{if } \delta_n < \theta \tag{15}$$

where i denotes the dance motion, [bpm] the bpm rounded to fit the bpm resolution, and $m = 1,2,3,\ldots$. Furthermore, if a lookup table value gets updated, the integrator of the feedback controller is reduced by the difference between the new and old lookup table value, so that jumps, when reading from a just updated value, are prevented.

Reading from the lookup table using the dance motion to be executed next and the current tempo then yields

$$u_{n,\text{feedforward}} = l_{i_n,[\text{bpm}_n]}\,, \tag{16}$$

with $\text{bpm}_n = \frac{60\,\text{s}}{d_n}$.

V. Results

To assess the performance of the different submodules, different experiments were conducted. Focus is laid primarily on the delay compensator, as this stage produces the most quantifiable results and is the main focus of this work.

A. Experimental Setup

Our experiments were conducted on the quadrupedal robot ANYmal [14] which is 0.5 m tall and weighs 30 kg. It features three torque-controllable actuated joints per leg and point feet equipped with contact sensors. The joint arrangement is chosen mammalian with successive hip abduction/adduction, hip flexion/extension, and knee flexion/extension. Onboard batteries allow for autonomous operation of up to two hours. Lastly, it is equipped with a mono microphone with a sampling rate of 44'100 Hz.

All experiments were performed on the real robot, with all computation being done onboard. A speaker set up next to the robot served as an external music source. The final trajectory output by our system acts as the reference signal for the whole-body controller which communicates with the drives at a rate of 400 Hz.

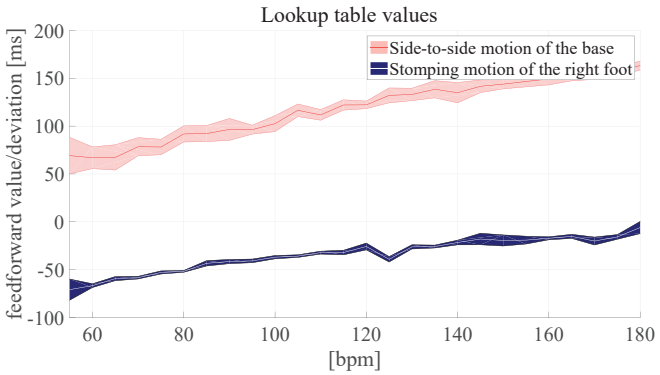

Fig. 6. Values and their standard deviation in the filled lookup table for different dance moves. An upward trend for higher tempos is observed.

At the end of this work, we had designed a total of 7 dance motions, including a side-to-side motion of the base, two different stomping motions of the right and left foot, and twerking dance motions.

B. Lookup Table Values

The lookup table was continuously filled while conducting experiments, both with simulated internal beats and beats from the beat tracker. Fig. 6 shows the lookup table values for two different dance motions. It is apparent that higher tempi lead to higher compensation values. This can be explained with the increasing robot velocities at higher tempi and the inherent difficulty for the whole-body controller to track faster motions.

Also, compensation values are different across different dance motions. Moreover, the stomping motion's delays are negative, meaning the motion finishes too early without control action. This is due to the high inertia of the robot base rendering base movements much harder to track than a lighter leg motion, and secondly, due to the stomping motion's end position being set below the floor, so that the end effector has a high impact.

C. Convergence for Alternating Music Tempo

In the following, the performance of our delay compensation system is evaluated. For this experiment, the available dance motions were reduced to a single one, namely a simple side-to-side motion of the base, as its visual trajectory can easily be interpreted. To solely determine the performance of our delay compensation method, we bypassed the beat tracker and fed a perfectly timed beat with 60 and then 120 bpm to our delay compensation stage. We compared performance between (1) having no delay controller, (2) only engaging the feedback delay controller, and (3) engaging the full combined feedback and feedforward controller. Each scenario was run ten times, in order to statistically assess the performance of the delay compensation. The trajectories can be seen in Fig. 7 and the delays are plotted in Fig. 8.

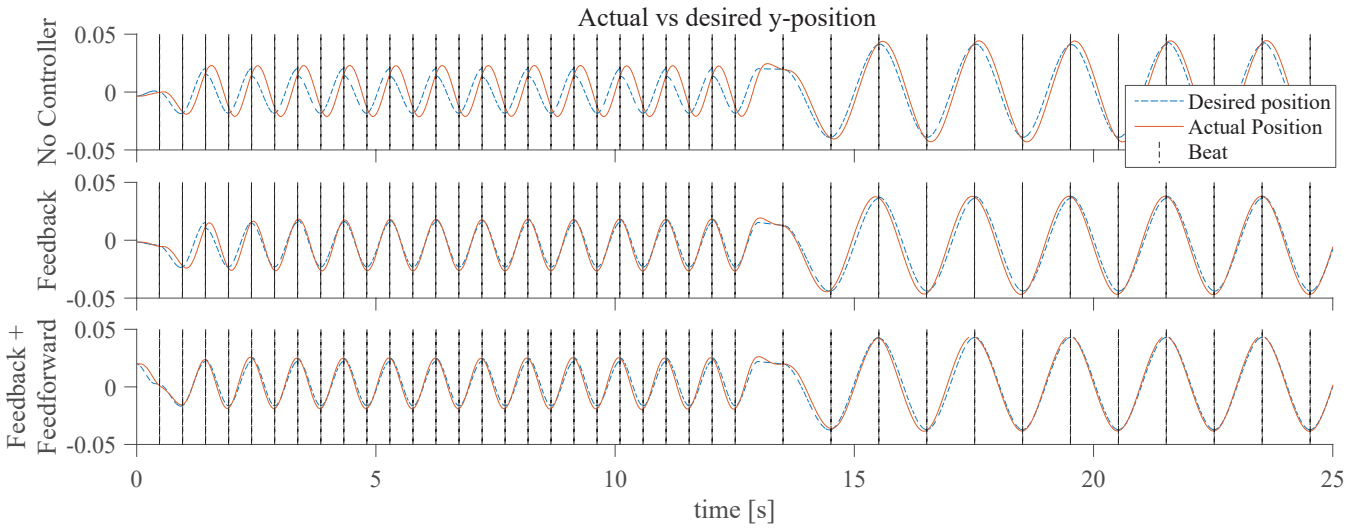

Fig. 7. The robot trajectory for a side-to-side motion, for (1) no controller, (2) feedback controller, and (3) feedback + forward controller.

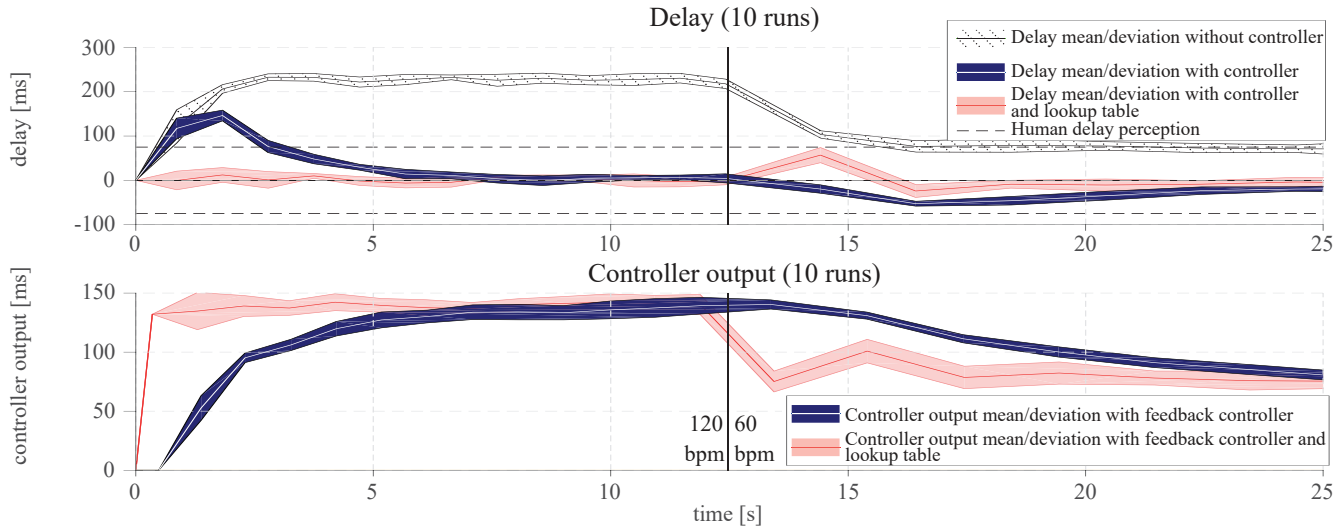

Fig. 8. A side-to-side motion of the robot was run ten times for an alternating tempo. The delay and correction mean and standard deviation are plotted for different control setups.

Here, the measure of interest is both the delay convergence time, i.e., the time it takes for the delay to converge to zero after recognizing a new tempo/song, and the time it takes to reach delays that cannot be humanly perceived. According to [19], any audio-visual delays within -75 ms to 175 ms are mostly perceived to be synchronous by human observers. These limits will thus serve as our measure of reaching a synchronized motion. Furthermore, we reduced this range to -75 ms to 75 ms, so as to treat negative and positive delays equally.

When disabling the controller, a steady-state delay settles in for both tempos. According to our pre-defined human audio-visual delay perception limits, this should be clearly noticeable for 120 bpm and barely noticeable at 60 bpm. These results justify the need for any form of delay compensation.

In the scenario of solely relying on the PID feedback controller, perceived synchrony is achieved after approximately 3 s and full convergence to 0 delay after 7 s. This is a relatively fast convergence, however, periods of undesirable asynchrony (delay outside of perception limits) are observed.

In the case of the full delay controller, immediate convergence can be observed, with the mean and its standard deviation never reaching humanly observable delays, emphasizing the added benefit of adding a feedforward term.

D. External Disturbances

As a four-legged robot, the ANYmal robot may often find itself exposed external and unforeseeable disturbances. For this reason, the performance of our system in reaction to external disturbance is evaluated.

First, the available dance motions were again reduced to the same side-to-side motion for the aforementioned reasons.

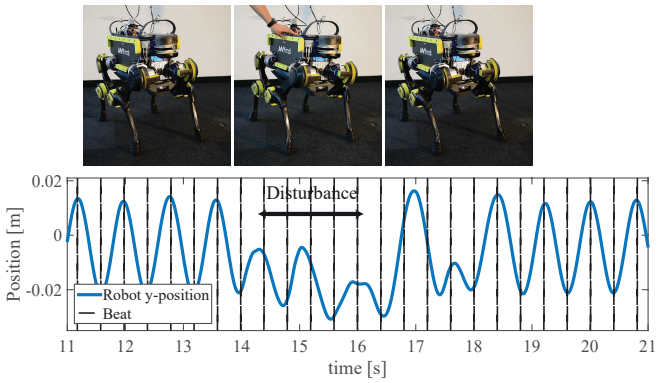

Fig. 9. A side-to-side motion of the robot is interrupted by pushing against the robot. Thanks to the delay compensation, the robot recovers quickly after being released.

Pushing against the robot once it reached one side, effectively hindering it from reaching the other side (see Fig. 9), served as our external disturbance.

Fig. 9 illustrates the results of the experiments. The robot was held in place starting at 14 s and was released at 16.5 s. After releasing the robot, they system executes one more full side-to-side motion, after which the delay induced by the disturbance is detected. Consequently the next motion is shifted back in time by the PID controller, and the robot is able to return to its synchronized state, within just one full motion.

E. End-to-End Behavior

The synchronization of music and dance has now been shown to achieve the desired results. However, as the final system is designed to appeal to a human observer, the quality of the end-to-end behavior is difficult to quantify without the evaluation of human observers. Therefore, the accompanying video demonstrates the robot's dancing capabilities when employing the full system. An additional video demonstrating the robot dancing to a live cello performance is available online[1].

VI. Conclusion

We designed a framework for autonomous dance as a reaction to external music on a quadrupedal robot. At the core of this framework lies our delay synchronization algorithm which determines the delay using LSE and controls the delay with a PID feedback plus feedforward controller. This controller achieves an immediate zero-delay convergence. With an estimated average of 4 s for the beat tracker to converge, the resulting end-to-end convergence from tempo change to synchronized motion adds up to around 4 s. Also, the steady-state delay is well within the limits of the human delay perception. Moreover, the system is able to react to external disturbances in a robust manner, and recovers within 1–2 s.

We have successfully built the foundation for a fully autonomous, improvising and synchronized dancing robot. Nevertheless, the robot is still limited in its ability to entertain for a long duration and appears robot-like, due to its still limited variation in dance choreography. Future work can be done to achieve our ultimate goal of building a system, which outputs natural human-like dance, with great variations in movement and seamless transitions from one song to another.

[1] `https://youtu.be/kHBLaw5nfzk`

References

[1] C. L. Sidner, C. Lee, C. D. Kidd, N. Lesh, and C. Rich, "Explorations in engagement for humans and robots," *Artificial Intelligence*, vol. 166, no. 1-2, pp. 140–164, 2005.

[2] "Robots - who is pepper?." `https://www.ald.softbankrobotics.com/en/cool-robots/pepper`. Accessed: 2018-07-26.

[3] J.-J. Aucouturier, "Cheek to Chip: Dancing Robots and AI's Future," *IEEE Intelligent Systems*, vol. 23, no. 2, pp. 74–84, 2008.

[4] L. Geppert, "Qrio, the robot that could," *IEEE Spectrum*, vol. 41, no. 5, pp. 34–37, 2004.

[5] S. Nakaoka, A. Nakazawa, F. Kanehiro, K. Kaneko, M. Morisawa, and K. Ikeuchi, "Task model of lower body motion for a biped humanoid robot to imitate human dances," in *IEEE Intelligent Robots and Systems*, pp. 3157–3162, 2005.

[6] C. Crick, M. Munz, and B. Scassellati, "Synchronization in Social Tasks: Robotic Drumming," in *IEEE International Symposium on Robot and Human Interactive Communication*, pp. 97–102, 2006.

[7] S. Bock, F. Krebs, A. Durand, S. Poll, and R. Balsyte, "Robod: a real-time online beat and offbeat drummer," 2016.

[8] K. Murata, K. Nakadai, K. Yoshii, R. Takeda, T. Torii, H. Okuno, Y. Hasegawa, and H. Tsujino, "A robot uses its own microphone to synchronize its steps to musical beats while scatting and singing," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2459–2464, 2008.

[9] A. Schöllig, F. Augugliaro, S. Lupashin, and R. D'Andrea, "Synchronizing the Motion of a Quadrocopter to Music," *IEEE International Conference on Robotics and Automation*, pp. 3355–3360, 2010.

[10] C. B. Santiago, J. L. Oliveira, L. P. Reis, and A. Sousa, "Autonomous robot dancing synchronized to musical rhythmic stimuli," in *Conference on Information Systems and Technologies*, IEEE, 2011.

[11] K. Yoshii, K. Nakadai, T. Torii, Y. Hasegawa, H. Tsujino, K. Komatani, T. Ogata, and H. G. Okuno, "A biped robot that keeps steps in time with musical beats while listening to music with its own ears," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1743–1750, 2007.

[12] D. Pongas, A. Billard, and S. Schaal, "Rapid synchronization and accurate phase-locking of rhythmic motor primitives," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2911–2916, 2005.

[13] S. Kotosaka and S. Schaal, "Synchronized Robot Drumming by Neural Oscillator.," *Journal of the Robotics Society of Japan*, vol. 19, no. 1, pp. 116–123, 2001.

[14] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. A. Hoepflinger, "ANYmal - A Highly Mobile and Dynamic Quadrupedal Robot," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[15] S. Böck, F. Krebs, and G. Widmer, "A multi-model approach to beat tracking considering heterogeneous music styles.," in *ISMIR*, 2014.

[16] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "madmom: a new Python Audio and Music Signal Processing Library," in *ACM International Conference on Multimedia*, pp. 1174–1178, 10 2016.

[17] P. Fankhauser, D. Bellicoso, C. Gehring, and M. Hutter, "Free Gait – An Architecture for the Versatile Control of Legged Robots," *IEEE-RAS International Conference Humanoid Robots (Humanoids)*, 2016.

[18] Lineartestpilot, "Vector cartoon robots dancing the robot - shutterstock." `https://www.shutterstock.com/image-vector/vector-cartoon-robots-dancing-robot-line-46013350`. Accessed: 2018-07-31.

[19] S. van de Par and A. Kohlrausch, "Sensitivity to auditory-visual asynchrony and to jitter in auditory-visual timing," pp. 234–242, International Society for Optics and Photonics, 2000.