Ponticello: An Interactive Conducting System for Mixed Music Performance

Nikolaus Knop
University of Music and Theatre Munich
nikolaus.knop@stud.hmtm.de

ABSTRACT

In composed music that combines acoustic instruments with electronic processing or fixed media, synchronizing acoustic and electronic layers remains a persistent challenge. The use of click tracks, while technically effective, significantly restricts the performers' freedom to expressively shape musical time. This paper presents Ponticello, a system that addresses the synchronization problem by inferring the ensemble's tempo from a video stream of the conductor in real-time. Instead of the ensemble being beholden to a fixed digital click track, the computer follows the flexible pulse indicated by the conductor, which already functions as a shared temporal reference for the human performers. Although the idea of interactive conducting systems is not new —it has been researched since the 1970s — research has largely focused on applications that simulate instrumental performances based on MIDI scores, which limits their applicability to the performance of mixed music. To support a broad range of compositional strategies for mixed music, Ponticello instead models the electronic part as a timeline of temporally extended, electronic processes whose playback tempo is continuously controlled by the conductor. The system has proven sufficiently reliable and accurate in rehearsal and concert settings across multiple conductors.

1. INTRODUCTION

In most music written for human performers, the free and spontaneous shaping of musical time is a central means of musical interpretation. Ritardandi and accelerandi, whether microscopically small or clearly perceptible, account for much of this music's expressive power. They allow phrases to breathe and musical form to be articulated organically.

In the context of composed mixed music, however, which combines acoustic instruments with fixed media or electronic processing, temporal flexibility is often constrained, when precise synchronization between the live performers and the playback of the electronic performance score is required. (The term *electronic performance score*, in this paper, refers to the representation of the electronic part of the composition. Analogous to instrumental parts for human musicians, it specifies which actions or processes the computer should perform and when these should occur.)

In common practice, synchronization is typically achieved using click tracks: the musicians hear a metronome through headphones and synchronize themselves with the computer in this way. This demands attention that would otherwise be directed toward one's own playing and interaction within the ensemble, and it allows little to no freedom in shaping musical time.

At the core of this problem lies a discrepancy between two different concepts of time: on the one hand, the elastic time of human perception; on the other hand, the wall clock time of the computer. Synchronization via a click track resolves this discrepancy in favor of the computer: the ensemble adapts to the computer's fixed temporal grid.

Ponticello reverses the direction of synchronization. Instead of requiring performers to align their playing to a fixed grid, the electronic system adapts to the flexible pulse indicated by the conductor, which already functions as a shared temporal reference for human performers. The conductor's gestures are captured by a camera and analyzed in real time to infer tempo information. This information is used to control the playback speed of an electronic performance score executed by the computer. The electronic performance score is modeled as a timeline of processes specifiable in the SuperCollider language and is editable through a graphical user interface.

By extending the role of the conductor to include the control of the electronic system, the computer becomes capable of following tempo variations in real time. In this way, the electronic performance score is no longer anchored to wall clock time, but to the elastic musical time indicated by the conductor. This allows for precise coordination without constraining the performers' freedom of expressive timing.

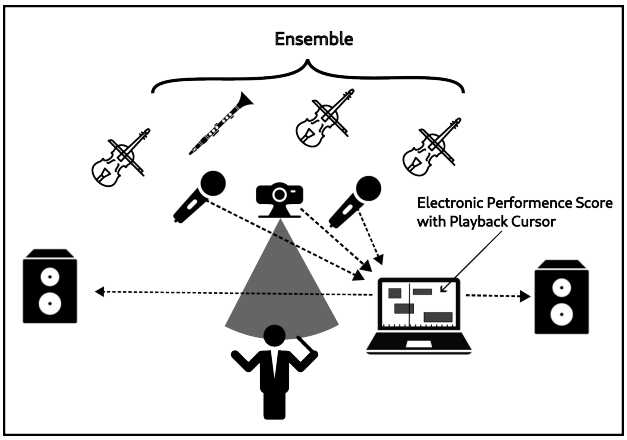

Figure 1. Performance setup

The system is, at the present stage, limited to the temporal coordination aspects of conducting. Other aspects associated with conducting, such as dynamics or articulation, are not addressed.

This paper is a preprint submitted to the International Computer Music Conference (ICMC 2026).

2. RELATED WORK

Research on interactive conducting systems has been conducted since at least the 1970s, spanning a wide range of technical approaches [1]. To situate the specific contribution of the present work, it is helpful to categorize this body of research according to both its intended purpose and its underlying technical design.

2.1 Categorization of Existing Systems

2.1.1 Application domains

According to Lee et al., interactive conducting systems can generally be grouped into three main application domains [1]. Most research has been devoted to applications in which the user conducts an entire virtual orchestra. Lee refers to this use case as a "personal orchestra." Early examples of this include Max Mathews' Radio Baton [2], which enabled users to control the playback tempo and dynamics of a MIDI score using custom hardware.

Closely related to this use case are pedagogical applications, in which interactive conducting systems are used to support the learning of conducting technique by providing visual or auditory feedback on gesture accuracy and expressivity. (For an example of this, see [3]).

The present work focuses on a third application domain identified by Lee et al., namely the integration of the computer into an ensemble of human musicians. In this context, interactive conducting systems can be used to coordinate fixed media playback or live processing with human performers.

2.1.2 Input modality

Early research in interactive conducting systems predominantly employed electronically augmented batons incorporating accelerometers and gyroscopes to capture conducting gestures. Nakra et al. [4] extended this approach by employing a wearable garment with multiple embedded sensors to capture not only conducting gestures but also physiological signals, such as muscle tension. More recent research has explored computer vision-based systems that infer gesture information directly from video input. *Ponticello* adopts this approach, employing body landmark detection based on machine learning.

2.1.3 Representation of the Electronic Performance Score

While some earlier systems have taken the approach of dynamic time-stretching of a single audio file (e.g., [5]), most systems relied on MIDI-based score representations, in which conducting gestures control the timing of discrete note events [6]. While effective for the "personal orchestra" use case, these approaches significantly constrain the range of compositional strategies available in mixed music. For example, they do not support techniques such as automation, LFO-based modulation of processing parameters, or live extraction of control data from properties of acoustically produced sounds.

This is where the present approach departs fundamentally from earlier research. Rather than specifying discrete events, the electronic performance score is modeled as a collection of temporally extended electronic processes, which can be specified using the SuperCollider language. The execution tempo of these processes is continuously adapted according to the tempo indicated by the conductor. Compared to MIDI-based representations, this approach supports a broader range of compositional strategies and interactions between human performers and the computer.

2.2 Audio-based Synchronization Systems

The problem of synchronizing an electronic score with live performers has previously been addressed by audio-based score-following systems, most notably *Antescofo* [7]. The primary difference with the present system is, of course, that *Antescofo* derives timing information from the acoustic output of the human performers through machine listening, whereas *Ponticello* is based on real-time analysis of conducting gestures. As a result, the latter does not require detailed knowledge of the instrumental parts or reliable detection of pitches or onsets, but only information about meter and reference tempi for different sections of the piece.

This makes the two systems suitable for different musical scenarios. While *Antescofo* is particularly well suited for solo or small-ensemble works with electronics, especially in situations where no conductor is present or where the use of a conductor is impractical or aesthetically undesirable, *Ponticello*, by contrast, is intended for ensemble contexts in which a conductor is naturally integrated into the performance.

Because synchronization does not depend on the precise execution of the instrumental parts, *Ponticello* is also well suited for works with dense, complex, or non-standard instrumental writing that may be difficult for audio-based systems to follow reliably. In addition, the system can be used for pieces containing "electronics solo" sections, which benefit from conductor-shaped expressive timing.

2.3 Machine Learning in Interactive Conducting Systems

Since Brecht and Garnett [8], the use of machine learning has been explored for the analysis of conducting gestures. Ilmonen [9] continued this line of research, although their system still relied on heuristic components. More recently, Lee [10] presented a machine-learning-based approach to conductor following. Whereas earlier research mostly relied on *feedforward* neural networks, *Ponticello* employs a *recurrent* architecture, to explicitly model temporal dependencies in conducting gestures.

2.4 Positioning of the Present Work

Prior work in interactive conducting systems has largely focused on MIDI-based electronic performance scores, limiting the applicability of these systems in mixed music performance contexts. *Ponticello* combines tempo inference based on machine vision with a SuperCollider-based, process-oriented electronic performance score model, supporting a broad range of compositional strategies in mixed music. Compared to earlier sensor-based approaches, setup complexity and hardware requirements are reduced. This makes it a practical option for musical contexts that are not well served by audio-based synchronization systems.

3. THE ELECTRONIC PERFORMANCE SCORE

The electronic performance score is modeled as a timeline of *score objects* specifying temporally extended processes. Each score object references a *SynthDef*, which encapsulates the synthesis or signal-processing code executed during playback. In SuperCollider, SynthDefs define a signal-processing graph composed of unit generators (*UGens*). Each UGen performs a specific digital signal processing function, such as oscillation, filtering, or physical modeling, and outputs a stream of values. UGens are connected by using the output of one as the input for another, or by reading from and writing to global *buses*.

SynthDefs define a set of parameters that expose the controllable aspects of the signal processing. Each score objects maintains a set of controls that are assigned to the parameters of its associated SynthDef. Controls can take several forms:

- constant values,
- control or audio buses,
- automation curves defined over score-time.
- SuperCollider expressions evaluated at runtime. When an expression evaluates to a UGen, it is used to modulate the corresponding parameter.

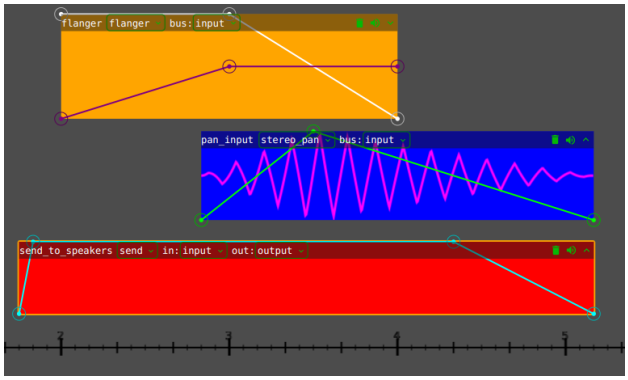

Figure 2. Section of an example electronic performance score as displayed in *Ponticello*'s graphical user interface. The tempo grid defines a 3/4-meter. Over the course of one measure, the depth of a flanger process is gradually increased. In the course of measure 3, a pan modulation process is introduced, whose modulation amplitude rises quickly and then decreases more slowly. The send object routes audio from the `input` bus to the `output`, allowing the live-processed signal to be played back through loudspeakers. The amp parameter is automated to fade-out the electronics at the end of the phrase.

In addition to regular score objects, the score contains one or more *tempo grids* that define the meter and reference tempo for the piece or individual sections of it. These grids establish a mapping from bars, beats and their subdivisions to *score-time*, which is measured in seconds and corresponds to the horizontal axis of the score. This provides the composer with orientation within the electronic performance score, relating its temporal positions to corresponding locations in the performers' parts. Tempo grids are not intended as prescriptive tempo indications for performance, but function solely as a reference, allowing the conductor to choose the performance tempo freely.

Through *Ponticello*'s graphical user interface, the user can define and edit SynthDefs, create score objects referencing these SynthDefs, reposition and resize them within the timeline, and assign or modify control mappings. In this way, *Ponticello* combines the capabilities of SuperCollider's sound synthesis model with direct interaction through a graphical score. The electronic performance score thus functions both as a compositional medium and as a performance interface.

3.1 Playback of the Electronic Performance Score

During performance, the tempo grids provide a reference against which the pulse indicated by the conductor can be interpreted by the system. A playback cursor traverses the score. Its speed is continuously adjusted according to the ratio between the tempo indicated by the conductor and the reference tempo defined by the currently active tempo grid.

When the cursor reaches the start of a score object, a corresponding *Synth* instance is created on the SuperCollider server. This instance remains active until the cursor reaches the end of the score object or playback stops, at which point the Synth is released.

The ordering of Synth nodes within the SuperCollider node tree is determined by the vertical position of the corresponding score objects, making the signal-processing order visually apparent and directly editable.

When creating and manipulating the digital score, the composer operates in terms of *score-time*, either directly or through a tempo grid. The *physical time*, on the other hand, that is required to traverse a given region of score-time, depends on the decisions of the conductor during performance. For example, if the reference tempo specified by the score is 100 BPM and the conductor indicates a tempo of 80 BPM at a given moment, the playback speed is adjusted to $\frac{80}{100} = \frac{4}{5} = 0.8$, stretching score-time by that factor. In this way, the electronic performance score becomes *elastic* during performance.

4. TEMPO INFERENCE

The tempo inference system is implemented as a real-time pipeline composed of three layers, operating on a continuous video stream of the conductor.

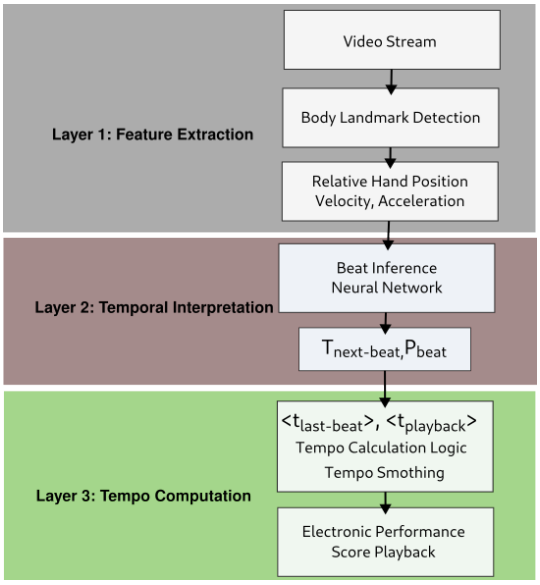

Figure 3. Overview of the tempo inference pipeline

Each video frame is first processed by a computer vision module to extract relevant body landmarks of the conductor. From these landmarks, the position, velocity, and acceleration of the conductor's right hand relative to the torso are computed, resulting in a representation suitable for temporal analysis.

The second layer constitutes the core musical intelligence of the system: a sliding window of extracted features is passed into a recurrent neural network, which is trained to predict two variables:

- $T_{next-beat}$, the estimated physical time interval until the conductor indicates the next beat.
- P_{beat}, the probability that the current frame corresponds to a beat.

The third layer is responsible for adjusting the playback speed of the digital score. It maintains two state variables:

- $t_{last-beat}$, the score-time position of the last recognized beat within the digital score.
- $t_{playback}$, the current position of the playback cursor within the digital score. It is continuosly updated based on the playback speed Δt computed by the tempo inference mechanism, making the two variables mutually dependent.

Based on these variables and the predictions of the neural network, the system continuously adjusts the playback speed of the digital score. At each video frame, the system computes the time interval from the current playback position to the next beat of the active tempo grid. The playback speed is then derived as the ratio between this score-time distance and the time interval to the next indicated beat as predicted by the neural network. To reduce noise, the playback speed is smoothed using an exponential filter, with the smoothing factor α configurable by the user.

Algorithm 1: Tempo inference pipeline.

$seq \leftarrow []$ `//sliding window of feature vectors`
$\Delta t \leftarrow 1$
for each received video frame **do**
  $(x, y) \leftarrow$ estimate relative right hand position
  $(\Delta x, \Delta y) \leftarrow$ calculate velocity
  $(\Delta^2 x, \Delta^2 y) \leftarrow$ calculate acceleration
  append $(x, y, \Delta x, \Delta y, \Delta^2 x, \Delta^2 y)$ to seq
  `//`N `denotes the feature window length`
  if $len(seq) \neq N$ **then**
    $(P_{beat}, T_{next-beat}) \leftarrow$ NN.predict(seq)
    `//`$tempo_{ref}$ `refers to the tempo defined by the active tempo grid`
    $t_{next-beat} \leftarrow t_{last-beat} + \frac{60}{tempo_{ref}}$
    $\Delta t_{next-beat} \leftarrow t_{next-beat} - t_{playback}$
    $\Delta t \leftarrow \frac{\Delta t_{next-beat}}{T_{next-beat}} \cdot \alpha + \Delta t \cdot (1 - \alpha)$
    remove first item from seq
    $dt_{phys} = wall\_clock\_time() - timestamp$
    if $P_{beat} \geq threshold$ **and** $dt_{phys} > min\_beat\_dur$ **then**
      $t_{last-beat} \leftarrow t_{next-beat}$
      $timestamp \leftarrow wall\_clock\_time()$
    end
  end
end

5. IMPLEMENTATION DETAILS

5.1 Programming Languages and Frameworks

The tempo inference system is implemented in Python using the Python API of Google's *MediaPipe* library, which provides pre-trained models for body landmark detection.

Ponticello's electronic performance score model and graphical user interface is implemented in *Kotlin*, a general-purpose, JVM-compatible programming language.

Communication between the performance score model, the tempo inference process, and SuperCollider is handled via OSC. The system communicates not directly with the synthesis server (*scsynth*), but with an *sclang* process to which it sends SuperCollider expressions for evaluation.

5.2 Dynamic Audio Time Stretching

To synchronize the playback of audio buffers with the conductor's gestures, the system must be able to adjust the buffer's playback rate continuously while preserving pitch. This is achieved through granular time stretching, using SuperCollider's *Warp1* unit generator.

A playback cursor moves through the audio buffer at a rate determined by the conductor's tempo. Rather than reading the buffer sequentially at a fixed speed, the system continuously extracts short grains from the buffer at the current cursor position. These grains are then recombined to produce a continuous audio signal.

This method allows audio buffers to expand or contract in real-time in response to conducting gestures without altering pitch. As a result, playback remains tightly synchronized with the ensemble even under significant tempo variation. (Granulation parameters, such as grain duration and density, provide compositional control over the texture of the resulting sound.)

5.3 Elastic Audio Recording

In some mixed music pieces, audio material recorded during the performance is reintroduced at later points in the piece. This presents an additional challenge with regards to synchronization when musical time is shaped freely by the ensemble. If recorded material is later replayed under a different conducted tempo, just time-stretching the audio material according to the new tempo fails to preserve the relative timing relationships.

To address this, the system captures a mapping between score-time and physical time while recording audio to be used during playback. During recording, audio is written into a buffer normally, while a continuous mapping from score-time to physical time is simultaneously stored in a separate buffer.

Algorithm 2: Recording the time mapping buffer

$t_{score} \leftarrow 0$;
$t_{phys} \leftarrow 0$;
for each audio frame **do**
  $T_{phys}[t_{score}] \leftarrow t_{phys}$
  $t_{score} \leftarrow t_{score} + \Delta t$
  $t_{phys} \leftarrow t_{phys} + \frac{1}{\text{audio framerate}}$
end

During playback, score-time advances according to the currently conducted tempo, and the grain index into the audio buffer is determined via the stored mapping. From the current score-time position the corresponding physical-time position is interpolated, which is then used to extract grains from the audio buffer.

Algorithm 3: Playback of live recorded buffers

$t_{score} \leftarrow 0$
for *each audio frame* **do**
  $t_{phys} \leftarrow$ interpolate $T_{phys}(t_{score})$
  move grain position cursor to t_{phys}
  $t_{score} \leftarrow t_{score} + \Delta t$
end

As a result, the relative temporal structure of the recorded material is preserved, while the overall playback adapts smoothly to the current conducted tempo.

This mechanism is implemented using two custom unit generators written in C++, one capturing the mapping from score-time to physical time in a separate buffer during recording and the other reading from that buffer and calculating the appropriate grain positions in the audio buffer during playback.

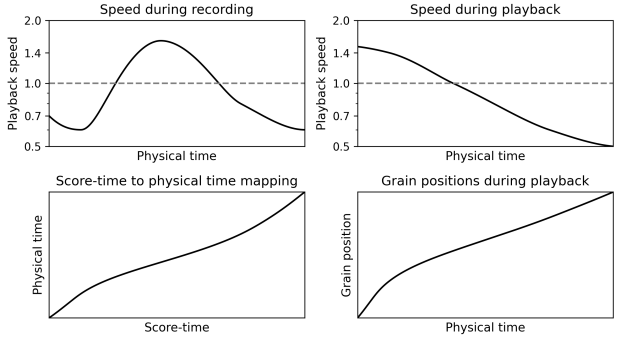

Figure 4. The example shows two tempo curves at the top: one captured during the recording of an audio buffer and the other during playback. The lower-left graph depicts the time-mapping function derived from the recorded tempo curve. The lower-right graph shows the grain index into the audio buffer during playback.

5.4 Neural Network Architecture

The model takes as input a sequence of feature vectors extracted from consecutive video frames and outputs a pair of values $(P_{beat}, T_{next-beat})$, as defined in section 4. During training, target values are derived from manually annotated beat positions in the corresponding video recordings.

The choice of feature window length involves a trade-off. Increasing the window size allows the model to incorporate a broader temporal context, but also results in increased training time and requires more frames to be accumulated before the first prediction can be produced during performance. Experiments indicate that a sequence length corresponding to approximately five seconds of conducting gesture data is sufficient for optimal results in most contexts.

Beat prediction from postural features is inherently a temporal inference problem. Recurrent neural networks are therefore well suited to this task, as they can model temporal dependencies in sequential feature data.

Among recurrent neural network architectures, Gated Recurrent Units (GRUs) offer a practical trade-off between model capacity and training effort. Their gating structure avoids the instability and vanishing-gradient problems commonly encountered with simple recurrent networks. Compared to more complex architectures such as LSTMs, GRUs require fewer parameters and thus typically less training data to achieve reliable performance,

6. DISCUSSION

6.1 Use in Rehearsal and Concert Situations

Through the system's graphical user interface, the user can select the trained neural network model file (.pth) used for beat inference during performance and configure associated parameters such as the smoothing factor, beat confidence threshold, and minimum beat distance.

Before starting the performance, the conductor can also select the desired starting point in the digital score by positioning the playback cursor at a specific bar. This allows sections of the piece to be rehearsed without restarting from the beginning. The system is initiated using a MIDI foot pedal. Following this trigger, the conductor conducts one full bar in advance, allowing the system and the musicians to establish a common tempo before the acoustic performance begins and the electronic playback starts.

6.2 Tempo Inference Latency

In practice, the system's real-time latency depends almost entirely on the frame rate of the input video stream. The additional latency introduced by body landmark extraction, neural network inference, and the tempo calculation is negligible by comparison. When using a standard USB video camera operating at 30 frames per second, the resulting latency is approximately 35 ms. This latency does not correspond to a constant temporal offset between the system and the performers, but instead reflects the time required for the system to respond to tempo changes indicated by the conductor, making it acceptable in a musical context.

6.3 Reliability and precision

In both rehearsal and performance, the system has demonstrated stable and reliable behavior, maintaining consistent synchronization between the ensemble and the electronic processes over extended musical passages. Given sufficient training data, the system very rarely misses beats, making false negatives highly improbable.

To reduce the occurrence of false positive beat detections, the system allows specifying a minimum time interval between consecutive beats. This parameter can be adjusted by the user according to the maximum tempo expected during performance.

The system achieves its highest accuracy for conducting styles that are represented in the training data, which currently comprises approximately three hours of video material. Initial tests indicate, however, that adaptation to new conductors or conducting styles can be achieved with relatively little additional data: approximately twenty minutes of annotated video material are sufficient to fine-tune the model for reliable operation.

7. CONCLUSION

This paper has presented an electronic performance score model and a system that supports synchronization between live performers and the playback of an electronic performance score, without constraining the ensemble's control over musical timing. By following the conductor's tempo, the electronic performance score becomes elastic during performance, in direct analogy to the temporal flexibility found in the interpretation of music notated for human musicians. Just as meter and rhythm in an instrumental score often provide a structural framework rather than a rigid prescription, the electronic performance score defines a relative temporal structure whose realization in physical time is shaped continuously by the conductor.

Rather than treating artificial intelligence as a replacement for human creativity, the approach employs neural networks as a mediating layer between human musical gestures and computational models. Advances in computer vision and machine learning facilitate the implementation of reliable, camera-based conducting interfaces that avoid the need for specialized or expensive hardware. This makes the technology accessible beyond institutions equipped with custom performance technology.

Ponticello has been used both in rehearsal settings and in concert performances and has proven sufficiently reliable and accurate across multiple conductors. It can be adapted with relatively little effort to conducting styles not represented in the core training data.

This opens up possibilities for composing new works of mixed music that require both tight synchronization between acoustic and electronic layers and the freedom for performers to expressively shape musical time. It also allows for the adoption of existing mixed music repertoire that can benefit aesthetically from a more flexible treatment of musical time.

8. FUTURE WORK

Future work will focus on expanding the training dataset to include a broader range of conducting styles. As the diversity of conducting styles included in the training set increases, the system's ability to generalize across different performers is expected to improve correspondingly.

The system's reliability may be further improved by considering the position of beats inside the measure. First beats inside a measure are typically indicated in a clearly distinct way by most conductors and thus should establish recovery points when beats are missed.

Another area of investigation concerns the integration of gestures commonly used in rehearsal. Conductors sometimes employ specific gestures to stop an ensemble, return to a previous rehearsal point, or restart from the beginning. Extending the system to recognize such gestures would improve its practical usability in rehearsal situations.

While the current system focuses on the temporal coordination aspects of conducting, conducting gestures convey additional musical information that could be incorporated into the model. Future extensions could aim to infer parameters such as dynamics and articulation from gesture features, allowing electronic processes to react not only to the conductor's tempo but also to other parameters indicated through conducting gestures.

9. REFERENCES

[1] K. Lee, M. Junokas, and G. Garnett, "A Review of Interactive Conducting Systems: 1970-2015," in *Proceedings of the 2015 International Computer Music Conference*, Utrecht, The Netherlands, 2016, pp. 485–491.

[2] M. V. Mathews, "The radio baton and conductor program, or, pitch, the most important and least expressive part of music," *Computer Music Journal*, vol. 15, pp. 37–46, 1991.

[3] G. Garnett, F. Malvar-Ruiz, and F. Stoltzfus, "Virtual Conducting Practice Environment," in *Proceedings of the 1999 International Computer Music Conference*, Beijing, China, 1999, pp. 371–374.

[4] T. M. Nakra, T. Machover, and R. W. Picard, "Inside the conductor's jacket: analysis, interpretation and musical synthesis of expressive gesture," 2000. [Online]. Available: https://api.semanticscholar.org/CorpusID:17902572

[5] J. O. Borchers, E. Lee, W. Samminger, and M. Mühlhäuser, "Personal orchestra: a real-time audio/video system for interactive conducting," *Multimedia Systems*, vol. 9, pp. 458–465, 2004.

[6] S. Bergen and T. Lokki, "Conductor Follower: Controlling sample-based synthesis with expressive gestural input," 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:20546548

[7] A. Cont, "ANTESCOFO: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music." in *Proceedings of the 2008 International Computer Music Conference*, Belfast, Ireland, 2008, pp. 33–40.

[8] B. Brecht and G. E. Garnett, "Conductor Following," in *Proceedings of the 1995 International Computer Music Conference*, Banff, AB, Canada, 1995, pp. 185–186.

[9] T. Ilmonen and T. Takala, "Conductor Following With Artificial Neural Networks," in *Proceedings of the 1999 International Computer Music Conference*, Beijing, China, 1999, pp. 367–370.

[10] M. Lee, "Deep Neural Network based Music Source Conducting System," in *Proceedings of the 2018 International Computer Music Conference*, Daegu, South Korea, 2018, pp. 17–20.