EURASIP Journal on Audio, Speech, and Music Processing
a SpringerOpen Journal

RESEARCH Open Access

Combined perception and control for timing in robotic music performances

Umut Şimşekli*, Orhan Sönmez, Barış Kurt and Ali Taylan Cemgil

Abstract

Interaction with human musicians is a challenging task for robots as it involves online perception and precise synchronization. In this paper, we present a consistent and theoretically sound framework for combining perception and control for accurate musical timing. For the perception, we develop a hierarchical hidden Markov model that combines event detection and tempo tracking. The robot performance is formulated as a linear quadratic control problem that is able to generate a surprisingly complex timing behavior in adapting the tempo. We provide results with both simulated and real data. In our experiments, a simple Lego robot percussionist accompanied the music by detecting the tempo and position of clave patterns in the polyphonic music. The robot successfully synchronized itself with the music by quickly adapting to the changes in the tempo.

Keywords: hidden Markov models, Markov decision processes, Kalman filters, robotic performance

1 Introduction

With the advances in computing power and accurate sensor technologies, increasingly more challenging tasks in human-machine interaction can be addressed, often with impressive results. In this context, programming robots that engage in music performance via real-time interaction remained as one of the challenging problems in the field. Yet, robotic performance is criticized for being to mechanical and robotic [1]. In this paper, we therefore focus on a methodology that would enable robots to participate in natural musical performances by mimicking what humans do.

Human-like musical interaction has roughly two main components: a perception module that senses what other musicians do and a control module that generates the necessary commands to steer the actuators. Yet, in contrast to many robotic tasks in the real world, musical performance has a very tight realtime requirement. The robot needs to be able to adapt and synchronize well with the tempo, dynamics and rhythmic feel of the performer and this needs to be achieved within hard real-time constraints. Unlike repetitive and dull tasks, such expressive aspects of musical performance are hard to formalize and realize on real robots. The existence of humans in the loop makes the task more challenging as a human performer can be often surprisingly unpredictable, even on seemingly simple musical material. In such scenarios, highly adaptive solutions, that combine perception and control in an effective manner, are needed.

Our goal in this paper is to illustrate the coupling of perception and control modules in music accompaniment systems and to reveal that even with the most basic hardware, it is possible to carry out this complex task in real time.

In the past, several impressive demonstrations of robotic performers have been displayed, see Kapur [2] as a recent survey. The improvements in the field of human-computer interaction and interactive computer music systems influenced the robotic performers to listen and respond to human musicians in a realistic manner. The main requirement for such an interaction is a tempo/beat tracker, which should run in real-time and enable the robot to synchronize well with the music.

As a pioneering work, Goto and Muraoka [3] presented a real-time beat tracking for audio signals without drums. Influenced by the idea of an untrained listener can track the musical beats without knowing the names of the chords or the notes being played, they based their method on detecting the chord changes. The method performed well on popular music; however, it is

* Correspondence: umut.simsekli@boun.edu.tr
Department of Computer Engineering, Boğaziçi University, 34342, Bebek, Istanbul, Turkey

hard to improve or adapt the algorithm for a specific domain since it was built on top of many heuristics. Another interesting work on beat tracking was presented in Kim et al. [4], where the proposed method estimates the tempo of rhythmic motions (like dancing or marching) through a visual input. They first capture the 'motion beats' from sample motions in order to capture the transition structure of the movements. Then, a new rhythmic motion synchronized with the background music is synthesized using this movement transition information.

An example of an interactive robot musician was presented by Kim et al. [5], where the humanoid robot accompanied the playing music. In the proposed method, they used both audio and visual information to track the tempo of the music. In the audio processing part, an autocorrelation method is employed to determine the periodicity in the audio signal, and then, a corresponding tempo value is estimated. Simultaneously, the robot tracks the movements of a conductor visually and makes another estimation for the tempo [6]. Finally, the results of these two modules are merged according to their confidences and supplied to the robot musician. However, this approach lacks an explicit feedback mechanism which is supposed to handle the synchronization between the robot and the music.

In this paper, rather than focusing on a particular piece of custom build hardware, we will focus on a deliberately simple design, namely a Lego robot percussionist. The goal of our percussionist will be to follow the tempo of a human performer and generate a pattern to play in sync with the performer. A generic solution to this task, while obviously simpler than that for an acoustic instrument, captures some of the central aspects or robotic performance, namely:

- Uncertainties in human expressive performance
- Superposition–sounds generated by the human performer and robot are mixed
- Imperfect perception
- Delays due to the communication and processing of sensory data
- Unreliable actuators and hardware–noise in robot controls causes often the actual output to be different than the desired one.

Our ultimate aim is to achieve an acceptable level of synchronization between the robot and a human performer, as can be measured via objective criteria that correlate well with human perception. Our novel contribution here is the combination of perception and control in a consistent and theoretically sound framework.

For the perception module, we develop a hierarchical hidden Markov model (a changepoint model) that combines event detection and tempo tracking. This module combines the template matching model proposed by Şimşekli and Cemgil [7] and the tempo tracking model by Whiteley et al. [8] for event detection in sound mixtures. This approach is attractive as it enables to separate sounds generated by the robot or a specific instrument of the human performer (clave, hi-hat) in a supervised and online manner.

The control model assumes that the perception module provides information about the human performer in terms of an observation vector (a bar position/tempo pair) and an associated uncertainty, as specified possibly by a covariance matrix. The controller combines the observation with the robots state vector (here, specified as an angular-position/angular-velocity pair) and generates an optimal control signal in terms of minimizing a cost function that penalizes a mismatch between the "positions" of the robot and the human performer. Here, the term position refers to the score position to be defined later. While arguably more realistic and musically more meaningful cost functions could be contemplated, in this paper, we constrain the cost to be quadratic to keep the controller linear.

A conceptually similar approach to ours was presented by Yoshii et al. [9], where the robot synchronizes its steps with the music by a real-time beat tracking and a simple control algorithm. The authors use a multi-agent strategy for real-time beat tracking where several agents monitor chord changes and drum patterns and propose their hypotheses; the most reliable hypothesis is selected. While the robot keeps stepping, the step intervals are sent as control signals from a motion controller. The controller calculates the step intervals in order to adjust and synchronize the robots stepping tempo together with beat timing. Similar to this work, Murata et al. [10] use the same robotic platform and controller with an improved beat-tracking algorithm that uses a spectro-temporal pattern matching technique and echo cancelation. Their tracking algorithm deals better with environmental noise and responds faster to tempo changes. However, the proposed controller only synchronizes the beat times without considering which beat it is. This is the major limitation of these systems since it may allow phase shifts in beats if somebody wants to synchronize a whole musical piece with the robot.

Our approach to tempo tracking is also similar to the musical accompaniment systems developed by Dannenberg [11], Orio [12], Cemgil and Kappen [13], Raphael [14], yet it has two notable novelties. The first one is a novel hierarchical model for accurate online tempo estimation that can be tuned to specific events, while not assuming the presence of a particular score. This enables us to use the system in a natural setting where the sounds generated by the robot and the other performers

are mixed. This is in contrast to existing approaches where the accompaniment only tracks a target performer while not listening to what it plays. The second novelty is the controller component, where we formulate the robot performance as a linear quadratic control problem. This approach requires only a handful of parameters and seems to be particularly effective for generating realistic and human-like expressive musical performances, while being fairly straightforward to implement.

The paper is organized as follows. In the sequel, we elaborate on the perception module for robustly inferring the tempo and the beat from polyphonic audio. Here, we describe a hierarchical hidden Markov model. Section 3 introduces briefly the theory of optimal linear quadratic control and describes the robot performance in this framework. Sections 4 describes simulation results. Section 5 describes experiments with our simple Lego robot system. Finally Section 6 describes the conclusions, along with some future directions for further research.

2 The perception model

In this study, the aim of the perception model is to jointly infer the tempo and the beat position (score position) of a human performer from streaming polyphonic audio data in an online fashion. Here, we assume that the observed audio includes a certain instrument that carries the tempo information such as a hi-hat or a bass drum. We assume that this particular instrument is known beforehand. The audio can include other instrument sounds, including the sound of the percussion instrument that the robot plays.

As the scenario in this paper, we assume that the performer is playing a *clave* pattern. The claves is the name for both a wooden percussive instrument and a rhythmic pattern that organizes the temporal structure and forms the rhythmic backbone in Afro-Cuban music. Note that, this is just an example, and our framework can be easily used to track other instruments and/or rhythmic patterns in a polyphonic mixture.

In the sequel, we will construct a probabilistic generative model which relates latent quantities, such as acoustic event labels, tempi, and beat positions, to the actual audio recording. This model is an extension that combines ideas from existing probabilistic models: the bar pointer model proposed by Whiteley et al. [8] for tempo and beat position tracking and an acoustic event detection and tracking model proposed by Şimşekli and Cemgil [7].

In the following subsections, we explain the probabilistic generative model and the associated training algorithm. The main novelty of the current model is that it integrates tempo tracking with minimum delay online event detection in polyphonic textures.

2.1 Tempo and acoustic event model

In [8], Whiteley et al. presented a probabilistic "bar pointer model", which modeled one period of a hidden rhythmical pattern in music. In this model, one period of a rhythmical pattern (i.e., one bar) is uniformly divided into M discrete points, so called the "position" variables, and a "velocity" variable is defined with a state space of N elements, which described the temporal evolution of these position variables. In the bar pointer model, we have the following property:

$$m_\tau = \left(\left\lfloor m_{\tau-1} + f(n_{\tau-1})\right\rfloor\right) \mod M. \tag{1}$$

Here, $m_\tau \in \{0, \ldots, M - 1\}$ are the position variables, $n_\tau \in \{0, \ldots, N\}$ are the velocity variables, $f(\cdot)$ is a mapping between the velocity variables n_τ and some real numbers, $\lfloor\cdot\rfloor$ is the floor operator, and τ denotes the time frame index. To be more precise, m_τ indicate the position of the music in a bar and n_τ determine how fast m_τ evolve in time. This evolution is deterministic or can be seen as probabilistic with a degenerate probability distribution. The velocity variables, n_τ , are directly proportional to the tempo of the music and have the following Markovian prior:

$$p(n_\tau|n_{\tau-1}) = \begin{cases} \frac{p_n}{2}, & n_\tau = n_{\tau-1} \pm 1 \\ 1 - p_n, & n_\tau = n_{\tau-1} \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where p_n is the probability of a change in velocity. When the velocity is at the boundaries, in other words if $n_\tau = 1$ or $n_\tau = N$, the velocity does not change with probability, p_n, or transitions respectively to $n_{\tau+1} = 2$ or $n_{\tau+1} = N - 1$ with probability $1 - p_n$. The modulo operator reflects the periodic nature of the model and ensures that the position variables stay in the set $\{0, \ldots, M - 1\}$.

In order to track a clave pattern from a sound mixture, we extend the bar pointer model by adding a new acoustic event variable. For each time frame τ, we define an indicator variable r_τ on a discrete state space of R elements, which determines the acoustic event label we are interested in. In our case, this state space may consist of event labels such as {claves hit, bongo hit, . . . , silence}. Since we are dealing with clave patterns, we can assume that the rhythmic structure of the percussive sound is constant, as the clave is usually repeated over the whole musical piece [15]. With this assumption, we come up with the following transition model for r_τ. For simplicity, we assume that $r_\tau = 1$ indicates r_τ = {claves hit}.

$$p(r_\tau|r_{\tau-1}, n_{\tau-1}, m_{\tau-1}) = \begin{cases} \frac{1}{R-1}, & r_\tau = i,\ r_{\tau-1} = 1,\ \forall i \in \{2, \ldots, R\} \\ 1, & r_\tau = 1,\ r_{\tau-1} \neq 1,\ \mu(m_\tau) = 1 \\ \frac{1}{R-1}, & r_\tau = i,\ r_{\tau-1} \neq 1,\ \mu(m_\tau) \neq 1,\ \forall i \in \{2, \ldots, R\} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where m_τ is defined as in Equation 1 and $\mu(\cdot)$ is a Boolean function which is defined as follows:

$$\mu(m) = \begin{cases} 1, & m \text{ is a position in a bar where a claves hit occurs} \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

Essentially, this transition model assumes that the claves hits can only occur on the beat positions, which are defined by the clave pattern. A similar idea for clave modeling was also proposed in Wright et al. [16].

By eliminating the self-transition of the claves hits, we prevent the "double detection" of a claves hit (i.e., detecting multiple claves hits in a very short amount of time). Figure 1 shows the *son clave* pattern, and Figure 2 illustrates the state transitions of the tempo and acoustic event model for the son clave. In the figure, the shaded nodes indicate the positions, where the claves hits can happen.

Note that, in the original bar pointer model definition, there are also other variables such as the *meter indicator* and the *rhythmic pattern indicator* variables, which we do not use in our generative model.

2.2 Signal model

Şimşekli and Cemgil presented two probabilistic models for acoustic event tracking in Şimşekli and Cemgil [7] and demonstrated that these models are sufficiently powerful to track different kinds of acoustic events such as pitch labels [7,17,18] and percussive sound events [19]. In our signal model, we use the same idea that was presented in the acoustic event tracking model [7]. Here, the audio signal is subdivided into frames and represented by their magnitude spectrum, which is calculated with discrete Fourier transform. We define $x_{\nu,\tau}$ as the magnitude spectrum of the audio data with frequency index ν and time frame index τ, where $\nu \in \{1, 2, \ldots, F\}$ and $\tau \in \{1, 2, \ldots, T\}$.

The main idea of the signal model is that each acoustic event (indicated by r_τ) has a certain characteristic spectral shape which is rendered by a specific hidden *volume* variable, v_τ. The spectral shapes, so-called *spectral templates*, are denoted by $t_{\nu,i}$. The ν index is again the frequency index, and the index i indicates the event labels. Here, i takes values between 1 and R, where R has been defined as the number of different acoustic events. The volume variables v_τ define the overall amplitude factor, by which the whole template is multiplied.

By combining the *tempo and acoustic event model* and the *signal model*, we define our hybrid *perception model* as follows:

$$\begin{aligned} n_0 &\sim p(n_0),\ m_0 \sim p(m_0),\ r_0 \sim p(r_0) \\ n_\tau \,|\, n_{\tau-1} &\sim p(n_\tau \,|\, n_{\tau-1}) \\ m_\tau \,|\, m_{\tau-1},\ n_{\tau-1} &= \left(\left\lfloor m_{\tau-1} + f(n_{\tau-1}) \right\rfloor\right) \mod M \\ r_\tau | r_{\tau-1},\ m_{\tau-1},\ n_{\tau-1} &\sim p(r_\tau \,|\, r_{\tau-1},\ m_{\tau-1},\ n_{\tau-1}) \\ v_\tau &\sim \mathcal{G}(v_\tau; a_v,\ b_v) \\ x_{\nu,\tau} | r_\tau, v_\tau &\sim \prod_{i=1}^{I} \mathcal{PO}(x_{\nu,\tau}; t_{\nu,i} v_\tau)^{[r_\tau = i]}, \end{aligned} \quad (5)$$

where, again, m_τ indicate the position in a bar, n_τ indicate the velocity, r_τ are the event labels (i.e., $r_\tau = 1$ indicates a claves hit), v_τ are the volume of the played template, $t_{\nu,i}$ are the spectral templates, and finally, $x_{\nu,\tau}$ are the observed audio spectra. Besides, here, the prior distrubutions, $p(n_\tau \,|\cdot)$ and $p(r_\tau \,|\cdot)$ are defined in Equations 2 and 3, respectively. $[x]$ is the indicator function, where $[x] = 1$ if x is true, $[x] = 0$ otherwise and the symbols $\mathcal{G}$ and $\mathcal{PO}$ represent the Gamma and the Poisson distributions respectively, where

$$\begin{aligned} \mathcal{G}(x; a,\ b) &= \exp((a-1)\log x - bx - \log\Gamma(a) + a\log(b)) \\ \mathcal{PO}(x; \lambda) &= \exp(x\log\lambda - \lambda - \log\Gamma(x+1)), \end{aligned} \quad (6)$$

where Γ is the Gamma function. Figure 3 shows the graphical model of the perception model. In the graphical model, the nodes correspond to probability distributions of model variables and edges to their conditional dependencies. The joint distribution can be rewritten by making use of the directed acyclic graph:

$$\begin{aligned} p(n_{1:T},\ m_{1:T},\ r_{1:T},\ v_{1:T},\ x_{1:F,1:T}) = \prod_{\tau=1}^{T} &\left(p(n_\tau \,|\, \mathrm{pa}(n_\tau))p(m_\tau \,|\, \mathrm{pa}(m_\tau))p(r_\tau \,|\, \mathrm{pa}(r_\tau))\right. \\ &\left. p(v_\tau \,|\, \mathrm{pa}(v_\tau)) \prod_{\nu=1}^{F} p(x_{\nu,\tau} \,|\, \mathrm{pa}(x_{\nu,\tau}))\right), \end{aligned} \quad (7)$$

where $\mathrm{pa}(\chi)$ denotes the *parent nodes* of χ.

The Poisson model is chosen to mimic the behavior of popular NMF models that use the KL divergence as the error metric when fitting a model to a spectrogram [20,21]. We also choose Gamma prior on v_τ to preserve

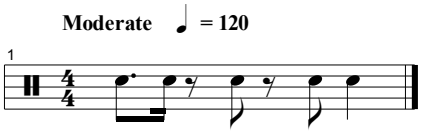


Figure 1 The 3 -2 *son clave* pattern which is written in 4/4. The hits are on the 1st, 4th, 7th, 11th, and the 13th sixteenth beats of a bar.

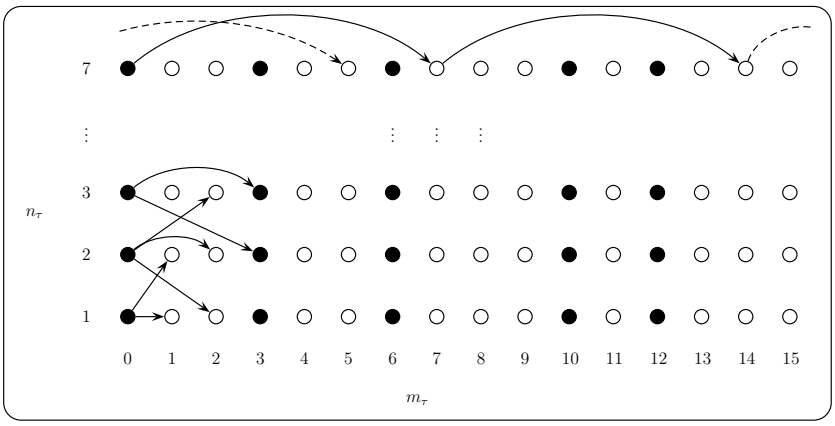


Figure 2 An example of state transition diagram of the position, velocity, and the acoustic event subspace for $M = 16$ and $N = 9$. The lines represent examples of possible state transitions where $f(n_\tau) = n_\tau$. The shaded nodes indicate the position in a bar where claves can hit, provided that the rhythm pattern is the son clave. In other words, for this particular model, $\mu(m) = 1, \forall m \in \{0, 3, 6, 10, 12\}$.

conjugacy and make use of the scaling property of the Gamma distribution.

An attractive property of the current model is that we can integrate out analytically the volume variables, v_τ. Hence, given that the templates $t_{\nu,i}$ are already known, the model reduces to a standard hidden Markov model with a Compound Poisson observation model and a latent state space of $D_n \times D_m \times D_r$, where $\times$ denotes the Cartesian product and D_n, D_m, and D_r are the state spaces of the discrete variables n_τ, m_τ, and r_τ, respectively. The Compound Poisson model is defined as follows (see Şimşekli [17] for details):

$$\begin{aligned} p(x_{1:F,\tau} \,|\, r_\tau = i) &= \int \mathrm{d}v_\tau \exp\left(\sum_{\nu=1}^{F} \log \mathcal{PO}(x_{\nu,\tau}; v_\tau t_{\nu,i}) + \log \mathcal{G}(v_\tau; a_\nu, b_\nu)\right) \\ &= \frac{\Gamma\left(\sum_{\nu=1}^{F} x_{\nu,\tau} + a_\nu\right)}{\Gamma(a_\nu) \prod_{\nu=1}^{F} \Gamma(x_{\nu,\tau} + 1)} \frac{b_\nu^{a_\nu} \prod_{\nu=1}^{F} t_{\nu,i}^{x_{\nu,\tau}}}{\left(\sum_{\nu=1}^{F} t_{\nu,i} + b_\nu\right)^{\sum_{\nu=1}^{F} x_{\nu,\tau} + a_\nu}}. \end{aligned} \tag{8}$$

Since we have a standard HMM from now on, we can run the forward-backward algorithm in order to

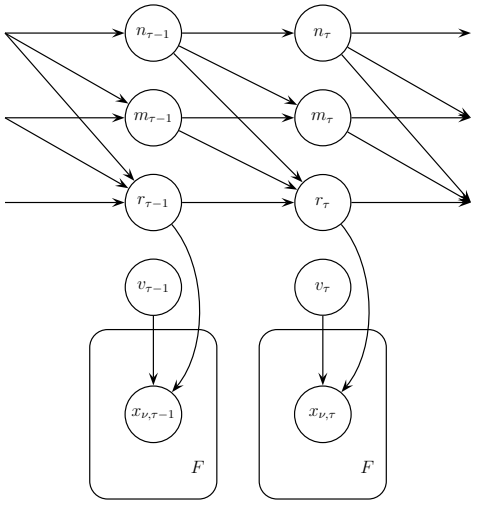


Figure 3 Graphical model of the perception model. This graph visualizes the conditional independence structure between the random variables and allows the joint distribution to be rewritten by utilizing Equation 7. Note that we use the plate notation for the observed variables where F distinct nodes (i.e., $x_{\nu,\tau}$ where $\nu \in \{1, \ldots, F\}$) are grouped and represented as a single node in the graphical model. In this case, F is the number or frequency bins.

compute the filtering or smoothing densities. Also, we can estimate the most probable state sequence by running the Viterbi algorithm. A benefit of having a standard HMM is that the inference algorithm can be made to run very fast. This lets the inference scheme to be implemented in real-time without any approximation [22]. Detailed information about the forward backward algorithm can be found in "Appendix A".

One point here deserves attention. The Poisson observation model described in this section is not scale invariant; i.e., turning up the volume can affect the performance. The Poisson model can be replaced by an alternative that would achieve scale invariance. For example, instead of modeling the intensity of a Poisson, we could assume conditionally Gaussian observations and model the variance. This approach corresponds to using a Itakura-Saito divergence rather than the Kullback-Leibler divergence [23]. However, in practice, scaling the input volume to a specific level is sufficiently good enough for acceptable tempo tracking performance.

2.3 Training

As we have constructed our inference algorithm with the assumption of the spectral templates $t_{v,i}$ to be known, they have to be learned at the beginning. In order to learn the spectral templates of the acoustic events, we do not need the tempo and the bar position information of the training data. Therefore, we reduce our model into the model that was proposed in Şimşekli et al. [19], so that we only care about the label and the volume of the spectral templates. The reduced model is as follows:

$$\begin{aligned} r_0 &\sim p(r_0) \\ r_\tau | r_{\tau-1} &\sim p(r_\tau | r_{\tau-1}) \\ v_\tau &\sim \mathcal{G}(v_\tau; a_v, b_v) \\ x_{v,\tau} | r_\tau, v_\tau &\sim \prod_{i=1}^{I} \mathcal{PO}(x_{v,\tau}; t_{v,i} v_\tau)^{[r_\tau = i]}. \end{aligned} \tag{9}$$

In order to learn the spectral templates, in this study, we utilize the expectation-maximization (EM) algorithm. This algorithm iteratively maximizes the log-likelihood via two steps:

E-step:

$$q(r_{1:T},\ v_{1:T})^{(n)} = p(r_{1:T},\ v_{1:T} \,|\, x_{1:F,1:T},\ t_{1:F,1:I}^{(n-1)}) \tag{10}$$

M-step:

$$t_{1:F,1:I}^{(n)} = \arg\max_{t_{1:F,1:I}} \langle \log p(r_{1:T},\ v_{1:T},\ x_{1:F,1:T} \,|\, t_{1:F,1:I}) \rangle_{q(r_{1:T}, v_{1:T})^{(n)}} \tag{11}$$

where $\langle f(x) \rangle_{p(x)} = \int p(x) f(x)\, \mathrm{d}x$ is the expectation of the function $f(x)$ with respect to $p(x)$.

In the E-step, we compute the posterior distributions of r_τ and v_τ. These quantities can be computed via the forward-backward algorithm (see "Appendix A"). In the M-step, we aim to find the $t_{v,i}$ that maximize the likelihood. Maximization over $t_{v,i}$ yields the following fixed-point equation:

$$t_{v,i}^{(n)} = \frac{\sum_{\tau=1}^{T} \langle [r_\tau = i] \rangle^{(n)} x_{v,\tau}}{\sum_{\tau=1}^{T} \langle [r_\tau = i] v_\tau \rangle^{(n)}}. \tag{12}$$

Intuitively, we can interpret this result as the weighted average of the normalized audio spectra with respect to v_τ.

3 The control model

The goal of the control module is to generate the necessary control signals to accelerate and decelerate the robot such that the performed rhythm matches the performance by its tempo and relative position. As observations, the control model makes use of the bar position and velocity (tempo) estimates m_τ and n_τ inferred by the perception module and possibly their associated uncertainties. In addition, the robot uses additional sensor readings to determine its own state, such as the angular velocity and angular position of its rotating motors axis that is connected directly to the drum sticks.

3.1 Dynamic linear system formulation

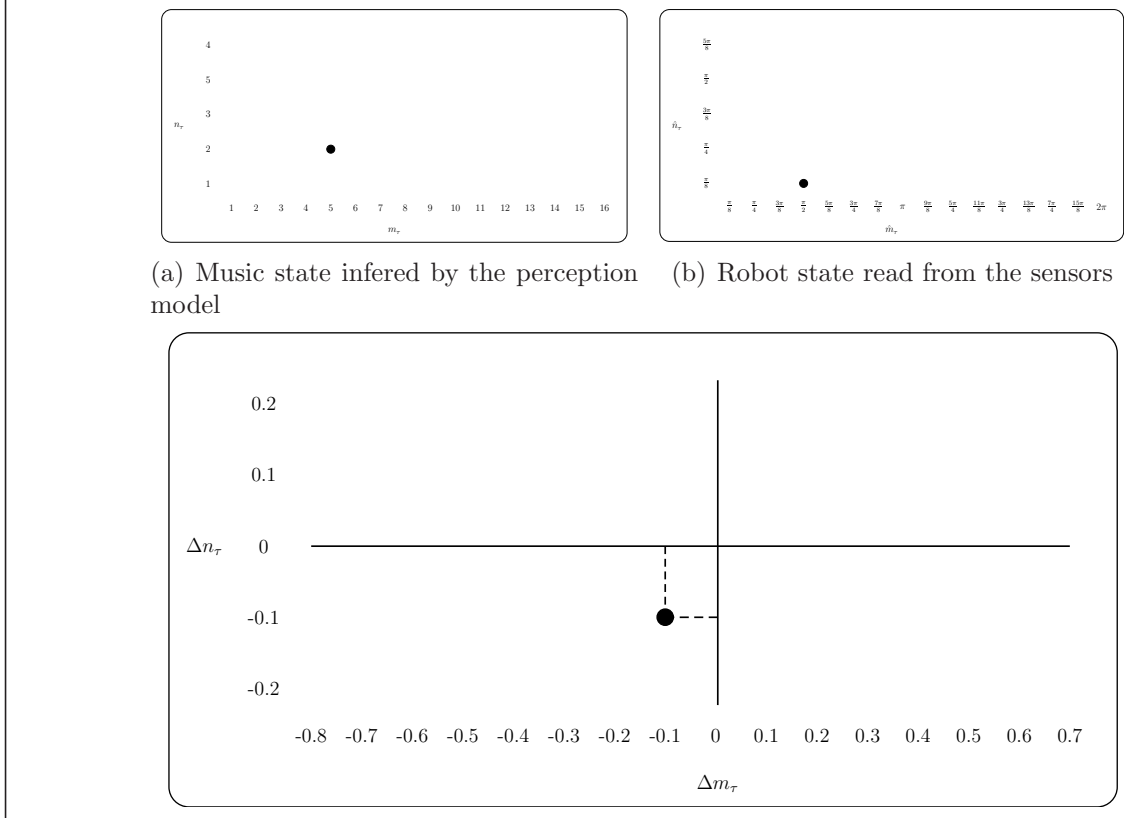
Formally, at each discrete time step τ, we represent the robot state by the motors angular position $\hat{m}_\tau \in [0, 2\pi)$ and angular velocity $\hat{n}_\tau > 0$. In our case, we assume these quantities are observed exactly without noise. Then, the robot has to determine the control action u_τ, which corresponds to an angular acceleration/deceleration value of its motor.

For correctly following the music, our main goal is to keep the relative distance between the observed performer state as in Figure 4a and the robot state as in Figure 4b. Here, states of the robot and music correspond to points on a two-dimensional space of velocity and bar position values. We can visualize the state space symbolically the difference between these states as in Figure 4c.

Hence, we can model the problem as a tracking problem that aims to keep the differences between the perceived tempo and the sensors values close to zero. Therefore, we define a new control state s_τ as,

$$s_\tau = \begin{bmatrix} \Delta m_\tau \\ \Delta n_\tau \end{bmatrix} \tag{13}$$

$$\Delta m_\tau = \frac{\hat{m}_\tau}{2\pi} - \frac{m_\tau}{M} \tag{14}$$



(a) Music state infered by the perception model

(b) Robot state read from the sensors

(c) Control state calculated as the normalized difference of the robot and music states

Figure 4 Illustration of the position and the velocity states.

$$\Delta n_\tau = \frac{\hat{n}_\tau}{2\pi} - \frac{n_\tau}{M} \tag{15}$$

Intuitively, the control state represents the drift of the robot relative to the performer; the goal of the controller will be to force the control state toward zero.

At each time step τ, the new bar position difference between the robot and the music Δm_τ is the sum of the previous bar position difference $\Delta m_{\tau-1}$ and the previous difference in velocity $\Delta n_{\tau-1}$. Additionally, the difference in velocity n_τ can only be affected by the acceleration of the robot motor u_τ. Hence, the transition model is explicitly formulated as follows,

$$s_{\tau+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} s_\tau + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_\tau + \varepsilon_\tau \tag{16}$$

where $u_\tau \in \mathbb{R}$ is the control signal to accelerate the motor and ε_τ is the zero-mean transition noise with Σ_A covariance. Here, the first coordinate of s_τ give the amount of difference in the score position of the performer and the robot.

For example, consider a case where the robot is lagging behind, so $\Delta m_\tau < 0$. If the velocity difference Δn_τ is also negative, i.e., the robot is "slower", then in subsequent time steps, the difference will grow in magnitude and the robot would lag further behind.

We write the model as a general linear dynamic system, where we define the transition matrix

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

and the control matrix $B = [0, 1]^{\mathrm{T}}$ to get

$$s_{\tau+1} = As_\tau + Bu_\tau + \varepsilon_\tau \tag{17}$$

To complete our control model, we need to specify an appropriate cost function. While one can contemplate various attractive choices, due to computational issues, we constrain ourselves to the quadratic case. The cost function should capture two aspects. The first one is the amount of difference in the score position. Explicitly, we do not care too much if the tempo is off as long as the robot can reproduce the correct timing of the beats. Hence, in the cost function, we only take the position difference into account. The second aspect is the smoothness of velocity changes. If abrupt changes in velocity are

allowed, the resulting performance would not sound realistic. Therefore, we also introduce a penalty on large control changes.

The following cost function represents both aspects described in the previous paragraph:

$$C_\tau(s_\tau,\ u_\tau) = \Delta m_\tau^2 + \kappa u_\tau^2 \tag{18}$$

where $\kappa \in \mathbb{R}^+$ is a penalty parameter to penalize large magnitude control signals.

In order to keep the representation standard, the quadratic cost function can also be shown in the matrix formulation as,

$$C_\tau(s_\tau,\ u_\tau) = s_\tau^T Q s_\tau + u_\tau^T R u_\tau \tag{19}$$

with explicit values, $R = \kappa$ and

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \tag{20}$$

Hence, after defining the corresponding linear dynamic system, the aim of the controller is to determine the optimal control signal, namely the acceleration of the robot motor u_τ given the transition and the control matrices and the cost function.

3.2 Linear-quadratic optimal control

In contrast to the general stochastic optimal control problems defined for general Markov decision processes (MDPs), linear systems with quadratic costs have an analytical solution.

When the transition model is written as in Equation 17, the cost function is defined as,

$$\begin{aligned} C_\tau(s_\tau,\ u_\tau) &= s_\tau^T Q s_\tau + u_\tau^T R u_\tau \quad \tau = 0, 1, \ \ldots,\ T-1 \\ C_T(s_T,\ u_T) &= s_T^T Q s_T \end{aligned} \tag{21}$$

the optimal control u_τ^* can be explicitly calculated for each state s_τ in the form of Bertsekas [24],

$$u^*(s_\tau) = L^* s_\tau \tag{22}$$

where gain matrix L^* is defined as,

$$L^* = -\left(B^T K^* B + R\right)^{-1} B^T K^* A \tag{23}$$

Here, K^* is the converged value of the recursively defined discrete-time Riccati equations,

$$\begin{aligned} K_t &= A^T\left(K_{t-1} - K_{t-1}B\left(B^T K_{t-1} B + R_t\right)^{-1} B^T K_{t-1}\right)A + Q \\ K_0 &= Q \end{aligned} \tag{24}$$

for stationary transition matrix A, control maxtrix B and state cost matrix Q.

Thus, in order to calculate the gain matrix L^*, a fixed-point iteration method with an initial point of $K_0 = Q$ is used to find the converged K value of $K^* = \lim_{t \to \infty} K_t$.

Finally, the control optimal action u_τ^* can be determined real-time simply by a vector multiplication at each time step τ. Choosing the control action $u_\tau = u_\tau^*$, Figure 5 shows an example of a simulated system.

3.3 Imperfect knowledge case

In the previous section, both perceived and sensor values are assumed to be true and noise free. However, possible errors of the perception module and noise of the sensors can be modeled as an uncertainty over the states. Actually, the perception module already infers a probability density over possible tempi and score positions. So, instead of a single point value, we can have a probability distribution as our belief state. However, this would bring us out of the framework of the linear-quadratic control into the more complicated general case of partially observed Markov decision processes (POMDPs) [24].

Fortunately, in the linear-quadratic Gaussian case, i.e., where the system is linear and the errors of the sensors and perception model are assumed to be Gaussian, the optimal control can still be calculated very similarly to the previous case as in Equation 22, by merely replacing s_τ with its expected value,

$$u^*(s_\tau) = L^* E[s_\tau]. \tag{25}$$

This expectation is with respect to the filtering density of s_τ . Since the system still behaves as a linear dynamical system due to the linear-quadratic Gaussian case assumption, this filtering density can be calculated in closed form using the Kalman filter [24].

In the sequel, we will denote this expectation as $E[s_\tau] = \mu_\tau$. In order to calculate the mean μ_τ, perceived values m_τ , n_τ and the sensor values $\hat{m}_\tau$, $\hat{n}_\tau$ are considered as the observations. Explicitly, we define the observation vector

$$\gamma_\tau = \begin{pmatrix} \Delta m_\tau \\ \Delta n_\tau \end{pmatrix} \tag{26}$$

Here, we assume the observation model

$$\gamma_\tau = s_\tau + \varepsilon_O \tag{27}$$

where ε_O is a zero-mean Gaussian noise with observation covariance matrix Σ_O which can be explicitly calculated as the weighted sum of the covariances of the perception model and the sensor noise as,

$$\Sigma_O = \frac{\Sigma_{\text{perception}}}{M^2} + \frac{\Sigma_{\text{robot}}}{(2\pi)^2} \tag{28}$$

where $\Sigma_{\text{perception}}$ is the estimated covariance of the tempo and position values inferred by the perception module by moment matching and Σ_{robot} is the covariance of the sensor noises specific to the actuators.

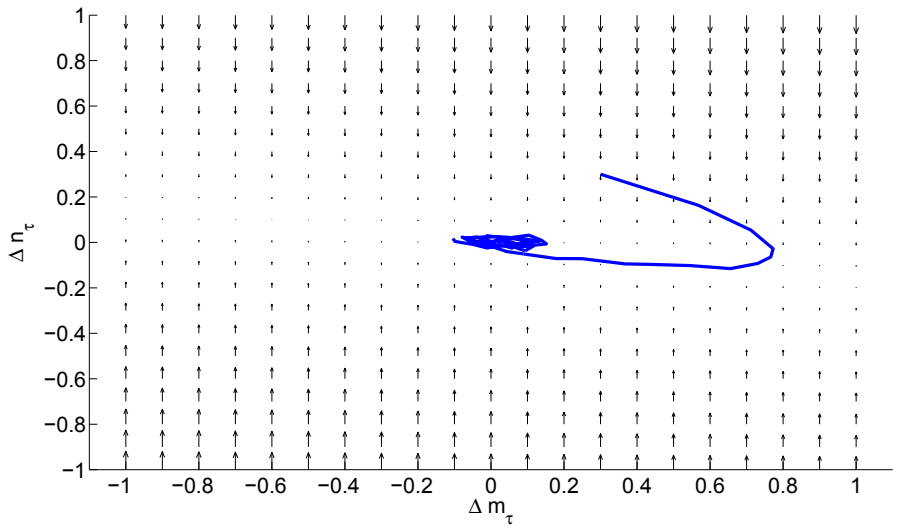


Figure 5 A trajectory with using optimal control u_τ for $\kappa = 150$. Here, the arrows denote the magnitude and the direction of the optimal control action u_τ as a function of the state. Since the control action is defined as acceleration or deceleration only, actions can only affect the velocity. Here, the robot was initially both faster, and its position was ahead of the performer. Hence, using the corresponding optimal control action, it tends to decelerate. However, it cannot directly catch the performer, since both the deceleration would affect the position in time, and there is also an associated penalty with large controls. Hence, the tempo cannot change quickly and the robot follows a non-trivial trajectory until convergence.

Given the model parameters, the expectation μ_τ is calculated at each time step by the Kalman filter.

$$\begin{aligned}\mu_\tau &= A\mu_{\tau-1} + G_\tau(\gamma_\tau - A\mu_{\tau-1})\\ \Sigma_\tau &= P_{\tau-1} - G_\tau P_{\tau-1}\end{aligned} \tag{29}$$

with initial values of,

$$\begin{aligned}\mu_0 &= \gamma_0\\ P_0 &= \Sigma_A\end{aligned} \tag{30}$$

Here, Σ_A is the variance of the transition noise and A is the transition matrix defined in Equation 17, G_τ is Kalman gain matrix and P_τ is the prediction variance defined as,

$$\begin{aligned}P_\tau &= A\Sigma_{\tau-1}A^T + \Sigma_A\\ G_\tau &= P_{\tau-1}(P_{\tau-1} + \Sigma_O)^{-1}\end{aligned} \tag{31}$$

4 Simulation results

Before implementing the whole system, we have evaluated our perception and the control models via several simulation scenarios. We have first evaluated the perception model on different parameter and problem settings, and then simulated the robot itself in order to evaluate the performance of both models and the synchronization level between them. At the end, we combine the Lego robot with the perception module and evaluate their joint performance.

4.1 Simulation of the perception model

In order to understand the effectiveness and the limitations of the perception model, we have conducted several experiments by simulating realistic scenarios. In our experiments, we generated the training and the testing data by using a MIDI synthesizer. We first trained the templates offline, and then, we tested our model by utilizing the previously learned templates.

At the training step, we run the EM algorithm which we described in Section 2.3, in order to estimate the spectral templates. For each acoustic event, we use a short isolated recording where the acoustic events consist of the claves hit, the conga hit (that is supposed to be produced by the robot itself), and silence. We also use templates in order to handle the polyphony in the music.

In the first experiment, we tested the model with a monophonic claves sound, where the son clave is played. At the beginning of the test file, the clave is played in medium tempo, where the tempo is increased rapidly in a couple of bars. In this particular example, we set M = 640, N = 35, R = 3, F = 513, p_n = 0:01, and the window length = 1,024 samples under 44.1 kHz sampling rate. With this parameter setting, the size of the transition matrix (see "Appendix A") becomes 67;200 × 67,200; however, only 0:87% of this matrix is non-zero. Therefore, by using sparse matrices, exact inference is still viable. As shown in Figure 6, the model captures the slight tempo change in the test file.

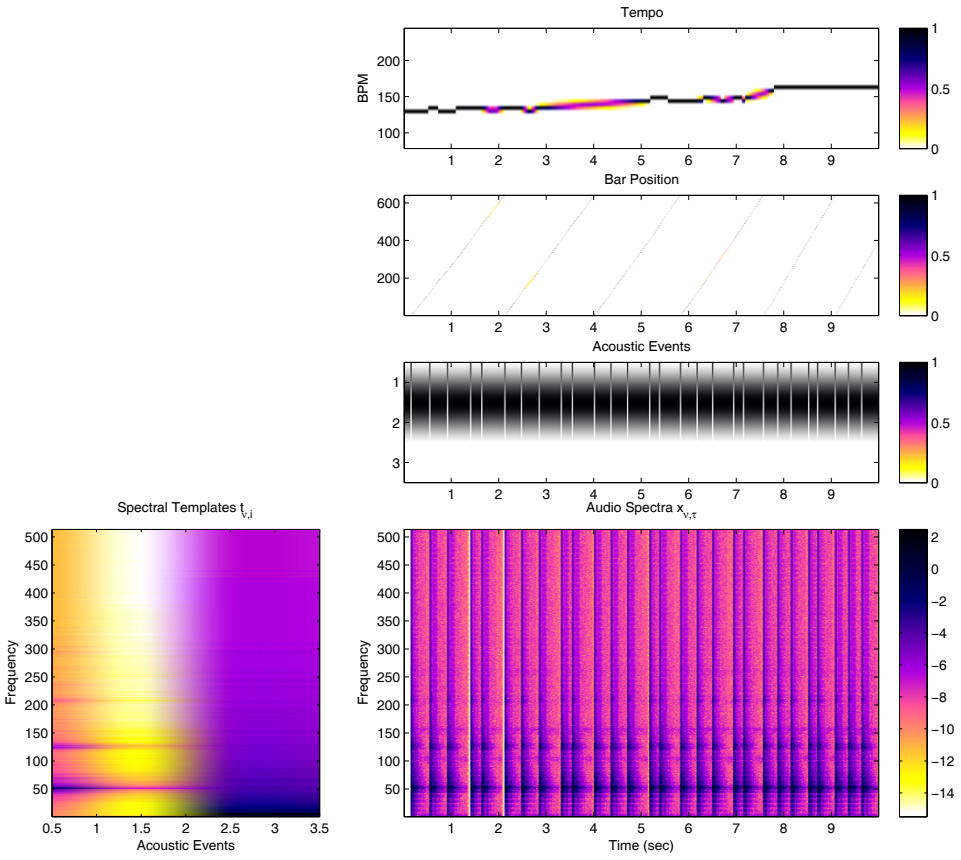


Figure 6 The performance of the perception model for a monophonic claves sound. In the leftmost figure, the spectral templates of the acoustic events are shown. These events consist of {1:claves hit, 2:silence, and 3:conga hit}. The topmost three figures illustrate the smoothing distribution of the velocity variables n_τ , bar position variables m_τ , and the acoustic event indicator variables r_τ . It can be observed that the model correctly captures the tempo change in the audio. Besides, the model correctly detects the claves hits as well, where it does not detect any false conga hits.

The *smoothing distribution*, which is defined as $p(n_\tau , m_\tau , r_\tau \mid x_{1:F,1:T})$, needs all the audio data to be accumulated. Since we are interested in online inference, we cannot use this quantity. Instead, we need to compute the *filtering distribution* $p(n_\tau , m_\tau , r_\tau \mid x_{1:F,1:\tau})$ or we can compute the *fixed-lag smoothing distribution* $p(n_\tau , m_\tau , r_\tau \mid x_{1:F,1:\tau+1})$ in order to have smoother estimates by introducing a fixed amount of latency (see "Appendix A" for details). Figure 7 shows the filtering, smoothing, and the fixed-lag smoothing distributions of the bar position, and the velocity variables provided the same audio data as in Figure 6.

In our second experiment, we evaluated the perception model on a polyphonic texture, where the sounds of the conga and the other instruments (brass section, synths, bass, etc.) are introduced. In order to deal with the polyphony, we trained spectral templates by using a polyphonic recording which does not include the claves and conga sound. In this experiment, apart from the spectral templates that are used in the previous experiment, we trained two more spectral templates by using the polyphonic recording that is going to be played during the robotic performance. Figure 8 visualizes the performance of the perception model on polyphonic audio. The parameter setting is the same as the first experiment described above, except in this example we set $N = 40$ and $R = 5$. It can be observed that the model performs sufficiently good enough for polyphonic cases. Besides, despite the fact that the model cannot detect some of the claves hits, it can still successfully track the tempo and the bar position.

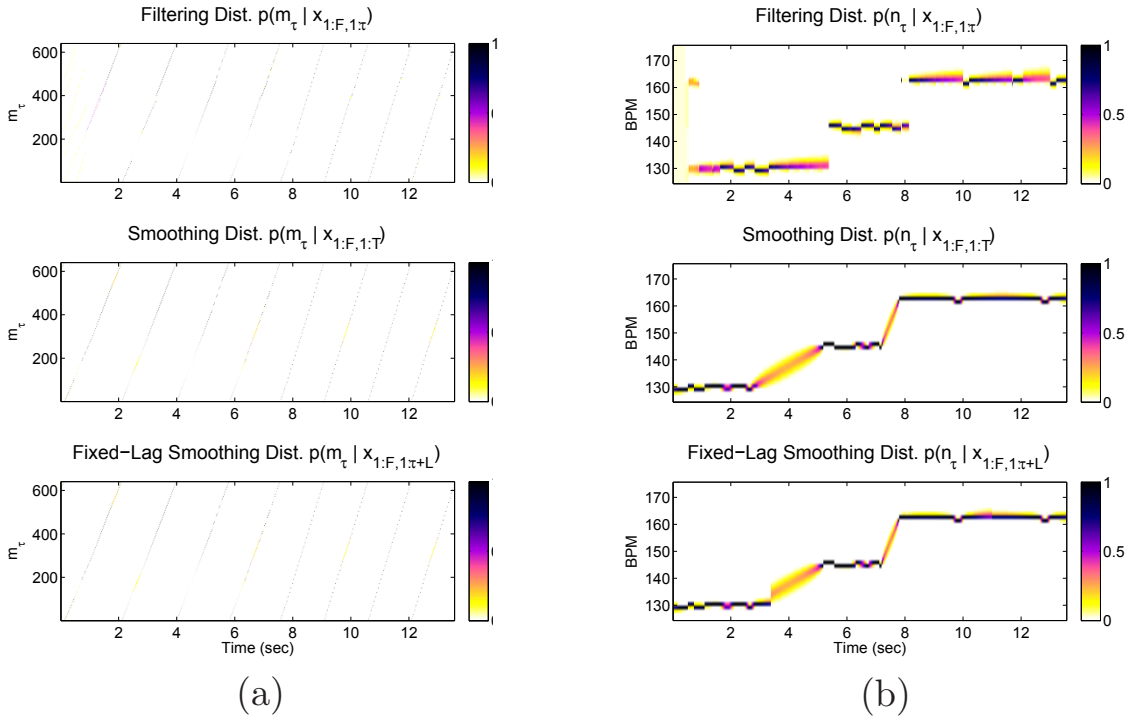


Figure 7 The filtering, smoothing, and the fixed-lag smoothing distributions of (a) the bar position variables m_τ and (b) the velocity variables n_τ . The lag in the fixed-lag smoothing distribution is selected to be 2 s (i.e., lagging one bar behind in 120 beats per minute). It can be observed that introducing a certain amount of lag yields smoother estimates and at the same time introduces a fixed amount of latency. Note that this experiment contains a dramatic tempo change where the tempo is increased by 40 BPMs in approximately 8 s.

4.2 Simulation of the robot

In this section, we wish to evaluate the convergence properties of the model under different parameter settings. In particular, we want to evaluate the effect of the perception estimates over the control model. Therefore, we have simulated a synthetic system where the robot follows the model described in Equation 17. Moreover, we simulate a conga hit whenever the state reaches to a predefined position as in Figure 9, and both signals from the clave and conga are mixed and fed back into the perception module, to simulate a realistic scenario. Before describing the results, we identify and propose solutions to some technicalities.

4.2.1 Practical issues

Due to the modulo operation of the bar position representation, using a simple subtraction operation causes irregularities at boundaries. Such as, when robot senses a bar position close to the end of a bar and the perception modules infers a bar position at the beginning of the next bar, the bar difference Δm_τ would be calculated close to 1 and the robot would tend to decelerate heavily. But, as soon as robot advances to the next bar, the difference becomes closer to 0. However, this time robot would have already slowed down greatly and would need to accelerate in order to get back on track. In order to circumvent this obstacle, a modular difference operation is defined that would return the smallest difference in magnitude,

$$\Delta m_\tau = \frac{\hat{m}_\tau}{2\pi} - \frac{m_\tau}{M} + b_\tau \tag{32}$$

where b_τ , namely bar difference between the robot and the perception module, was defined as,

$$b_\tau = \underset{b_\tau \in \{-1,0,1\}}{\arg\min} \left(\frac{\hat{m}_\tau}{2\pi} - \frac{m_\tau}{M} + b_\tau \right)^2 . \tag{33}$$

Additionally, even though the optimal control u_τ could be in $\mathbb{R}^+$, due to the physical properties of the robot, it is actually in a bounded set such as $[0, u_{\max}]$ during the experiments with robot. Hence, its value is truncated when working with the robot in order to keep it in the constrained set. However, while this violates our theoretical assumptions, the simulations are not affected from this non-linearity.

4.2.2 Results

In the first experiment, we illustrate the effect of the action costs on the convergence by testing different values of κ. First, κ is chosen as 0:1 to see the behavior of the system with low action costs. During the

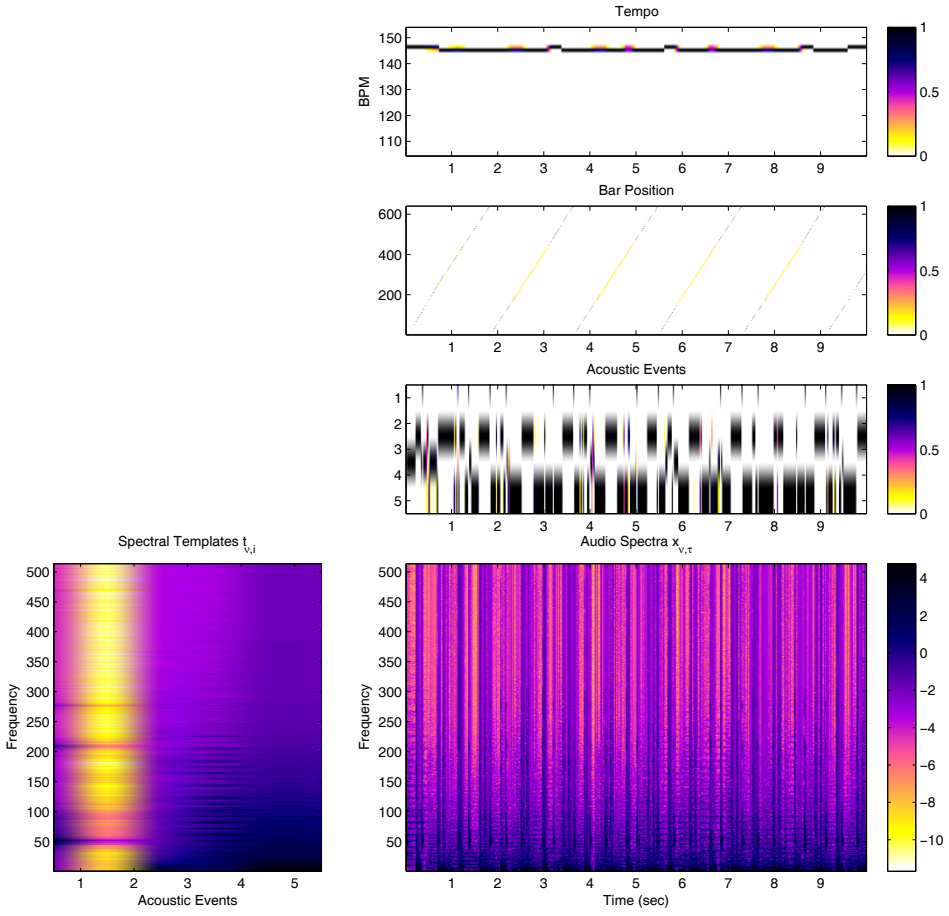


Figure 8 The performance of the perception model on polyphonic audio. In this experiment, the acoustic events consist of {1:claves hit, 2: silence, 3:conga hit, 4: polyphonic texture1, 5: polyphonic texture2}.

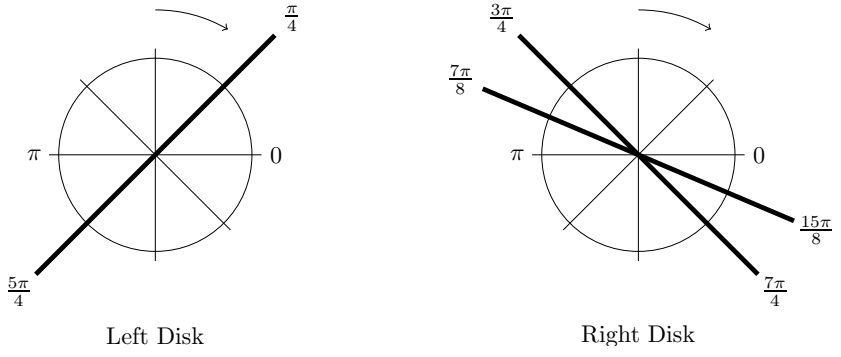


Figure 9 Robot's disks. One complete cycle of a disk completes a one 4/4 bar in the music. The disks are rotated with the same speed over the congas. The positions of the sticks are adjusted according to the positions of the conga hits specified by the sheet music in Figure 16.

simulation, the robot managed to track the bar position as expected as in Figure 10a. However, while doing so, it did not track the velocity, but instead, it fluctuated around its actual value as shown in Figure 10b.

In the following experiment, while keeping $\kappa = 0.1$, the cost function is chosen as,

$$C_\tau(s_\tau,\ u_\tau) = s'_\tau \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} s_\tau + u'_\tau[\kappa]u_\tau \tag{34}$$

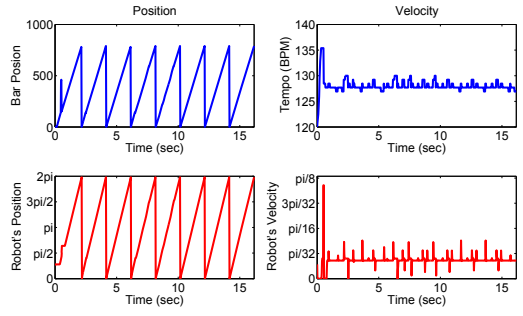
in order to make the robot explicitly track velocity in addition to bar position. However, as in Figure 10c and 10d it was easily affected by the perception module errors and fluctuate a lot before converging. This behavior mainly occurs because the initial velocity of the robot is zero and the robot tends to accelerate quickly in order to track the tempo of the music. However, with this rapid increase in the velocity, its bar position gets ahead of the bar position of the music. As a response the controller would decelerate, and this would cause the fluctuating behavior until the robot reaches a stable tracking position.

In order to get smooth changes in velocity, κ is chosen larger ($\kappa = 150$) to penalize large magnitude controls. In this setting, in addition to explicit tracking of bar position, robot also implicitly tracked the velocity without making big jumps as in Figure 11. In addition to good tracking results, the control module was also more robust against the possible errors of the perception module. As seen in Figure 12, even the perception module made a significant estimation error in the beginning of the experiment, the controller module was only slightly affected by this error and kept on following the correct track with a small error.

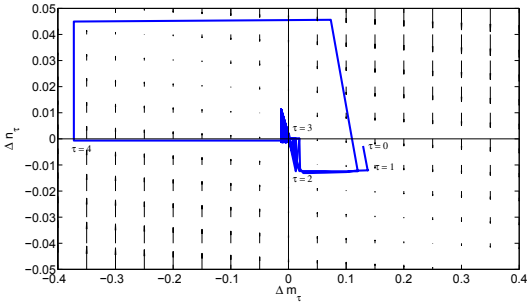
As a general conclusion about the control module, it could not track the performer in the first bar of the songs, because the estimations of the perception module are not yet accurate, and the initial position of the robot is arbitrary. However, as soon as the second bar starts, control state, expected normalized difference between the robot state and the music state, starts to converge to the origin.

Also note that, when κ is chosen close to 0, velocity values of the robot tend to oscillate a lot. Even sometimes they became 0 as in Figure 10a and 10c. This means that the robot has to stop in order to wait the performer because of its previous actions with high magnitudes.

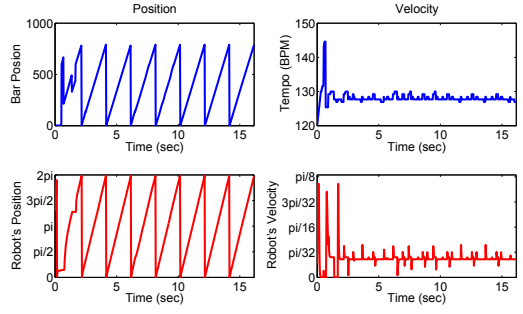
In the experiments, we observe that the simulated system is able to converge quickly in a variety of parameter



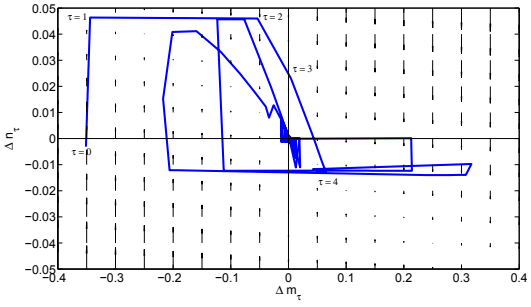
(a) Simulation results for $\kappa = 0.1$



(b) Trajectory of control state for $\kappa = 0.1$



(c) Simulation results for $\kappa = 0.1$ with velocity difference cost



(d) Trajectory of control state for $\kappa = 0.1$ with velocity difference cost

Figure 10 Simulation results for $\kappa = 0.1$. Plotted in red, the robot aims to track the position inferred by the perception module plotted in blue.

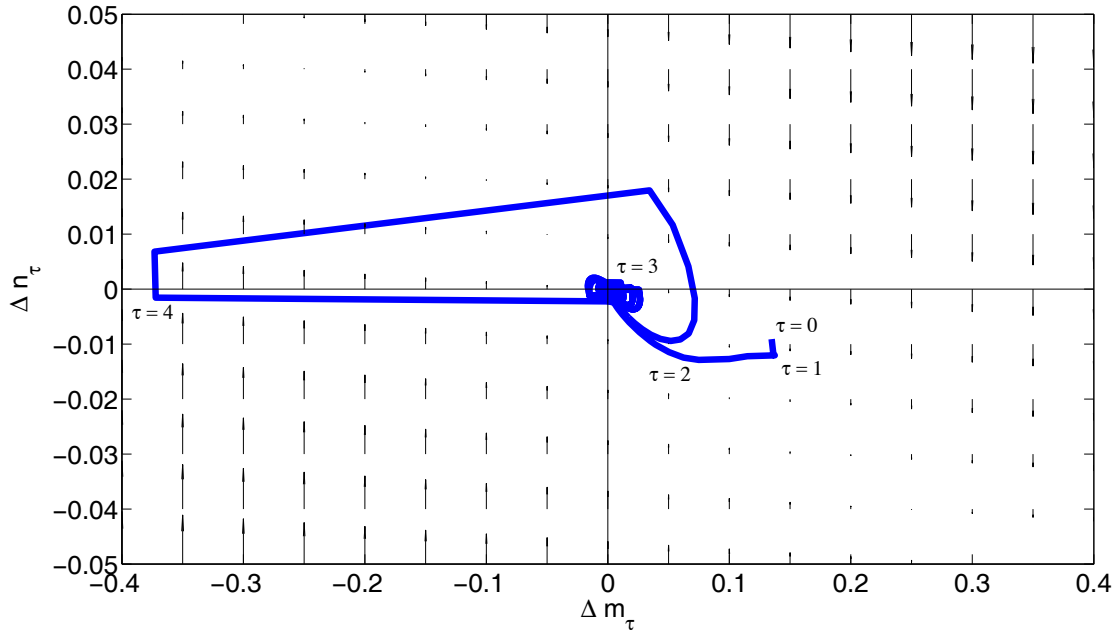


Figure 11 Trajectory of control state for κ = 150.

settings, as can be seen from control state diagrams. We omit quantitative results for the synthetic model at this stage and provide those for the Lego robot. In this final experiment, we combine the Lego robot with the perception module and run an experiment with a monophonic claves example with steady tempo. Here, we estimate the tempo and score position and try to synchronize the robot via optimal control signals. We also compare the effects of different cost functions provided that the clave is played in steady tempo, and the other parameters are selected to be similar to the ones that are described in synthetic data experiments. While

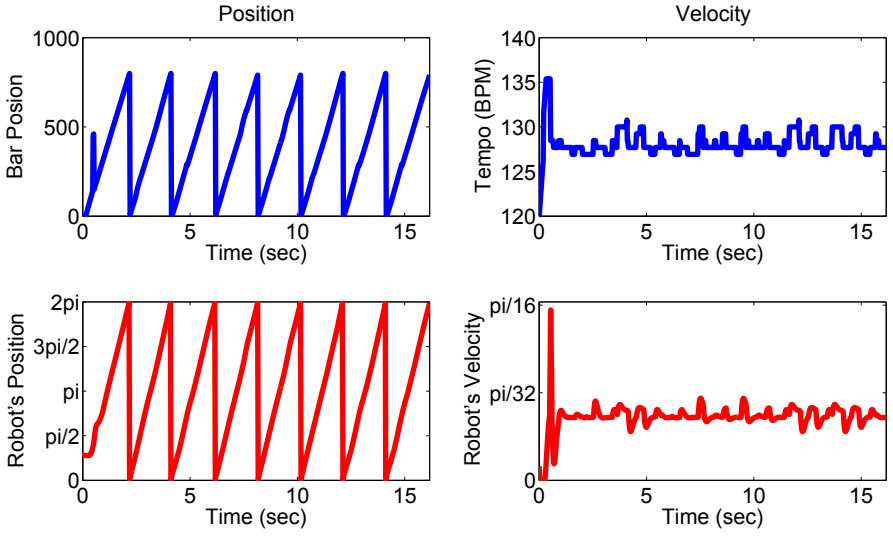


Figure 12 Simulation results for κ = 150.

perceptually more relevant measures can be found, for simplicity, we just monitor and report the mean square error.

In Figure 13a(b) shown are the average difference between the position (velocity) of music and the position (velocity) of the robot. In these experiments, we tried two different cost matrices

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad Q^{\text{pos}} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}. \tag{35}$$

Here, Q penalizes both the position and velocity error, where Q^{pos} penalizes only the position. The results seem to confirm our intuition: the control cost parameter κ needs to be chosen carefully to tradeoff elasticity versus rigidity. The figures visualize the corresponding control behaviors for the three different parameter regimes: converging with early fluctuations, close-to-optimal converging and converging slowly, respectively.

We also observe that the cost function taking into account only the score position difference is competitive generally. Considering the tempo estimate Δn_τ does not significantly improve the tracking performance other than the extremely small chosen $\kappa < 1$ which actually is not an appropriate choice for κ.

5 Experiments with a Lego robot

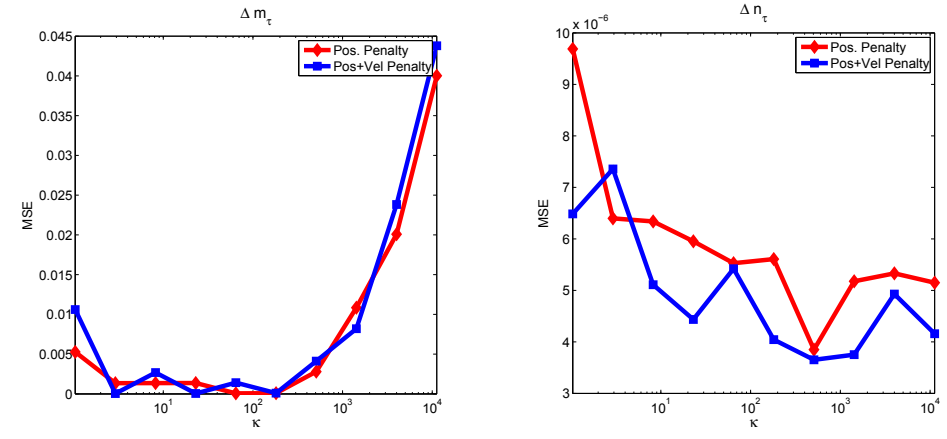
In this section, we describe a prototype system for musical interaction. The system is composed of a human claves player, a robot conga player, and a central computer as shown in Figure 14. The central computer listens to the polyphonic music played by all parties and jointly infers the tempo, and bar position, and the acoustic event. We will describe this quantities in the following section. The main goal of the system is to illustrate the feasibility of coupling listening (probabilistic inference) with taking actions (optimal control).

Since the microcontroller used on the robot is not powerful enough to run the perception module, the perception module runs on the central computer. The perception module sends the tempo and bar position information to the robot through a Bluetooth connection. On the other hand, the control module runs on the robot by taking into account its internal motor speed and position sensors and the tempo and bar position information. The central computer also controls a MIDI synthesizer that plays the other instrumental parts upon the rhythm.

5.1 The robot

The conga player robot is designed with Lego Mindstorm NXT programmable robotics kit. The kit includes a 48-MHz, 32-bits microcontroller with 64 KB memory. The controller is capable of driving 3 servo motors and 4 sensors of different kinds. The controller provides a USB and a Bluetooth communication interface.

The robot plays the congas by hitting them with sticks attached to rotating disks as shown in Figure 15. The disks are rotated by a single servo motor, attached to another motor which adjusts the distance between the congas and the sticks at the beginning of the experiment. Once this distance calibration is done (with the help of the human supervisor), the motor locks in its final position, and disks start to rotate to catch the tempo of the music. Although it looks more natural, we did not choose to build a robot with arms hitting the



(a) MSE of position estimates (b) MSE of velocity estimates

Figure 13 Mean-squared errors as a function of κ. The blue (square) and red (diamond) correspond for the cost matrices Q and Q^{pos}, respectively as defined in Equation 35.

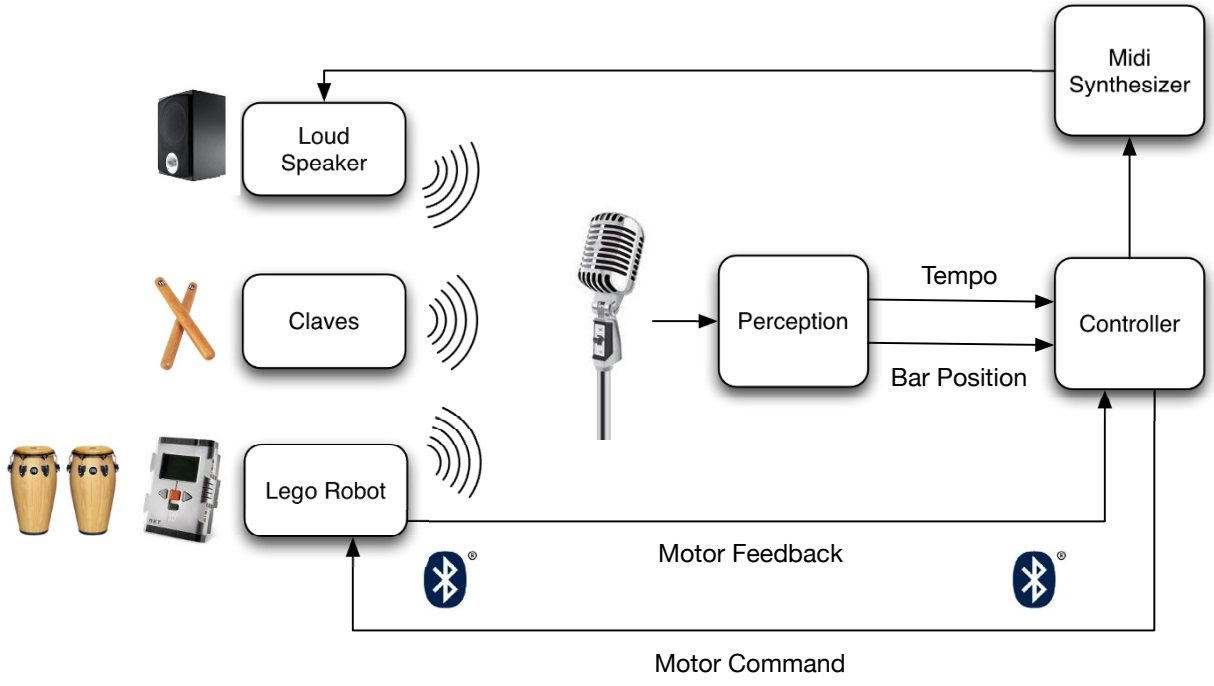


Figure 14 The overall system. The claves are played by a human performer. The tempo and bar position of the claves rhythm is perceived by a central computer, and the song, accompanying the rhythm, is synthesized by a MIDI synthesizer. The central computer also sends the tempo and position information to a conga playing Lego robot.

Figure 15 The Lego robot. Disks are attached on the same spindle, which is rotated by one servo motor. The other servo helps to adjust the distance between the sticks and the congas.

congas with drum sticks because the Lego kits are not appropriate to build robust and precisely controllable robotics arms,

The rhythm to be played by the robot is given in Figure 16. The robot is supposed to hit the left conga at 3rd and 11th, and the right conga at 7th, 8th, 15th and 16th sixteenth beats of the bar. In order to play this rhythm by constantly rotating disks, the rhythm must be hardcoded on the disks. For each conga, we designed a disk with sticks attached in appropriate positions such that each stick corresponds to a conga hit as shown in Figure 9. As the disks rotate, the sticks hit the congas at the time instances specified in the sheet music.

5.2 Evaluation of the system

We evaluated the real-time performance of our robot controller by feeding the tempo and score position estimates directly from the listening module. In the first experiment, we generated synthetic data that simulate a rhythm starting at a tempo of 60 bpm; initially accelerating followed by a ritardando. These data, without any observation noise, are sent to the robot in real time; e.g., the bar position and velocity values are sent in every 23 ms. The controller algorithm is run on the robot. While the robot rotates, we monitor its tachometer as an accurate estimate of its position and compare it with target bar position.

We observe that the robot successfully followed the rhythm as shown in Figure 17. In the second experiment we used the same setup but this time the output of the tempo tracker is send to the robot as input. The response of the robot is given in Figure 18. The errors in tempo at the beginning of the sequence comes from the tracker's error in detecting the actual bar position.

The mean-squared errors for the bar position and velocity for the experiments are given in the Table 1. We see that the robot is able to follow the score position very accurately while there are relatively large fluctuations in the instantaneous tempo. Remember that in our cost function 21, we are not penalizing the tempo discrepancy but only errors in score position. We believe that such controlled fluctuations make the timing more realistic and human like.

6 Conclusions

In this paper, we have described a system for robotic interaction, especially useful for percussion performance that consists of a perception and a control module. The perception model is a hierarchical HMM that does online event detection and separation while the control module is based on linear-quadratic control. The combined system is able to track the tempo quite robustly and respond in real time in a flexible manner.

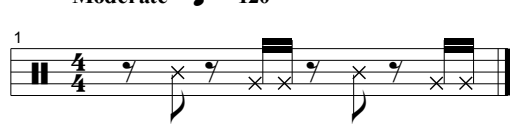


Figure 16 The conga rhythm to be played by the robot.

One important aspect of the approach is that it can be trained to distinguish between the performance sounds and the sounds generated by the robot itself. In synthetic and real experiments, the validity of the approach is illustrated. Besides, the model incorporates domain-specific knowledge and contributes to the area of *Computational Ethnomusicology* [25].

We also realized that and we will investigate another platform for such demonstrations and evaluations as a future work.

While our approach to tempo tracking is conceptually similar to the musical accompaniment systems reviewed earlier, our approach here has a notable novelty, where we formulate the robot performance as a linear quadratic control problem. This approach requires only a handful of parameters and seems to be particularly effective for generating realistic and human-like expressive musical performances, while being straightforward to implement. In some sense, we circumvent a precise statistical characterization of expressive timing deviations and still are able to generate a variety of rhythmic "feels" such as rushing or lagging quite easily. Such aspects of musical performance are hard to quantify objectively, but the reader is invited to visit our web page for audio examples and a video demonstration at http://www.cmpe.boun.edu.tr/~umut/orumbata/. As such, the approach has also potential to be useful in generating MIDI accompaniments that mimics a real human musicians behavior, control of complicated physical sound synthesis models or control of animated visual avatars.

Clearly, a Lego system is not solid enough to create convincing performances (including articulation and dynamics); however, our robot is more a proof of concept rather than a complete robotic performance system, and one could anticipate several improvements in the hardware design. One possible improvement for the perception model is to introduce different kinds of rhythmic patterns, i.e., clave patterns, to the perception model. This can be done by utilizing the rhythm indicator variable, which is presented in Whiteley et al. [8]. One other possible improvement is to introduce continuous state space for bar position and the velocity variables in order to have more accurate estimates and eliminate the computational needs of the large state space of the perception model. However, in that case exact inference will not be tractable, therefore, one should resort to approximate inference schemata, as discussed, for example in Whiteley et al. [26]. As for the control system, it is also possible to investigate

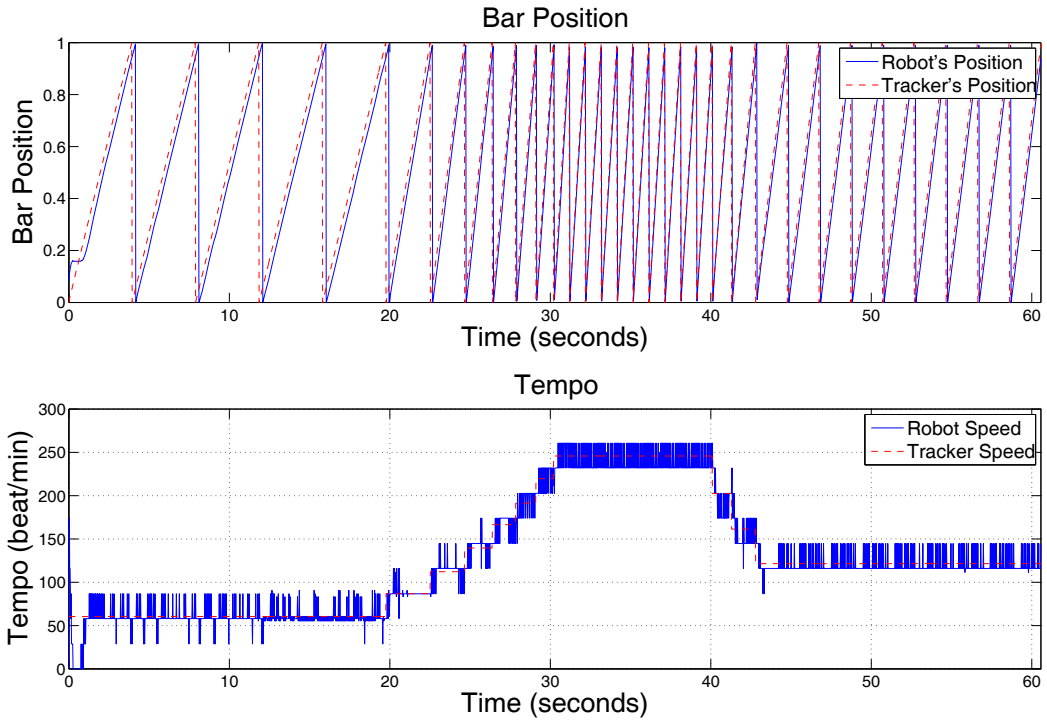


Figure 17 Robots performance with synthetic data.

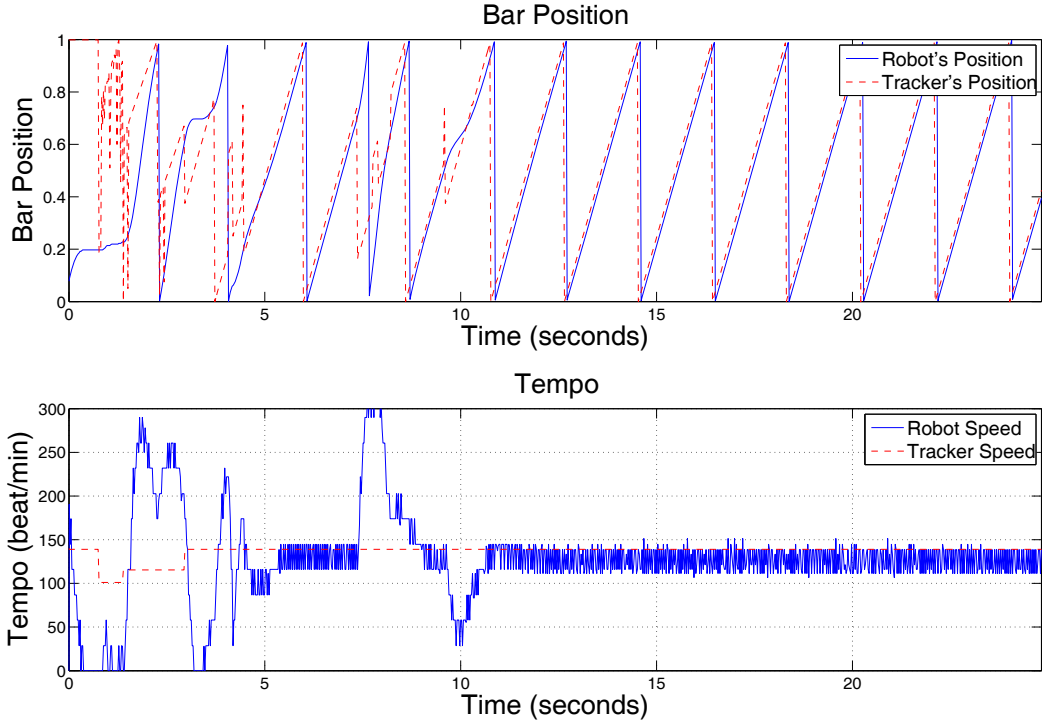


Figure 18 Robots performance with real data from beat tracker.

Table 1 Mean-squared errors for real-time robot performance

	Δ*m*	Δ*n*
Synthetic data	0.04	1.98×10^{-6}
Tempo-tracker data	0.08	2.83×10^{-5}

POMDP techniques to deal with more diverse cost functions or extend the set of actions for controlling, besides timing, other aspects of expressive performance such as articulation, intensity, or volume.

Appendix

A Inference in the perception model

Inference is a fundamental issue in probabilistic modeling where we ask the question "what can be the hidden variables as we have some observations?" [27]. For online processing, we are interested in the computation of the so-called filtering density: $p(n_\tau, m_\tau, r_\tau \mid x_{1:F,1:\tau})$, that reflects the information about the current state $\{n_\tau, m_\tau, r_\tau\}$ given all the observations so far $x_{1:F,1:\tau}$. The filtering density can be computed online, however the estimates that can be obtained from it are not necessarily very accurate as future observations are not accounted for.

An inherently better estimate can be obtained from the so-called fixed-lag smoothing density, if we can afford to wait a few steps more. In other words, in order to estimate $\{n_\tau, m_\tau, r_\tau\}$, if we accumulate L more observations, at time $\tau + L$, we can compute the distribution $p(n_\tau, m_\tau, r_\tau \mid x_{1:F,1:\tau+L})$ and estimate $\{n_\tau, m_\tau, r_\tau\}$ via:

$$\{n_\tau^*, m_\tau^*, r_\tau^*\} = \arg\max_{n_\tau m_\tau, r_\tau} p(n_{1:\tau+L}, m_{1:\tau+L}, r_{1:\tau+L} \mid x_{1:F,1:\tau+L}). \quad (36)$$

Here, L is a specified lag and it determines the trade off between the accuracy and the latency.

As a reference to compare against, we compute an inherently batch quantity: the most likely state trajectory given all the observations, the so-called the Viterbi path

$$\{n_{1:T}^*, m_{1:T}^*, r_{1:T}^*\} = \arg\max_{n_{1:T}, m_{1:T}, r_{1:T}} p(n_{1:T}, m_{1:T}, r_{1:T} \mid x_{1:F,1:T}). \quad (37)$$

This quantity requires that we accumulate all data before estimation and should give a high accuracy at the cost of very long latency.

Briefly, the goal of inference in the HMM is computing the filtering and the (fixed-lag) smoothing distributions and the (fixed-lag) Viterbi path. These quantities can be computed by the well-known forward-backward and the Viterbi algorithms.

Before going into details, we define the variable $\Psi_\tau \equiv [n_\tau, m_\tau, r_\tau]$, which encapsulates the state of the system at time frame τ. By introducing this variable, we reduce the number of latent variables to one, where we can write the transition model as follows:

$$\begin{aligned} p(\Psi_0) &= p(n_0)p(m_0)p(r_0), \\ p(\Psi_\tau \mid \Psi_{\tau-1}) &= p(n_\tau \mid n_{\tau-1})p(m_\tau \mid n_{\tau-1}, m_{\tau-1})p(r_\tau \mid n_{\tau-1}, m_{\tau-1}, r_{\tau-1}). \end{aligned} \quad (38)$$

Here $p(m_\tau \mid \cdot)$ is the degenerate probability distribution, which is defined in Equation 1. For practical purposes, the set of all possible states (in $Dn \times D_m \times D_r$) can be listed in a vector Ω and the state of the system at the time slice τ can be represented as $\Psi_\tau = \Omega(j)$, where $j \in \{1, 2, \ldots, (NMR)\}$. The transition matrix of the HMM, A can be constructed by using Equation 38, where

$$A(i, j) = p(\Psi_{\tau+1} = \Omega(i) \mid \Psi_\tau = \Omega(j)). \quad (39)$$

For big values of N, M, and R this matrix becomes extremely large, but sufficiently sparse so that making exact inference is viable.

Now, we can define the forward (α) and the backward (β) messages as follows:

$$\begin{aligned} \alpha_\tau(\Psi_\tau) &= p(\Psi_\tau, x_{1:F,1:\tau}), \\ \beta_\tau(\Psi_\tau) &= p(x_{1:F,\tau+1:T} \mid \Psi_\tau). \end{aligned} \quad (40)$$

We can compute these messages via the following recursions:

$$\begin{aligned} \alpha_\tau(\Psi_\tau) &= p(x_{1:F,\tau} \mid \Psi_\tau) \sum_{\Psi_{\tau-1}} p(\Psi_\tau \mid \Psi_{\tau-1}) \alpha_{\tau-1}(\Psi_{\tau-1}), \\ \beta_\tau(\Psi_\tau) &= \sum_{\Psi_{\tau+1}} p(\Psi_{\tau+1} \mid \Psi_\tau) p(x_{1:F,\tau+1} \mid \Psi_{\tau+1}) \beta_{\tau+1}(\Psi_{\tau+1}). \end{aligned} \quad (41)$$

Here, $\alpha_0(\Psi_0) = p(\Psi_0)$, $\beta_T(\Psi_T) = 1$ [28], and $p(x_{1:F,\tau} \mid \Psi_\tau) \equiv p(x_{1:F,\tau} \mid r_\tau)$. Once these messages are computed, the smoothing distribution can be computed easily by multiplying the forward and backward messages as

$$p(\Psi_\tau \mid x_{1:F,1:T}) \propto \alpha_\tau(\Psi_\tau)\beta_\tau(\Psi_\tau), \quad (42)$$

where $\propto$ denotes the proportionality up to a multiplicative constant. Besides, the Viterbi path is obtained by replacing the *summations* over r_τ by *maximization* in the forward recursion.

Acknowledgements

We are grateful to Prof. Levent Akin and the members of the AI lab for letting us to use their resources (lab space and Lego© robots) during this study. We also thank Antti Jylhä and Cumhur Erkut of the acoustics labs Aalto University, Finland for the fruitful discussions. We also want to thank Sabanc[3] University Music Club (Müzikus) for providing the percussion instruments. We would like to also thank Ömer Temel and Alper Güngörmüşler for their contributions in program development. We thank the reviewers for their constructive feedback. This work is partially funded by The Scientific and Technical Research Council of Turkey (TÜBİTAK) grant number 110E292, project "Bayesian matrix and tensor factorisations (BAYTEN)" and Boğaziçi University research fund BAP 5723. The work of Umut Şimşekli and Orhan Sönmez is supported by the Ph.D. scholarship (2211) from TÜBİTAK.

Authors' contributions
UŞ and ATC conceived and designed the perception model. OS and ATC carried out the control model. UŞ and BK implemented the Lego© robot. UŞ, OS, BK, and ATC wrote the paper. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Received: 16 April 2011 Accepted: 3 February 2012
Published: 3 February 2012

References

1. T Otsuka, K Nakadai, T Takahashi, T Ogata, HG Okuno, Real-time audio-to-score alignment using particle filter for coplayer music robots. EURASIP J Adv Signal Process. **2011**, 2:1–2:13 (2011)
2. A Kapur, A history of robotic musical instruments, in *International Computer Music Conference (ICMC)* (September 2005)
3. M Goto, Y Muraoka, Real-time beat tracking for drumless audio signals: chord change detection for musical decisions. Speech Commun. **27**(3-4), 311–335 (1999). doi:10.1016/S0167-6393(98)00076-4
4. T-h Kim, SI Park, SY Shin, Rhythmic-motion synthesis based on motion-beat analysis. in *ACM SIGGRAPH 2003 Papers* 392–401 (2003)
5. YE Kim, AM Batula, D Grunberg, DM Lofaro, J Oh, P Oh, Developing humanoids for musical interaction, in *International Conference on Intelligent Robots and Systems* (2010)
6. DM Lofaro, P Oh, J Oh, Y Kim, Interactive musical participation with humanoid robots through the use of novel musical tempo and beat tracking techniques in the absence of auditory cues. in *2010 10th IEEE-RAS International Conference on Humanoid Robots (Humanoids)* 436–441 (2010)
7. U Şimşekli, AT Cemgil, Probabilistic models for real-time acoustic event detection with application to pitch tracking. J New Music Res. **40**, 175–185 (2011). doi:10.1080/09298215.2011.573561
8. N Whiteley, AT Cemgil, SJ Godsill, Bayesian modelling of temporal structure in musical audio. in *Proceedings of International Conference on Music Information Retrieval* (2006)
9. K Yoshii, K Nakadai, T Torii, Y Hasegawa, H Tsujino, K Komatani, T Ogata, HG Okuno, in *IROS* 1743–1750 (2007)
10. K Murata, K Nakadai, K Yoshii, R Takeda, T Torii, HG Okuno, Y Hasegawa, H Tsujino, A robot uses its own microphone to synchronize its steps to musical beats while scatting and singing. IROS 2459–2464 (2008)
11. R Dannenberg, An on-line algorithm for real-time accompaniment. in *International Computer Music Conference* 193–198 (1984)
12. N Orio, An automatic accompanist based on hidden markov models, in in *Proceedings of the 7th Congress of the Italian Association for Artificial Intelligence on Advances in Artificial Intelligence*, (ser. AI*IA 01. London: Springer, 2001), pp. 64–69 http://portal.acm.org/citation.cfm?id=648152.751104
13. AT Cemgil, HJ Kappen, Monte carlo methods for tempo tracking and rhythm quantization. J Artif Intell Res. **18**, 45–81 (2003)
14. C Raphael, Music plus one and machine learning. in *International Conference on Machine Learning* 21–28 (2010)
15. T Völkel, J Abeßer, C Dittmar, H Großmann, Automatic genre classification of latin american music using characteristic rhythmic patterns, in *Proceedings of the 5th Audio Mostly Conference: A Conference on Interaction with Sound*, vol. 16. (ser. AM '10. New York, NY, USA: ACM, 2010), pp. 1–16:7 http://doi.acm.org/10.1145/1859799.1859815
16. M Wright, WA Schloss, G Tzanetakis, Analyzing afro-cuban rhythms using rotation-aware clave template matching with dynamic programming. in *ISMIR* 647–652 (2008)
17. U Şimşekli, *Bayesian methods for real-time pitch tracking*, (Master's thesis, Boğaziçi University, 2010)
18. U Şimşekli, AT Cemgil, A comparison of probabilistic models for online pitch tracking. in *Proceedings of the 7th Sound and Music Computing Conference (SMC)* (July 2010)
19. U Şimşekli, A Jylhä, C Erkut, AT Cemgil, Real-time recognition of percussive sounds by a model-based method. EURASIP J Adv Signal Process (in press) (2011)
20. C Févotte, N Bertin, J-L Durrieu, Nonnegative matrix factorization with the Itakura-Saito divergence. with application to music analysis. Neural Comput. **21**(3), 793–830 (2009). doi:10.1162/neco.2008.04-08-771
21. E Vincent, N Bertin, R Badeau, Harmonic and inharmonic nonnegative matrix factorization for polyphonic pitch transcription. in *ICASSP* (2008)
22. E Alpaydin, *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*, (Cambridge: The MIT Press, 2004)
23. C Fevotte, AT Cemgil, Nonnegative matrix factorisations as probabilistic inference in composite models. in *Proceedings of the 17th European Signal Processing Conference (EUSIPCO'09)* (2009)
24. D Bertsekas, *Dynamic Programming and Optimal Control*, (Belmont: Athena Scientific, 1995)
25. G Tzanetakis, A Kapur, WA Schloss, M Wright, Computational ethnomusicology. J Interdiscip Music Stud. **1**(2), 1–24 (2007)
26. N Whiteley, AT Cemgil, S Godsill, Sequential inference of rhythmic structure in musical audio. in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2007*. **4**, IV-1321–IV-1324 (2007)
27. O Cappé, E Moulines, T Ryden, *Inference in Hidden Markov Models (Springer Series in Statistics)*, (Secaucus: Springer, 2005)
28. D Barber, AT Cemgil, Graphical models for time series. IEEE Signal Process Mag Special Issue Graph. Models. **27**(27), 18–28 (2010)

doi:10.1186/1687-4722-2012-8
Cite this article as: Şimşekli *et al.*: **Combined perception and control for timing in robotic music performances.** *EURASIP Journal on Audio, Speech, and Music Processing* 2012 **2012**:8.