Predicting an Orchestral Conductor's Baton Movements Using Machine Learning

Donald G. Dansereau,* Nathan Brock,† and Jeremy R. Cooperstock**

*Australian Centre for Field Robotics
School of Aerospace, Mechanical and Mechatronic Engineering
Rose Street Building J04
The University of Sydney 2006 NSW
Sydney, Australia
d.dansereau@acfr.usyd.edu.au

†California Institute for Telecommunications and Information Technology
University of California at San Diego

**Centre for Intelligent Machines
McGill University
3480 University Street
Montreal, QC H3A 2A7, Canada
jer@cim.mcgill.ca

Abstract: Telematic musical performance, in which performers at two or more sites collaborate via networked audio and video, suffers significantly from latency. In the extreme case, performers at all sites slow to match their delayed counterparts, resulting in a steadily decreasing tempo. Introducing video of a conductor does not immediately solve the problem, as conductor video is also subjected to network latencies. This article lays the groundwork for an alternative approach to mitigating the effects of latency in distributed orchestral performances, based on generation of a predicted version of the conductor's baton trajectory. The prediction step is the most fundamental problem in this scheme, for which we propose the use of conventional machine learning techniques. Specifically, we demonstrate a particle filter and an extended Kalman filter that each track the location of the baton's tip and predict it multiple beats into the future; we compare these with a conventional feature-based method. We also describe a generic two-part framework that prescribes the incorporation of rehearsal data into a probabilistic model, which is then adapted during live performance. Finally, we suggest a framework and experimental methodology for establishing perceptually based metrics for predicted baton paths. Note that the perceptual efficacy of the presented methods requires experimental confirmation beyond the scope of this article.

A conductor is tasked with coordinating all the players in an orchestral performance, controlling the group like a sort of meta-instrument. Though first introduced for synchronization of players in a large performance, the conductor's role has evolved to include significant influence over how a piece is interpreted, with fine-grained control over sections of performers or even individuals, in dynamics, tempo, and articulation. Automated interpretation of the complex vocabulary of gestures utilized by a conductor is an ongoing problem, with previous work showing methods of tracking the interpretive content of gestures (Ilmonen and Takala 2005) and predicting the temporal location of an impending musical beat (Murphy 2004).

The present work is focused on establishing the viability of predicting the conductor's gestures multiple beats into the future, as relevant in the context of telematic performance, i.e., distributed musical performance over computer networks. In the case of distributed networked performance, as depicted in Figure 1, a major challenge is dealing with network lag between sites. It is well established that network latency can have serious deleterious effects on the quality of a performance (Willey 1990; Bartlette et al. 2006; Chafe, Cáceres, and Gurevich 2010). In the worst case, each musician will slow to match the other players across the network, leading to a steadily decreasing tempo as the performers attempt to synchronize.

Computer Music Journal, 37:2, pp. 28–45, Summer 2013
doi:10.1162/COMJ_a_00173

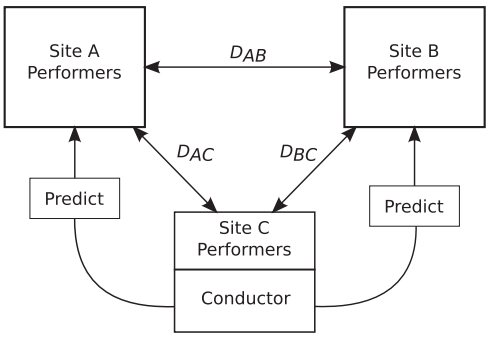


Figure 1. Using gesture prediction to mitigate delay in distributed orchestral performance: The delays D *normally experienced between sites are canceled, through prediction, for the conductor's gestures.*

Following from the similar rationale for introducing a conductor to synchronize the performance of copresent musicians, we might assume that this same approach would be beneficial in the distributed context. Network delay does not distinguish between audio and video data, however; both are subject to significant latencies, and thus the video of a remote conductor does not necessarily improve synchronization. A partially improved solution would be to provide all sites with video of the conductor's gestures simultaneously. Note that performers would continue to perceive their remote counterparts as lagging by the amount of network latency, but the hope is that they would rely on the conductor's gestures for synchronization, as in the co-localized case, eliminating problems with drifting tempo. In practice, musicians are more inclined to attend to the audio cues, but with effort, can focus on the conductor's lead (Olmos et al. 2009).

Synchronized conductor video can be created by using prediction to counteract the conductor-to-performer lag, as depicted by the *predict* paths in Figure 1. In this scenario, a motion capture system encodes the movements of the conductor's baton and arm, and a prediction system generates an estimated pose, with the prediction range set to counteract the exact network latency on the uplink paths. Once received, the estimated pose is used to render an animated baton and arm, and this rendering is blended, using techniques from computer graphics, with the (non-predicted) live feed of the rest of the conductor. It is hoped that, because the bulk of the time-critical information is associated with the baton's movements, the network lag between predicted and live elements will not prove unacceptably distracting. The result of this setup is that the performers perceive the conductor with zero delay, and the conductor perceives the performance delayed by only the one-way network latency associated with the return network path. Still more ambitious scenarios might introduce further prediction, counteracting even this one-way delay.

The most fundamental problem in combating latency in this way is in the prediction step, and the key contribution of this article is to establish the viability of performing such prediction using conventional machine learning techniques. Specifically, it makes use of particle filters for tracking and prediction of the baton tip's location, and lays out a generic two-part framework that prescribes the incorporation of rehearsal data into a probabilistic model, which is then adapted during live performance. We make no claims as to the optimality of particle filters for this purpose, and expect that other methods from machine learning would also be applicable. As an example, we include experimental confirmation that the extended Kalman filter (EKF) is capable of replacing the particle filter for tracking and prediction. Note that our primary concern is establishing the feasibility of machine learning techniques in general, as they provide a flexible and robust alternative to existing approaches. As such, we focus primarily on the particle filter for its ease of implementation and robustness.

To facilitate evaluation, the article also suggests a framework and experimental methodology for establishing perceptually based metrics for predicted baton paths. Taken together, these contributions lay the groundwork for mitigating latency in distributed orchestral performances using prediction of conductor's baton movements. It should be noted that the perceptual efficacy of these methods requires experimental confirmation beyond the scope of this article.

Background

Conductors follow a set of rules that both dictate the rough shape of the path taken by the tip of their baton and that prescribe ways of manipulating those shapes

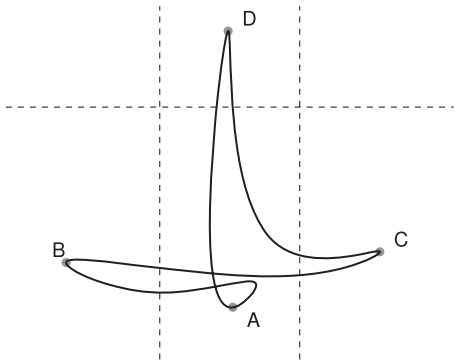


Figure 2. Feature locations shown on a generic 4/4 conducting gesture. Note that A, B, C, and D correspond to four easily tracked features of the gesture, and do not correspond exactly to beat locations.

in order to communicate intent to the performers. A typical conducting gesture for a 4/4 beat pattern (Rudolf 1995) is shown in Figure 2. The shape of the gesture, and the time spent at its extremities, communicate articulation (how smoothly a piece is played) and the scale of the gesture communicates dynamics (how loudly a performer should play). Translation of the gesture can be used to indicate that it is directed toward individual performers or sections of performers, and the rate at which the gesture is traversed communicates tempo, with individual beats indicated by vertical extrema. We refer to such communicative elements of the gestures as their *interpretive* or *perceptual* content.

Previous work has focused on tracking and interpretation of conductor's gestures (Morita, Hashimoto, and Ohteru 1991; Lee et al. 2006; Nijholt et al. 2008). Prediction has been concerned primarily with the timing of an impending beat (Murphy 2004) rather than with predicting the trajectory of the baton. Much of this previous work has relied on recognizing characteristic features of the conducting gesture, such as inflection points, changes of direction, and axis crossings. Although such recognition can yield good results, it relies on accurate placement of the features on the gesture, which can be sensitive to noise and ambiguity in the baton's path. Moreover, the feature extraction process implies that some or all of the baton's motion is ignored between features.

Other novel work has included the use of neural networks for interpreting hand gestures in music control (Modler and Myatt 2004) and for interpreting the emotional content of conducting gestures (Ilmonen and Takala 2005). Meanwhile, particle filters have been applied successfully to more general forms of gesture recognition (Black and Jepson 1998; Visell and Cooperstock 2007; Lee 2008). Similarly, effective prediction of quasi-periodic heartbeat signals using EKFs has been demonstrated (Yuen, Novotny, and Howe 2008), and hidden Markov models have been applied to gesture following and recognition (Bevilacqua et al. 2010; Kolesnik and Wanderley 2004). The EKF is a generalization of the Kalman filter, capable of coping with nonlinear system dynamics through the addition of a linearization step (Welch and Bishop 1995). For such tasks, the EKF is an attractive option because it is well understood and easy to implement, with reference implementations readily available.

This article seeks to utilize a particle filter for tracking and predicting baton trajectories. The challenges to doing this are formidable, and we therefore limit our scope to the prediction of two-dimensional (2-D) baton motion, and assume that this motion is available from a computer vision or motion capture system. We specifically ignore, for the moment, the problems associated with start up and long pauses. Once a gesture is underway, predicting the baton's path is feasible, but in the moments of stillness preceding a performance, or during long pauses, accurate prediction is difficult or impossible without additional information. A potential approach, in the context of combating network latency, is to begin a performance with no prediction in place, then slowly increase the prediction range to the desired level. Alternatively, the conductor may be asked to provide additional cues, for example a preparatory indicator prior to the first "real" gesture. This problem will require further experimentation to address fully.

Particle Filter

The particle filter performs tracking and prediction based on a Bayesian framework, modeling the posterior probability density of a system's state with a set of weighted state estimates (particles). For this work, we encode in a *state vector* a set of high-level properties of the conductor's gestures,

such as position and scale. A population of particles is generated, with each particle representing a hypothesis regarding the true value of the state vector. This population of particles is culled and resampled over time according to how well each one predicts certain directly observable variables—in our case, the position of the tip of the conductor's baton. This process is iterative, such that over time the particles converge to a population that does a good job of representing the true state of the system. If the system state changes, the particle population will also evolve to reflect this change, and as such, the particles effectively *track* the values in the state vector, without ever being able to observe them directly. The remainder of this section details how the particle filter is constructed and extended to effect prediction.

We implemented a classic sampling importance resampling filter, also know as a weighted bootstrap filter (Gordon, Salmond, and Smith 1993). Supposing $\mathbf{x}_k$ and $\mathbf{z}_k$ denote the system state and observation at time k, the probability distribution of the system state $\Pr(\mathbf{x}_k|\mathbf{z}_{0\ldots k})$ is estimated by a weighted particle set $\mathbf{X}_k = \{\mathbf{x}_k^i, q_k^i\}$, $i = 1, \ldots, N$, where N is the total number of particles, $\mathbf{x}_k^i$ is the hypothetical state of the system represented by particle i at time k, and q_k^i is the particle's weight.

An estimate of the system state is formed prior to each observation by propagating the previous state estimate forward via a motion model

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k) + \mathbf{w}_k \tag{1}$$

where $\mathbf{f}()$ is the system transition function and $\mathbf{w}$ is a random variable approximating the system noise. Observations are related to the state vector via the observation model

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k \tag{2}$$

where $\mathbf{z}_k$ is the observation, $\mathbf{v}$ is the observation noise, and $\mathbf{h}()$ is the measurement function, which encodes state as an observable quantity. When each new observation arrives, the particle set is re-weighted by evaluating the likelihood of each prior sample

$$q_k^i = \frac{\Pr(\mathbf{z}_k|\mathbf{x}_k^i)}{\sum_{i=1}^{N} \Pr(\mathbf{z}_k|\mathbf{x}_k^i)} \tag{3}$$

Finally, a new set of particles $\mathbf{X}_k^*$ is drawn from the set $\mathbf{X}_k$, such that the probability of finding a given particle in the resampled set is proportional to its weight, i.e., for any j, $\Pr(\mathbf{x}_k^j = \mathbf{x}_k^{*i}) = q_i$.

There are a number of possible approaches to applying a particle filter to the task of predicting a conductor's gestures. Our proposed approach is to encode prior knowledge of conducting gestures in the observation model, favoring an extremely simple motion model, while encoding the system state in just enough dimensions to maintain the expressive content of the gestures. We further propose using the particle filter in two stages: training and prediction. In the training stage, no prediction is performed, but a set of generic gestural templates is used to track the baton's motion and create adapted templates specific to the conductor and the piece of music. Then, in the prediction stage, the adapted gestural templates are used to track and predict the baton's movements.

State Vector

The number of particles required for stable operation of a filter grows exponentially with the dimensionality of the state vector (Khan, Balch, and Dellaert 2004). Keeping in mind the goal of real-time operation, the choice of state vector is therefore extremely important, as we must retain sufficient information for following and predicting the conductor's baton movements, while representing that information with a minimum of dimensions.

A straightforward state vector might track the 2-D position and velocity of the baton tip, yielding a simple four-dimensional model, and exploiting momentum to predict future states. This approach would likely exhibit weak predictive ability, however, as it ignores all prior knowledge of conducting technique. A more powerful approach needs to incorporate prior knowledge of the conducting

Table 1. State Vector x

Variable	*Encodes*	*Interpretation*
ϕ	Phase	Temporal position in piece, gesture
ϕ'	Phase Rate	Tempo
M	Scale	Dynamics
b_x, b_y	x, y bias	Emphasizing and directing gestures

gestures, but avoid unnecessarily increasing the dimensionality of the state vector.

As mentioned, our proposed approach is to encode the path associated with the conducting gesture in the observation model, ideally tracking only the *interpretive* or *perceptual* content of the gesture in the state vector. The intent is to track the underlying message being communicated by the conductor, independent of the specific language (gestures) used to convey it. This gives us sufficient information for tracking and prediction, while shifting the complexity of encoding specific conducting paths to the observation model. Of course, extracting perceptual content from conducting gestures is an open problem. For the purposes of this work we propose to track the subset of perceptual content summarized by the five-dimensional state vector depicted in Table 1. In this representation, phase encodes temporal position both within the musical piece and within the current gesture as a single variable.

Articulation information is deliberately absent from our state vector as we have chosen to encode this in the gestural templates, following the approach of Murphy (2004). Recall that articulation is expressed in the shape of the gesture and the time spent at its extremities, both of which will be encoded in our gestural templates, as shape and knot density, respectively. The result is that articulation will mostly be predetermined during the training stage. Extension of our method to tracking and predicting articulation at performance time would involve the addition of variable articulation to the state vector, as discussed in the Conclusions.

Observation Model

The observation model (Equation 2) is the mechanism that relates observed positions to the state vector. In our case, the measurement function $\mathbf{h}()$ must transform a given state vector to a position $\mathbf{z}$. This is done by first mapping a given phase to a spatial location via a gestural template $\mathbf{T}()$, then applying scaling and bias to yield a final position, as in

$$\mathbf{h}(\mathbf{x}_k) = M_k \cdot \mathbf{T}(\phi_k) + \mathbf{b}_k \tag{4}$$

where it is understood that the specific template utilized depends on the temporal position in the musical score.

Ideally, the template function $\mathbf{T}()$ will be sufficiently flexible to be easily adapted during training. We chose to implement our template as a circular cubic spline, for which the first and last knots, and derivatives at those knots, are forced to be equal. This generates a seamless, repeating gesture, which can be modified using straightforward spline manipulation techniques. An example template was shown in Figure 2.

Motion Model

When a new observation arrives, the motion model (Equation 1) generates a prior estimate of the system state, which is then subjected to resampling based on the observation. We base our formation of the prior on the concept of coherence: given a certain tempo, scale and bias, we assume the best estimate is that these variables will remain unchanged at the next observation. This introduces momentum to the system, and will limit the predictive range of the filter to the coherence time of the state variables, except phase. This should prove acceptable for the relatively short prediction ranges required for latency compensation.

Note that although we leave most of the state variables unchanged in the motion model, they do evolve over time to correspond to the observed performance, through the processes of roughening and resampling, discussed subsequently.

For the state vector described above, the motion model is therefore extremely simple: all variables propagate forward unchanged, except ϕ, for which

$$\phi_{k+1} = \phi_k + \phi'_k \Delta t \tag{5}$$

More complex models, incorporating momentum into the translation and scale dimensions, for example, may yield superior predictive ability, although this comes at the cost of a higher-dimensional state vector, as discussed in the Conclusions.

Likelihood Function and Resampling

The likelihood function (Equation 3) is used to reweight and ultimately resample the population of particles based on the most recent observation, so that they better represent the probability density of the system state. We use a standard uncorrelated Gaussian likelihood function, which in two dimensions simplifies to

$$\Pr(\mathbf{z}_k|\mathbf{x}_k^i) = \frac{1}{2\pi\sigma_v^2} \exp\left(\frac{-||\mathbf{z}_k - \mathbf{h}(\mathbf{x}_k^i)||^2}{2\sigma_v^2}\right) \qquad (6)$$

where σ_v is an estimate of the standard deviation of the observation noise, assumed to be identical in all directions, and $||\cdot||$ denotes Euclidean distance. Resampling is performed using the standard algorithm presented by Gordon, Salmond, and Smith (1993).

The observation noise estimate σ_v in the resampling process controls the selectivity with which particles propagate forward in time. For low noise estimates, only those particles closely surrounding the tracked system state survive, while higher estimates allow a more spread-out set of particles to survive. As discussed below, there is an interplay between selection of observation noise, particle count, and roughening constants.

Roughening and Noise

Roughening mutates the set of particles into novel portions of the state space, giving the particle filter its ability to track changes in the system state. The amount of noise added to the particles controls the extent to which the filter is capable of diverging from its previously tracked state. Excessive roughening spreads the particles too thinly, resulting in a loss of precision in the tracked state, whereas insufficient roughening makes the filter incapable of tracking novel changes.

Empirically, we have found that zero-mean independent additive Gaussian noise works well with our chosen state vector. Different amounts of roughening are required in training and prediction: Training must tolerate a high level of novelty, and so large roughening constants and a high estimate of observation noise are used, allowing the system to diverge significantly from the anticipated state. This necessitates the use of more particles in maintaining statistical significance and stability, but because training can be performed offline, the increased complexity is tolerable. During prediction, the use of templates specifically adapted to the conductor and the musical piece reduces the amount of novelty required; therefore roughening and observation noise can be reduced, as can the number of particles, reducing the computational burden.

An easily implemented optimization that we have pursued is to reshape the noise applied to each dimension such that out-of-bounds values are disallowed. Tempo, for example, must always be positive, thus any roughening value that yields a non-positive tempo is replaced. A similar logic applies to changes in phase. Although it is possible for the system to overestimate phase, thus necessitating a negative phase change to correct itself, we find that this never actually occurs if the rest of the system is correctly tuned.

Training

In training mode, the system takes a set of universal gestural templates and adapts them to the specific gestural style of the given conductor and piece of music. There is no prediction involved, and the process can be carried out offline, based on a video recording.

The first step is to break the music into sections, such that each one is well represented by a specific beat pattern and articulation. In future, even this step might be automated by running concurrent particle populations, each representing a competing hypothesis, and selecting the one that most accurately tracks the performance. In the present implementation the music is divided into relatively large sections, each containing many repetitions of

the same gestural template, i.e., many measures of music.

Next, the training filter is run over each section of music, modifying the generic template to a new, adapted one. This starts by forming a prior estimate of the system state $\mathbf{x}_{k+1}$ using the motion model (Equation 5), applying the observation model (Equation 4) to obtain a spatial estimate, and correcting nearby spline knots in the direction of the error with respect to the observation z_k, as in

$$\mathbf{t}^{\phi}_{k+1} = \mathbf{t}^{\phi}_{k} + \lambda d^{\phi}[\mathbf{z}_k - \mathbf{h}(\mathbf{x}_{k+1})] \quad (7)$$

where $\mathbf{t}^{\phi}_{k+1}$ is a set of spline knots centered on the current phase ϕ, λ is an adaptation rate constant, and d^{ϕ} is a window limiting the effect of the adaptation on knots far from the current phase. This has the effect of deforming the template with a vector proportional to the error between the prior estimate and the newly observed data. The strength of the deformation, and the extent of its influence on the spline, are tuned to achieve stable but responsive behavior. A further spline-smoothing step ensures that the spline remains well formed. This process may be repeated over the same section of music several times in order to form a stable, representative template.

The adapted template contains all the expressive elements described by the state vector $\mathbf{x}$ (see Table 1). This is undesirable, as we wish to maximize expressive freedom during performance. If such expressive elements were retained from training, prediction would be inferior when the conductor deviates from the training performance. Finding the optimal balance between training and spontaneity is an open question, although allowing maximum spontaneity seems a prudent starting point.

Removal of expressive content from the adapted templates is already accomplished in part by virtue of learning over extended time spans: Changes of short duration relative to the training period are averaged out. Content that exists over longer timescales may remain, however, as in the example of an extended period of high dynamics. As such, we have maintained the overall gesture shape, but normalized the scale and bias in order to ensure that both short- and long-timescale expressive elements are removed. The normalization is carried out straightforwardly by dividing by the standard deviation of the spline knot positions and subtracting their mean. Note that we have not removed articulation from the adapted templates, in that the shape of the gesture and the amount of time spent at its extremities is learned and not normalized, and as a result this element is not varied at performance time. An extension to do so is discussed in the Conclusions.

Finally, the set of adapted templates must be processed such that their endpoints align. This ensures that major transitions in the music (e.g., transitions between beat patterns) can be carried out smoothly and without pauses. This may be performed using spline smoothing, applied over the template function sequence, or by introducing a gradual translation to the gestural template, which, over the course of a section of music, aligns its endpoint with the starting point of the next template. We have not evaluated either of these techniques experimentally.

Prediction

With the particle filter tracking the state of the baton using an adapted template, prediction is a simple matter of projecting the system state into the future by the required amount of time. Given a state vector $\mathbf{x_k}$, the predicted state vector is found using the motion model (Equation 5) with the desired prediction range as Δt. Recall that we have proposed a very simple motion model, in which scale, translation and tempo remain unchanged, while phase is updated based on tempo. Because this motion model is linear, we need not carry out prediction on all particles individually, but can rather obtain the average system state and project that into the future in a single step, saving considerable computational effort. The predicted position of the baton tip is found from the predicted system state using the observation model (Equation 4) and the adapted template. A simple recursive low-pass filter on the predicted position takes the mean value of the present prediction and the previous filtered prediction, yielding a smoother predicted trajectory.

Table 2. Tracked Features in the 4/4 Beat Pattern

Point	*Velocity zero-crossing*	*Position bounds*
A	$-/+$ in v_y	$\lvert p_x - b_x \rvert < M \cdot K_x$
B	$-/+$ in v_x	$p_x - b_x < -M \cdot K_x$
C	$+/-$ in v_x	$p_x - b_x > M \cdot K_x$
D	$+/-$ in v_y	$p_y - b_y > M \cdot K_y$

Feature-based Prediction

For the sake of comparison we also implemented a simple feature-based prediction system. Prior work has established a method for prediction of beat location while tracking scale, translation, and tempo (Murphy 2004). Extension to prediction of trajectory over arbitrary durations based on these tracked variables is relatively straightforward.

We again restrict our focus to a 4/4 beat pattern, assume access to a suitable gestural template, and track the system state in the same five dimensions as for the particle filter, shown in Table 1. Tracking relies on recognizing features as they occur in the conductor's gestures. Because we are not interested in exact beat locations, we chose to track features based entirely on their suitability for automated detection. Zero crossings of velocity, for example, can be detected simply, and have the added benefits of being independent of scale and translation, and of representing well-defined points on the gestural template.

The four features we track are listed in Table 2. An additional constraint on position is added to improve robustness, although this introduces a dependence on scale and translation, which are taken from the current state estimate. A simple state machine is added to constrain the order in which the features may arise, and for greater robustness a minimum duration is introduced to each state, based on prior knowledge of the tempo of the piece. The features' locations were shown on a generic four-beat template in Figure 2.

As each new sample arrives, the tracked phase is incremented using the current estimate of tempo. When a feature is recognized, a temporal ratio is found between the observed duration between features and the expected duration based on the template. As each gesture is completed, an estimate of the instantaneous tempo is updated based on the average values of the temporal ratios over the gesture. Translation is estimated as the mean of the baton's position over the gesture, and scale is based on the absolute value of the baton's position relative to its translated center. Note that these definitions of translation and scale are somewhat arbitrary, in that they do not correspond to any physically significant quantities, but were chosen for computational simplicity and robustness to noise. Note also that we are using data from the entire gesture to find scale and translation.

Based on the tracked state, a prediction is synthesized from the template, as

$$\mathbf{p}(\phi_k, \Delta t) = M_k/M_T \cdot \mathbf{T}(\phi_k + \phi'_k \Delta t) + \mathbf{b}_k \qquad (8)$$

where $\mathbf{p}()$ is the predicted position for a given starting phase and prediction range Δt, M_k/M_T is the ratio of tracked scale to that of the template, $\mathbf{T}()$ is the template function with zero mean, and ϕ_k, ϕ'_k, and $\mathbf{b}_k$ are the tracked phase, tempo, and translation.

Evaluation Criteria

The final measure of the impact of a conductor's gestures is in the musicians' response to them in a real performance context. The challenge here is that orchestral performances can vary significantly, even when presented with identical gestures. Moreover, evaluation of performance is difficult, requiring experienced, skilled experts, whose evaluations are necessarily at least partially subjective. Moreover, such evaluations tend towards qualitative rather than quantitative information. An automated approach producing objective, quantitative results is desirable.

Unfortunately, even in our very specialized case of comparing two ideally identical performances, automated evaluation is difficult. Establishing the time shift between performances should be relatively easy using methods from musical score following (Orio, Lemouton, and Schwarz 2003), but the other aspects of a conductor's influence on the performers—articulation and dynamics, in

Table 3. Mathematical Estimators of Perceptual Metrics

Shorthand	*Metric*	*Notes*
$\mathbf{p}$	$\|\mathbf{p}_A - \mathbf{p}_B\|^2$	Position, in a sense, encompasses all metrics, but is sensitive to absolute scale and translation
$\|\mathbf{v}\|$	$(\|\mathbf{v}_A\| - \|\mathbf{v}_B\|)^2$	Speed is useful for tempo, articulation and scale, but is sensitive to absolute scale
$\mathbf{p}/\|\mathbf{v}\|$	$\frac{\|\mathbf{p}_A - \mathbf{p}_B\|^2}{\|\mathbf{v}_A\|}$	Position normalized to speed, no scale dependency
$\mathbf{v}$	$\|\mathbf{v}_A - \mathbf{v}_B\|^2$	Velocity mostly represents tempo, direction, and scale, but is sensitive to absolute scale
$\mathbf{a}$	$\|\mathbf{a}_A - \mathbf{a}_B\|^2$	Acceleration mostly represents curvature and changes in tempo and scale, but is sensitive to absolute scale
$\|\mathbf{a}\|$	$(\|\mathbf{a}_A\| - \|\mathbf{a}_B\|)^2$	Magnitude of acceleration is useful for curvature, articulation, and scale, but is sensitive to absolute scale
$\mathbf{v}/\|\mathbf{v}\|$	$\left\| \frac{\mathbf{v}_A}{\|\mathbf{v}_A\|} - \frac{\mathbf{v}_B}{\|\mathbf{v}_B\|} \right\|^2$	Direction of travel, no dependency on absolute scale or translation
$\mathbf{a}/\|\mathbf{a}\|$	$\left\| \frac{\mathbf{a}_A}{\|\mathbf{a}_A\|} - \frac{\mathbf{a}_B}{\|\mathbf{a}_B\|} \right\|^2$	Direction of acceleration, (partially) represents curvature, with no dependency on absolute scale or translation

particular—would be very difficult to evaluate in an autonomous manner. For these reasons, we have foregone evaluation of orchestral performance for the time being, in favor of working directly with baton trajectories.

As a first attempt at establishing perceptually based metrics for conducting gestures, we have attempted to identify a set of quantifiable concepts that encapsulate the salient content of a conductor's performance. These include absolute tempo, changes in tempo near extrema (articulation), curvature near beats and extrema (beat location, articulation), direction of travel, changes in scale (dynamics), and changes in translation. It should be stressed that these metrics exceed the level of detail expressed in our particle filter's state vector (described in Table 1), as appropriate for the purpose of providing a meaningful evaluation.

Absolute scale and translation are of course important, but we believe these to be less critical than changes in these dimensions. If a conductor starts a performance with slightly larger gestures, for example, and effectively scales the rest of his or her performance to match, the resulting orchestral performance may be generally unchanged.

We wish to approximate these concepts with a set of metrics that may be applied directly to the baton's trajectory. Thus, we present a somewhat exhaustive set of potential metrics (see Table 3) with the intention of evaluating them and selecting the most informative. Note that the labels A and B refer to the two performances to compare, and the shorthand nomenclature is used to label plots in the following sections.

We operate exclusively on the instantaneous position $\mathbf{p}$, velocity $\mathbf{v}$, and acceleration $\mathbf{a}$ of the baton tip, with the last two easily estimated as the first and second difference of position, respectively. Although it would be possible to use more-complex schemes that attempt to compare segments of trajectories rather than instantaneous points, the added complexity does not appear to be justified by the limited additional information such an approach would provide. In conducting gestures, timing is essential, after all, and thus introducing tolerance to temporal error into our metrics is undesirable. Note also that the metrics may be applied uniformly across the gesture, as is our intent, or weighting may be applied so as to emphasize the importance of some metrics near beat locations and extrema in the gesture. Exploration of these options is left for future development. At present, all metrics are computed as an unweighted mean across the full set of observations of the baton tip's position available on each trial.

Establishing an overall performance metric for the conductor's baton paths may be as simple

as formulating a weighted sum of the metrics listed here. The most meaningful way of doing this would involve a set of experiments asking a number of expert conductors and musicians to rank the similarities of sets of conducting gestures. By presenting the test group with appropriately varied gestures, estimates of the perceptual impact of each metric may be formed and applied to an overall perceptually based metric. For the purposes of this article, however, we will simply present the individual metrics.

Simulation Results

A number of trials were performed using simulated data to verify correct operation of the particle filter, operating only as a tracker, i.e., with no prediction. Simulated baton trajectories were created by replaying a motion template with modulated tempo, bias and scale. Both scale and tempo were varied sinusoidally around nominal values of 1.0; completing two full cycles over the sequence, and with modulated values falling between 0.9 and 1.1. We use normalized tempo, such that a value of 1.0 corresponds to 20 repetitions of the gesture over the 45-sec sequence.

Using a normalized template of one spatial unit in amplitude, bias followed a circular path, with a radius of 0.3 spatial units, also completing two full cycles over the sequence. Initially, the generic motion template depicted in Figure 2 was utilized, but the more relevant results shown here make use of the motion template adapted to motion-capture data. This is described in the following section, along with the process of tuning the filter.

Validation via simulation is particularly appealing in this case, as ground-truth values are available for the entire state vector. In contrast, for real-world data, such perfect knowledge of tempo, scale, and bias, which serve to confirm that the filter is correctly adapting to the input sequence, is not available. Figure 3 depicts the results of the simulation, showing that the filter is indeed adapting to variations in the performance. Root-mean-square (RMS) errors are indicated in the figure. Phase and position results are omitted because, in the absence of noise or deviation from the motion model, the tracked and ideal results are nearly identical, with RMS errors in x and y positions of 0.0191 and 0.0244 spatial units, respectively, and in phase of 0.0421 gesture repetitions.

These results do not establish the viability of this technique for real-world data—that is left to the following section. Rather, it shows that the filter is capable of correct operation over a nontrivial range of potential inputs.

Results

In the previous section we established the viability of the particle filter approach using synthetic data. Here we test the particle filter on motion capture data of a real conductor, comparing the filter with EKF and feature-based approaches.

Data Collection

Two sets of data were collected using a Vicon motion capture system operating at 120 Hz. Several motion markers were included in the capture, but only the marker on the tip of the baton was used in the experiment. These sessions involved a conductor "performing" with a recording of Mozart's opera *Don Giovanni*, playing at 180 bpm. The use of a recording rather than a live responsive orchestra was a concern, the worry being that the static nature of the performance might yield unrealistically consistent conducting gestures. To alleviate this concern the recordings were separated by several days, with the goal of increasing variation between performances. Indeed, a manual inspection of the resulting data reveals significant variations between the two performances in terms of translation, scale, shape of trajectory, and even phase. A sample starting 16 sec into the piece and showing variations in vertical baton movement for the two performances is shown in Figure 4. In the results that follow, the first of the two captures was treated as training data, and the second as a performance to be predicted. In both cases, the first 30 sec of performance were used. Source data and sample videos of the particle filter in training

Figure 3. Particle filter tracking a simulated baton trajectory with modulated tempo, scale, and bias: (a) normalized tempo (RMS error 0.0337), (b) scale (RMS error 0.0611), and (c) x, y *bias (RMS error 0.02882 and 0.02516).*

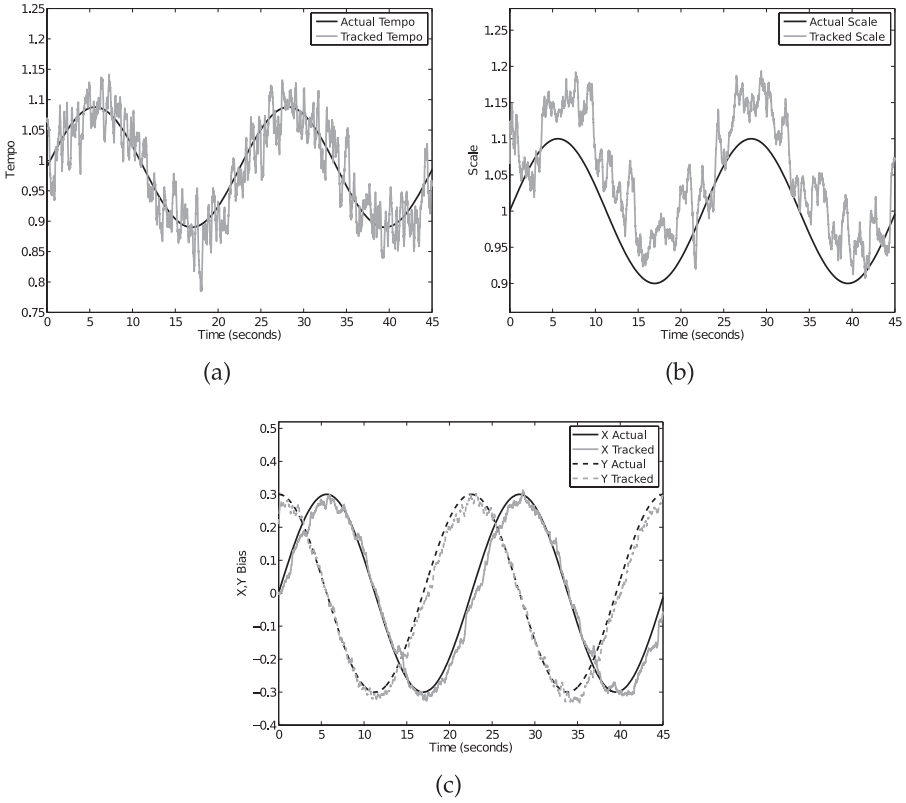


Figure 3

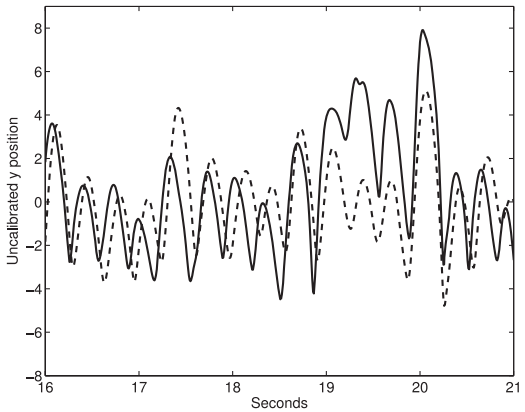


Figure 4

and prediction modes are available online from www.cim.mcgill.ca/sre/projects/worldopera/baton and on the DVD that will accompany *Computer Music Journal* 37:4.

Figure 4. Typical differences in vertical baton motion for two performances of Don Giovanni*. The solid line depicts the first performance and the dashed line depicts the second.*

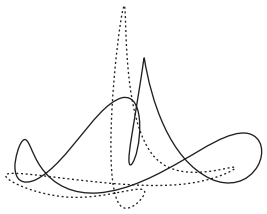


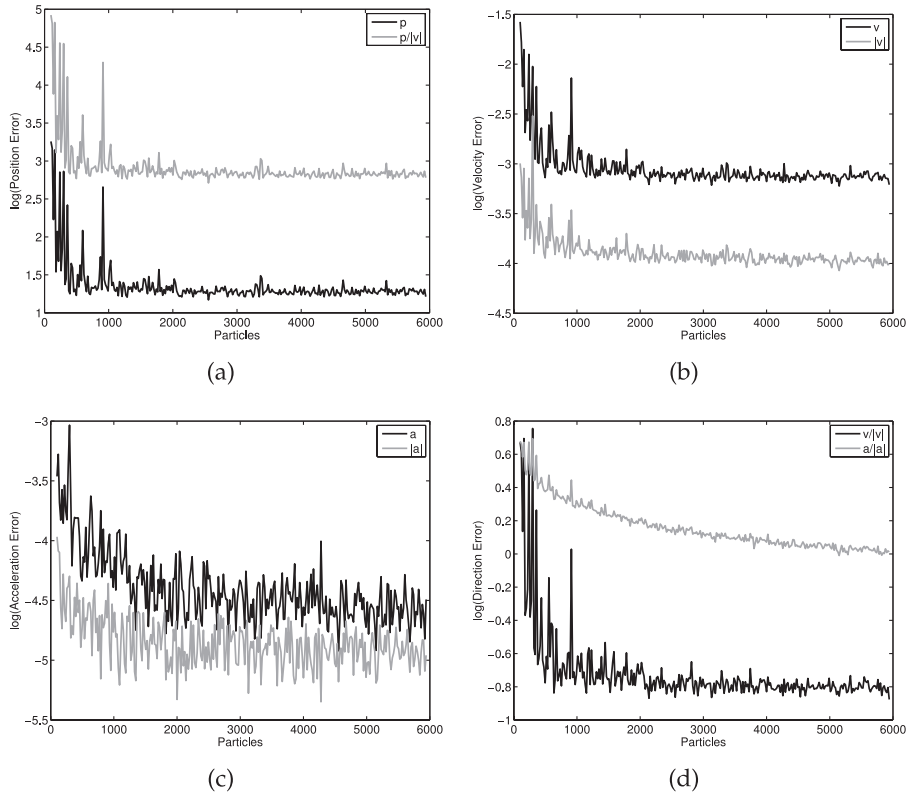
Figure 5

Figure 5. Template adapted using training data. The dashed line depicts the generic template and the solid line depicts the adapted one.

Training

The particle filter was applied in training mode to the generic template of Figure 2, with parameters tuned as described in the following. During the first 30 sec of the training data, the particle filter generated the adapted template shown in Figure 5. The adapted template is representative of the mean of the baton's trajectory, which varied over the entire

Figure 6. Prediction error of particle filter as a function of particle count: (a) position, (b) velocity, (c) acceleration, and (d) direction error.



30 sec. Training over shorter durations yields a more accurately adapted template, at the risk of overtraining to a specific segment of the performance, and thus reducing the expressive capability of the prediction system.

Tuning the Particle Filter

The roughening constants, observation noise estimate, and number of particles associated with the particle filter must be chosen. This presents an opportunity to utilize and validate the metrics described in Table 3: Plots of error as a function of each parameter should reveal useful trends in selecting their values. Initially, 4,000 particles were used, and the remaining parameters roughly hand-tuned to obtain stable results. Next, plots of each metric as a function of the individual variables were produced and utilized to refine the settings. The procedure was repeated over a few iterations, as the variables in play are not independent, and each iteration brings performance closer to a global optimum.

Plots of each error metric as a function of the number of particles and observation noise estimate are shown in Figures 6 and 7, respectively, and the parameters selected for the particle filter are given in Table 4. For all parameters, there is a region of instability associated with lower values, meaning that the filter was unable to track the baton's motion (e.g., in Figure 6 with fewer than 800 particles and in Figure 7 for observation noise less than about 1.7). In the case of the roughening parameters and noise estimate, the region of instability is followed by a global minimum, then a region of decreasing performance as the parameter value increases. The decrease in performance as roughening increases is due to the spreading of hypotheses over the state space, which results in a less accurate representation of the correct hypothesis. Parameters were

Figure 7. Prediction error of particle filter as a function of estimated standard deviation of observation noise: (a) position, (b) velocity, (c) acceleration, and (d) direction error.

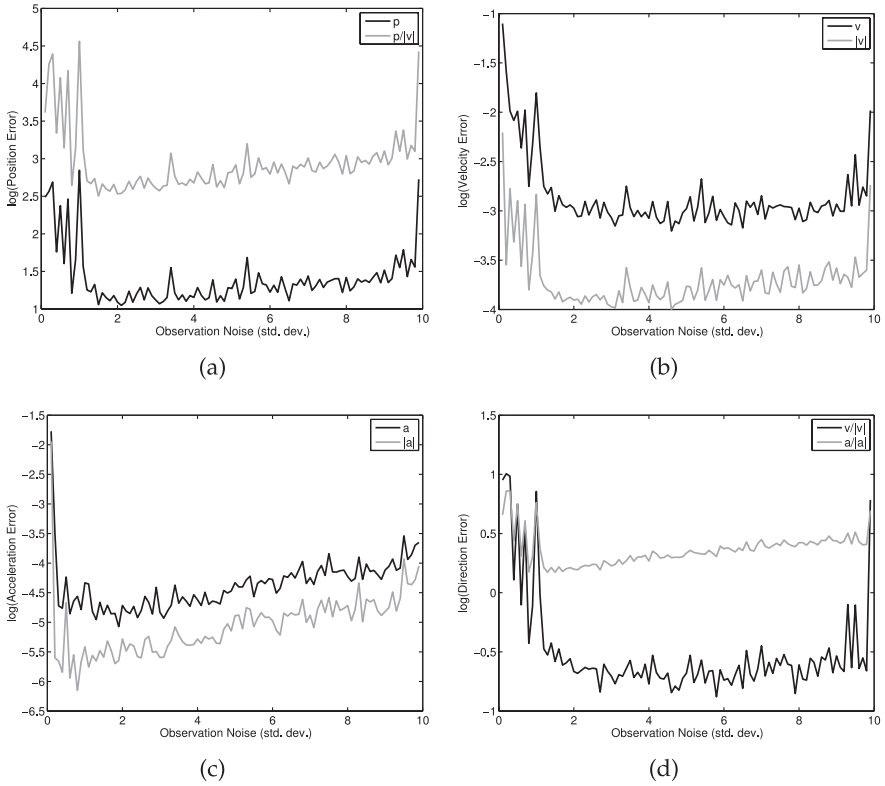


Table 4. Particle Filter Parameters

Particle Count	1020
Observation Noise (std. dev.)	4.0
Roughen ϕ	1.4×10^{-3} rad
Roughen ϕ'	1.3×10^{-4} rad/sec
Roughen M	1.7×10^{-3}
Roughen b_x	0.27
Roughen b_y	0.19

empirically chosen to balance error and stability, and did not generally correspond to the error minima.

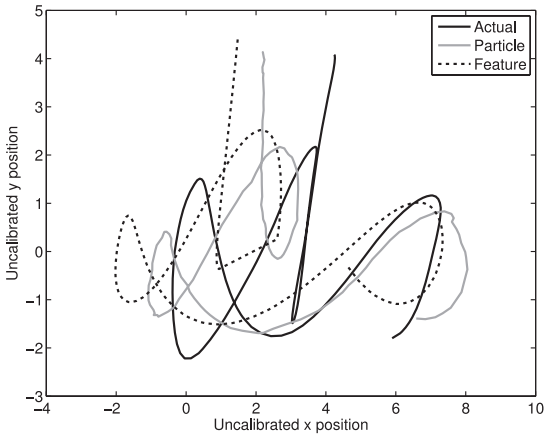
The error metrics were consistent, providing similar trends in the regions explored. Some metrics proved to be more useful, displaying more gradual and smoother variation. In particular, acceleration and direction metrics exhibited steady variation over the parameter range. In contrast, position metrics exhibited little variation beyond the region of instability in the plots of particle count vs. error. Because all the metrics conveyed some useful information, we consider the full set in evaluating prediction methods.

Comparing Predictors

The adapted gestural template formed by the training particle filter was used with both particle- and feature-based predictors—this maximizes the performance of the feature-based method, and simulates tailoring a generic template to better match the conductor's style. For a prediction range of 200 msec, the predictors generated the paths shown in Figures 8 and 9. In these and the plots that follow, the actual performance is shown for reference, time-shifted by the prediction range to align with the predicted results.

The results demonstrate both predictors successfully tracking changes in translation, scale, and

Figure 8. Section of predicted paths (prediction range of 200 msec) during an increase in scale and translation along positive x.



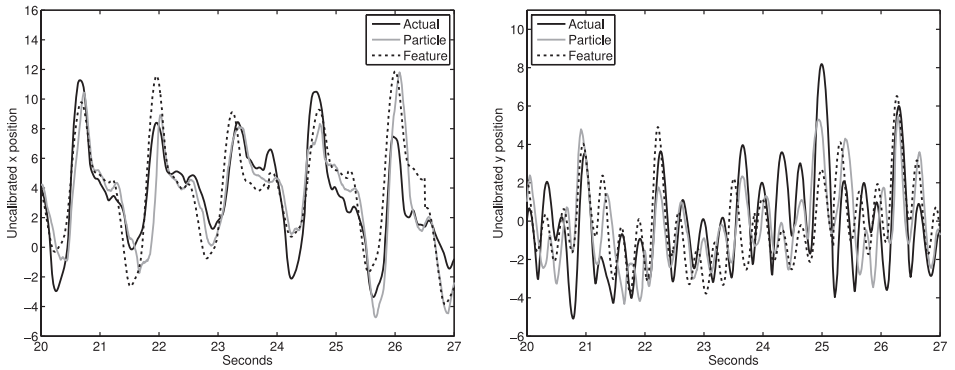
phase, although it is clear that the particle filter is adapting more quickly and fluidly to these changes. Note, for example, the abrupt discontinuity in the feature-based trace in Figure 8, where an update of parameters leads to a sudden jump in the output of the predictor. Such discontinuities are less likely with the particle filter, for which parameters are continually updated with each new input sample. This has the drawback, however, of yielding a rougher trajectory, a shortcoming that is partially addressed by the smoothing filter on the predicted baton position.

Next, prediction error was evaluated as a function of prediction range, for both filter types, with the results shown in Figure 10. Notably, neither method demonstrates a clear advantage over the entire prediction range. For most metrics, the particle filter offers better predictions for less than one or two seconds, and the feature-based method shows a clear advantage for a range of two seconds or more. We suspect this is a function of the temporal coherence of the state variables for the specific musical selection. A gesture at a given scale has a high probability of staying at that scale in the immediate future, but that probability falls off with time. Because the particle filter responds more quickly, it is better at dealing with effects with short coherence times. As the prediction range increases, however, closely tracking changes with short coherence times presents less of an advantage, and may indeed be detrimental: The scale of the gesture in four seconds may not correlate well with its present scale, and may indeed show a negative correlation. This is the case in the musical selection utilized in evaluating these methods, as there is a section of alternating high and low dynamics. Because the particle filter more accurately follows the scale of the gestures, it actually suffers for prediction ranges that correspond to these regions of negative correlation, whereas the feature-based method has a longer response time, and so acts as a sort of low-pass filter over the alternating scale. This is most clearly seen in Figure 10d, in which the error associated with the particle filter increases beyond that of the feature-based method, then begins to level off, and finally re-approaches the performance of the feature-based method. For prediction ranges below one second, the particle filter offers superior results by all metrics. This makes the particle filter the more appropriate method for our applications, which call for prediction ranges on the order of 100 msec. In terms of computational efficiency, the feature-based method is significantly simpler, though both methods operate in real-time on modern computers. If substantially more particles were required by the particle filter (as, for example, necessitated by a more complex state vector), real-time operation may become difficult.

As confirmation that other machine learning techniques might apply, we tested an EKF in the roles of tracking and prediction. Replacing the tracking and prediction particle filter with an EKF was straightforward, making use of the same state vector, observation model, and motion model as the particle filter. Despite its simplicity, the EKF actually outperformed the particle filter on short prediction intervals in all metrics, as seen in Figure 11. The main shortcoming of the EKF is that, unlike the particle filter, it is unable to simultaneously track multiple hypotheses of the system state. Because of this shortcoming, we were unable to utilize the EKF in the training role in a straightforward manner: It lost track of baton movements more easily than the particle filter, making training impossible. This shortcoming may also partially explain the loss of performance in the EKF at higher prediction ranges, seen most clearly in the acceleration metrics in Figure 11c.

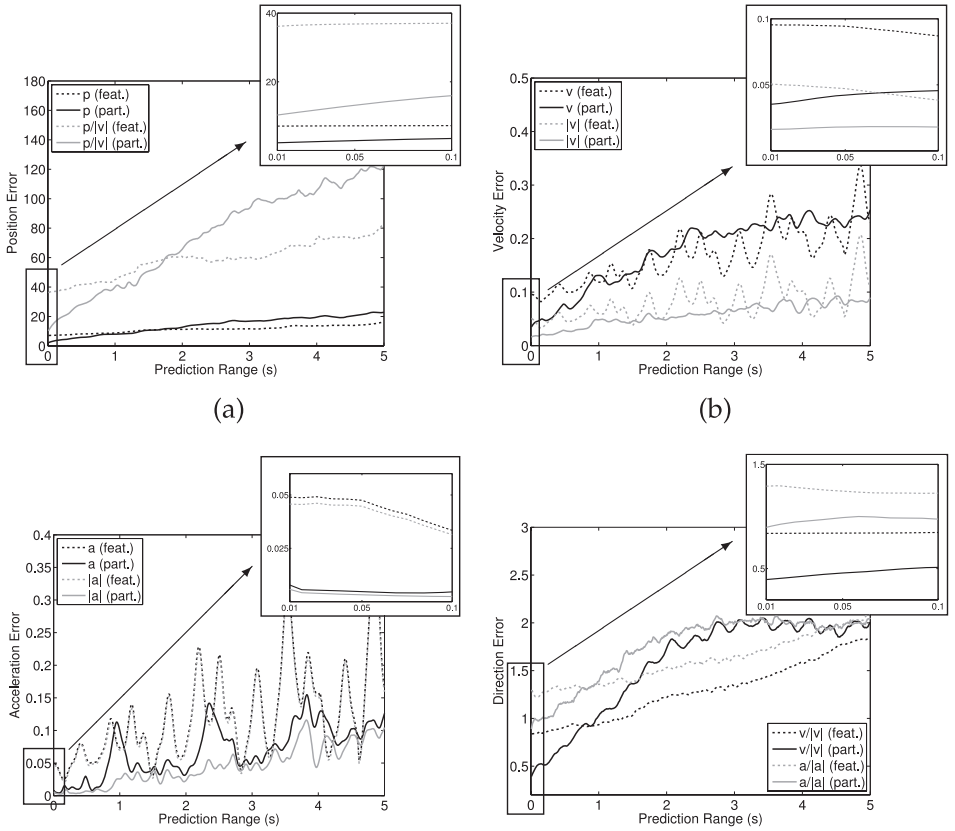
Figure 9. Predicted paths in (a) x and (b) y dimensions (prediction range of 200 msec).

Figure 10. Prediction error as a function of prediction range, comparing particle filter and feature-based method: (a) position, (b) velocity, (c) acceleration, and (d) direction error.



(a) (b)

Figure 9

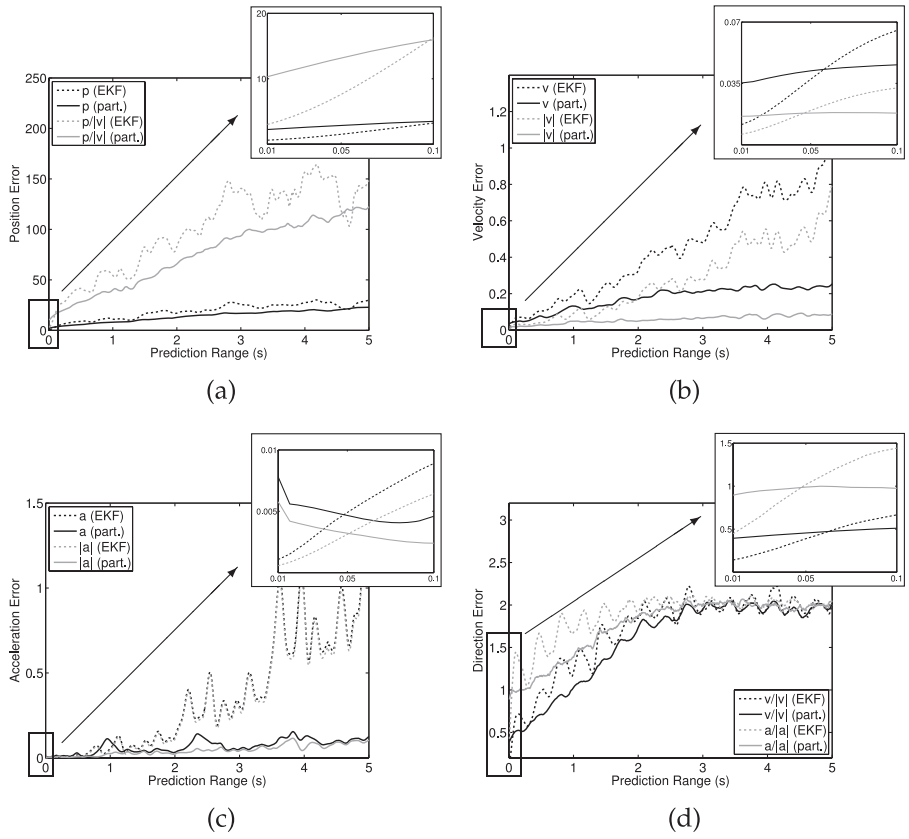


(a) (b)

(c) (d)

Figure 10

Figure 11. Prediction error as a function of prediction range, comparing particle filter and EKF: (a) position, (b) velocity, (c) acceleration, and (d) direction error.



Qualitative Assessment

As a preliminary qualitative assessment of the predictive methods, a conductor was asked to blindly rank and assess five short performances, comprising particle filter and feature-based predictions 50 and 100 msec into the future, and the original recording. Each sample consisted of the trajectory of the baton tip and the corresponding audio track, temporally aligned with the prediction look-ahead where relevant. The conductor ranked the original performance as the best, followed by the particle filter at 100 msec, then at 50 msec (this result due to slight phase variations throughout the piece), and finally the feature-based method at 50 msec and 100 msec. The most obviously distracting element of the predicted paths was the discontinuities in the feature-based method. Dynamics tracking was evaluated as muted, even in the best cases. The conductor observed that the predictions were imperfect, but promising in establishing the viability of prediction in general. Note that the smoothing output filter was not yet in place when this evaluation was performed.

Conclusions and Future Work

The results indicate that for relatively low prediction ranges—one second or less, in the case of our specific musical selection—our particle filter-based predictor outperforms the feature-based predictor. The results were assessed qualitatively by a conductor, and again the particle filter outperformed the feature-based methods. A number of concerns were raised by the conductor, including insufficient responsiveness in the dynamics and variations in phase even over a subset of the piece.

With respect to the intended application, the output achieved from both methods is likely sufficient for the purposes of carrying out an initial set of experiments in combating latency in distributed orchestral performances. Because these applications will likely require a prediction range of less than one second, the particle filter is a natural choice.

As evidence that other machine learning techniques might apply, we also demonstrated an EKF taking the place of the particle filter in tracking and prediction over short intervals. Its inability to track multiple hypotheses makes use of the EKF in the training task less straightforward, however.

An obvious next step is to repeat the experiments presented here on a range of musical pieces, with a number of different conductors. Doing so with data collected under real performance conditions would better validate the utility of the system in real-world conditions. To address the issues raised in the qualitative assessment, the particle filter might be adjusted to maximize performance with respect to perceptually based metrics. For example, the filter could be re-tuned to exhibit greater responsiveness to acceleration and velocity changes in areas where the dynamics were considered to be muted.

One might further increase the predictive accuracy and range of both methods through the use of a more comprehensive state space. Including articulation as a state variable, for example, would increase the flexibility of the system significantly. Similarly, adding velocity to the translation and rate of change to the scale dimensions should increase the predictive ability of the system. Each of these enhancements necessitates more particles, however, posing a challenge to real-time implementation. The use of Rao-Blackwellization (Khan, Balch, and Dellaert 2004) to exploit redundancy in the state space might simplify the filter sufficiently for real-time operation, even with this higher dimensionality.

Additional work is needed in addressing the problems of starting a performance, automated segmentation for training, and automatic determination of model, some of which could be derived directly from the input data. Looking further ahead, these techniques might be applied to a more complex motion model, including the 3-D structure of the conductor's arm. As these problems are solved, we look forward to testing the predictive techniques to combat latency in an actual distributed performance.

Acknowledgments

The authors would like to thank the Centre for Interdiscplinary Research in Music Media and Technology (CIRMMT) at McGill, the Center for Research in Computing and the Arts (CRCA) at the University of California at San Diego, the Fonds Quebeçois de la Recherche sur la Nature et les Technologies (FQRNT), and the Natural Sciences and Engineering Research Council of Canada (NSERC)/Canada Council for the Arts New Media Initiative for their support of this research.

Dedication

Our co-author, Dr. Nathan Brock, died unexpectedly in July 2012 at the young age of 35. Those who had the pleasure of working with Nathan remember him as a consummate professional and an inspiration as a colleague. He deserves significant credit both for the initial idea underlying this article, and for the experimentation, which could not have been carried out without his efforts. We are deeply saddened by the loss of our friend and colleague, and dedicate this article, of which he was very proud, to his memory.

References

Bartlette, C., et al. 2006. "Effect of Network Latency on Interactive Musical Performance." *Music Perception* 24(1):49–59.

Bevilacqua, F., et al. 2010. "Continuous Realtime Gesture Following and Recognition." In S. Kopp and I. Wachsmuth, eds. *Gesture in Embodied Communication and Human-Computer Interaction, Lecture Notes in Computer Science*, vol. 5934. Berlin: Springer, pp. 73–84.

Black, M., and A. Jepson. 1998. "Recognizing Temporal Trajectories Using the Condensation Algorithm." In *Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 16–21.

Chafe, C., J. Cáceres, and M. Gurevich. 2010. "Effect of temporal separation on synchronization in rhythmic performance." *Perception* 39(7):982–992.

Gordon, N. J., D. J. Salmond, and A. F. M. Smith. 1993. "Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation." *Radar and Signal Processing* 140(2):107–113.

Ilmonen, T., and T. Takala. 2005. "Detecting Emotional Content from the Motion of an Orchestra Conductor." In *Proceedings of the Sixth International Conference on Gesture in Human-Computer Interaction and Simulation*, pp. 292–295.

Khan, Z., T. Balch, and F. Dellaert. 2004. "A Rao-Blackwellized Particle Filter for EigenTracking." In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 980–986.

Kolesnik, P., and M. Wanderley. 2004. "Recognition, Analysis, and Performance with Expressive Conducting Gestures." In *Proceedings of the International Computer Music Conference*, pp. 572–575.

Lee, E., et al. 2006. "CONGA: A Framework for Adaptive Conducting Gesture Analysis." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 260–265.

Lee, Y. W. 2008. "Application of the Particle Filter for Simple Gesture Recognition." In *Proceedings of the Fourth International Conference on Intelligent Computing: Advanced Intelligent Computing Theories and Applications*, pp. 534–540.

Modler, P., and T. Myatt. 2004. "A Video System for Recognizing Gestures by Artificial Neural Networks for Expressive Musical Control." In *Gesture-Based Communication in Human-Computer Interaction*, pp. 541–548.

Morita, H., S. Hashimoto, and S. Ohteru. 1991. "A Computer Music System that Follows a Human Conductor." *Computer* 24(7):44–53.

Murphy, D. 2004. "Live Interpretation of Conductors' Beat Patterns." In *Proceedings of the 13th Danish Conference on Pattern Recognition and Image Analysis*, pp. 111–120.

Nijholt, A., et al. 2008. "The Virtual Conductor: Learning and Teaching about Music, Performing, and Conducting." In *Proceedings of the International Conference on Advanced Learning Technologies*, pp. 897–899.

Olmos, A., et al. 2009. "Exploring the Role of Latency and Orchestra Placement on the Networked Performance of a Distributed Opera." In *Twelfth Annual International Workshop on Presence*, p. 9.

Orio, N., S. Lemouton, and D. Schwarz. 2003. "Score Following: State of the Art and New Developments." In *Proceeding of the International Conference on New Interfaces for Musical Expression*, pp. 36–41.

Rudolf, M. 1995. *The Grammar of Conducting: A Comprehensive Guide to Baton Technique and Interpretation*. 3rd ed. New York: Schirmer Books.

Visell, Y., and J. Cooperstock. 2007. "Modeling and Continuous Sonification of Affordances for Gesture-Based Interfaces." In *Proceedings of the 13th International Conference on Auditory Display*, pp. 423–429.

Welch, G., and G. Bishop. 1995. "An Introduction to the Kalman Filter." Technical Report TR95-041, University of North Carolina, Department of Computer Science, Chapel Hill, North Carolina.

Willey, B. 1990. "The Relationship between Tempo and Delay and its Effect on Musical Performance." *The Journal of the Acoustical Society of America* 88(S1):71.

Yuen, S., P. Novotny, and R. Howe. 2008. "Quasiperiodic Predictive Filtering for Robot-assisted Beating Heart Surgery." In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3875–3880.