Multimedia Systems 9: 458–465 (2004)
Digital Object Identifier (DOI) 10.1007/s00530-003-0119-y

Multimedia Systems

Personal orchestra: a real-time audio/video system for interactive conducting

Jan Borchers[1], **Eric Lee**[1], **Wolfgang Samminger**[2], **Max Mühlhäuser**[3]

[1] Media Computing Group, Department of Computer Science, RWTH Aachen 52056, Aachen, Germany
e-mail: borchers@cs.rwth-aachen.de
[2] Telecooperation Department, University of Linz, 4040 Linz, Austria
e-mail: wolfgang.samminger@liwest.at
[3] Telecooperation Group, Darmstadt University, D-64283 Darmstadt, Germany
e-mail: max@informatik.tu-darmstadt.de

Abstract. We present the first multimedia system to conduct a realistic electronic orchestra. Users can control tempo, dynamics, and instrument emphasis of the orchestra through natural conducting gestures with an infrared baton. Using gesture recognition and tempo adjustment algorithms, the system plays back an audio and video recording of an actual orchestra that follows the user's conducting in real time. A major achievement of this system is its ability to vary playback speed in real time while avoiding audio artifacts such as pitch changes. The system has been deployed as an exhibit and has become a major attraction of a large Vienna-based music exhibition center.

Keywords: Conducting – Interactive exhibit – Time stretching – Music – Design patterns

1 Introduction

In contrast to the rapidly growing abilities of today's computing architectures in recording and playing back multimedia contents, innovation in the area of interacting with these data streams has fallen behind. For example, there are many established ways of interacting with music (such as humming, improvising, or conducting) that could offer users powerful new ways to access and control multimedia, but they are rarely available in mainstream systems that claim to be "multimedia enabled". The user interface, therefore, is rapidly becoming the bottleneck when it comes to new ways of deploying computing technologies.

Conducting is one good example: people frequently enjoy acting as if they were conducting a classical piece played back from a CD. Of course, they are really just conducting alongside the fixed recording, without actually influencing the performance. The experience would be much more realistic if the user could indeed *control* the orchestra playing or, even better, if the user could see, as well as hear, the orchestra playing. The goal of this work is to create such a system, where the user can interact with digital media in a novel but natural way.

2 Requirements

Such a system suggests the following basic requirements:

Conducting device: The system requires a batonlike device to allow the user control of the orchestra in a simple but natural way. Since users in general cannot be expected to know professional conducting patterns, the system needs to be able to interpret a simple one-hand, up-down conducting motion.

Gesture recognition: Input from the baton device needs to be analyzed to determine the desired volume, tempo, and emphasis on certain instrument sections (if any).

Time-stretching algorithm: The audio/video recording of the orchestra needs to be played back at varying speeds, say, from 50% to 200% of the original tempo. However, changing the playback speed must not produce any noticeable artifacts such as changes in audio pitch.

The following additional requirements arose because the system was to be deployed in the HOUSE OF MUSIC VIENNA [2], a major music museum and exhibition center in Vienna, Austria:

Audio quality: The system was to feature a prestigious orchestra (the Vienna Philharmonic), and thus the audio quality could not be compromised.

Usability: As a system to be deployed in a public museum, the system design had to ensure a minimal learning curve and immediate usability by a wide variety of users. As an interactive exhibit, the system also had to fulfill other requirements that are particularly important in this class of systems, such as a nontechnical, innovative appearance, universal accessibility, and robustness [3].

3 Usage scenario

The exhibit itself was designed to re-create the Vienna Philharmonic's concert hall, complete with props such as tapestries, traditional note stands, and, of course, the conductor's podium. A large rear video projection shows the orchestra, softly rehearsing, waiting for a conductor to become active (Fig. 1).

When a user picks up the infrared baton and activates it by pressing the button on it, she is guided through two menus to

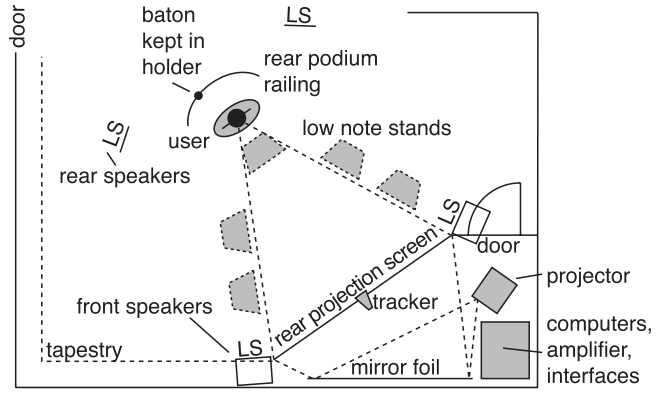

Fig. 1. Overview of the *Personal Orchestra* exhibit space

allow language and piece selection. Once a piece is selected, the orchestra appears on the screen, waiting. When the user begins to conduct, the orchestra starts playing, following the user's gestures. The players continue until the piece is over, at which point they rise to congratulate the conductor with applause from the audience. The players also stop if the user continuously conducts badly (see below).

Downward turning points of the baton trajectory identify the conductor's beats. In accordance with traditional conducting [12], vertical size (amplitude) of the conducting gesture controls volume and horizontal direction (conducting "toward" certain instrument sections) lets the user raise that section above the rest of the orchestra.

Users should not be allowed to conduct too slowly or too quickly. There are two reasons for this. Technically, the audio quality diminishes as the time-stretching factor increases. Also, the Vienna Philharmonic, being a prestigious group, would not appreciate being presented in a silly manner, which would occur if the user could conduct at arbitrarily fast speeds. To preserve the immersive experience of the exhibit, we chose to use a natural and realistic error message: if the user "teases" the orchestra too much by conducting very quickly, slowly, or stopping completely, the orchestra reacts by stopping their playing, and one player gets up to complain about the conductor's skills. The system includes suitable tolerance rules for the orchestra, detects conducting that breaks these rules, and shows the corresponding complaint sequence without noticeable interruptions.

4 System design

4.1 Design patterns

Our software design was based on a set of *human-computer interaction* (HCI) *design patterns* for interactive exhibits [4]. These HCI design patterns capture principles and guidelines of interaction design for this class of systems.

Each pattern is a textual and graphical description of a successful solution to a recurring usability problem in interactive exhibits and contains the same components: its *name* is used to refer to the pattern easily and create a vocabulary for the design team, its *ranking* shows how valid and universal the author considers the pattern, and a *sensitizing example* shows a picture of a real interface to illustrate the idea that the pattern captures. This is followed by a *problem statement* explaining what UI design problem the pattern addresses, and a set of *examples* or other empirical results are then used to show how this problem has been solved in similar ways in different systems.

These examples are generalized into the *solution*, a more reusable design guideline for the problem of this pattern. A *diagram* shows the essential idea of the solution in graphical form. Each pattern also refers to its *context* (when it should be applied) by pointers from other patterns in the language that address larger-scale design issues, and it itself refers in turn to smaller-scale patterns (its *references*) to consider next in order to implement and further unfold the design solution that this pattern suggests.

As an example, the EASY HANDOVER pattern from that language makes the following design recommendation:

- At interactive exhibits, one user often takes over from the previous one, possibly in the middle of the interaction and without necessarily having observed or knowing much about the interaction history of his predecessor.
- Therefore, minimize the dialogue history that a new user needs to know to begin using the interactive exhibit. Offer an obvious way to return the system to its initial state. Let users change critical, user-specific parameters (such as language) at any time during the interaction.

Of course, this list is just the essence (the problem and solution statements) of the EASY HANDOVER pattern. The entire pattern consists of three pages of text and graphics, including examples of existing systems using this solution successfully, context and reference pointers to other patterns in the language, and all other pattern constituents listed earlier.

Nevertheless, this excerpt should convey an idea of how these patterns were able to help us design *Personal Orchestra*: for example, we used the above pattern to decide that no special gestures or other actions should be necessary anywhere during the conducting to stop or otherwise control the exhibit. While these gestures could have been explained in the initial opening screens, there was no guarantee that any particular user would have actually seen those instructions.

While further discussion of this approach is beyond the scope of this paper, it is explained in detail in [4].

4.2 System architecture

Figure 2 shows the resulting system architecture. The visitor conducts using an infrared baton whose signals are picked up by a tracker and sent to the *POServer* machine. There, tempo, volume, and orchestra section emphases are determined by gesture recognition and prediction. This "heartbeat" information is sent via our TCP-based *Personal Orchestra Control Protocol* (POCP) to the *POClient* computer, which renders the selected piece accordingly in audio and video, permanently adjusting playback parameters to follow the conducting.

During the initial selection of language and piece and upon finishing or breaking off a piece, POServer sends similar POCP commands to POClient to display the corresponding screens and movie sequences. POCP is a simple, HTTP-like protocol that sends textual messages about the current speed, volume, instrument emphasis, and state from the server to the client and returns movie positions from the client.

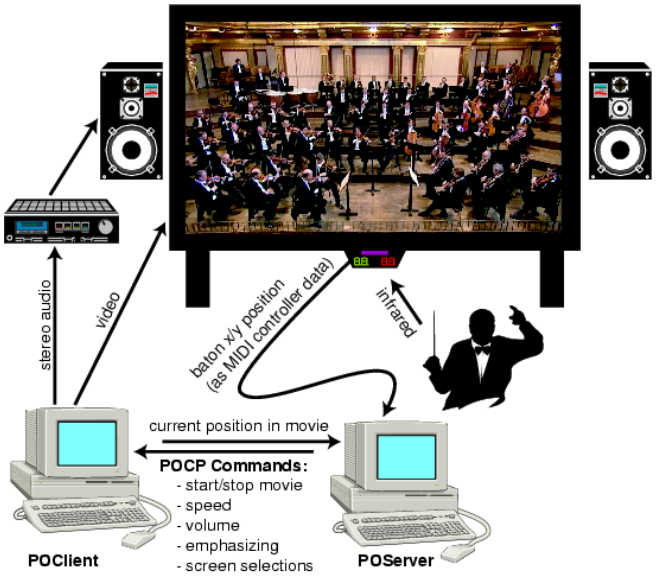

Fig. 2. *Personal Orchestra* system architecture

4.3 Input technology

We used Don Buchla's *Lightning II* infrared baton system [5]. It translates input from the infrared-emitting, battery-operated baton received by a tracker mounted below the screen, into MIDI controller signals representing x/y baton coordinates at a resolution of 7 bits per value. A third, binary signal represents the baton button.

4.4 Gesture recognition and prediction

From continuously monitoring the position of the baton, its current x/y position as well as approximations for its first derivatives are known. Every time the system detects a downward turning point in the gesture (negative-to-positive sign change of the first derivative of the y coordinate in the baton trajectory), it is interpreted as a "downbeat". These downbeats correspond to a series of positions in the movie premarked manually as the "beats" in the music.

The current playback speed is then adjusted so that the orchestra always follows the conductor. There are two major problems with this scheme, however. First, while a conductor may be conducting at the same speed as the orchestra plays, the two may be out of *phase* – for example, the orchestra would always play their "beat" half a beat after the conductor's downward beat gesture.

Second, when a conductor changes the tempo, a part of the current beat has not been played yet when the next, first conductor beat gesture at the new tempo arrives: the orchestra has to "catch up" with the conductor (Fig. 3). To resynchronize with the conductor, playback speed has to be increased above the new target (measured) conducting rate until the movie and conductor are at the same point in the piece, at which the tempo levels off to the actual new conducting speed. The following formulae describe these relations:

Let b_r be the time when the last, and b'_r the time ("*r*eal time") when the previous beat has been conducted by the user. Similarly, let b_m and b'_m be the playback position ("*m*ovie time") in the movie at time b_r and b'_r, respectively. Then the relative velocity (tempo) with which the user is conducting the movie is

$$v_u = \frac{b_m - b'_m}{b_r - b'_r}.$$

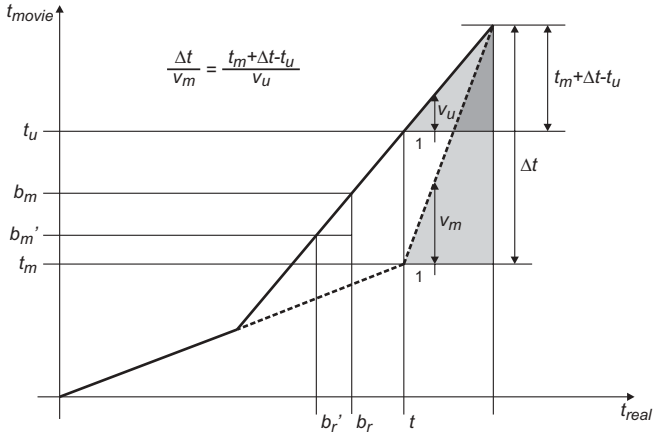

Fig. 3. When a conductor speeds up, the orchestra has to play faster than the new tempo to get back in phase

Under the realistic assumption that, within a single conducted beat, the conducted tempo does not change dramatically, the current position t_u to which the user has conducted the movie at time t now equals

$$t_u = b_m + v_u \cdot (t - b_r).$$

Then, if the movie is currently at position t_m, the new relative velocity v_m of the movie ($v_m = 1$ for the originally recorded tempo) to catch up with the conductor within a time window of Δt is

$$v_m = \frac{\Delta t \cdot v_u}{t_m + \Delta t - t_u}.$$

Of course, since this adjustment happens at every beat, the catch interval is not used in its entirety if it is longer than one beat; instead, the movie speed will gradually converge back to the new conducted speed, reducing its overestimate in a series of adjustments.

The larger the time window Δt in which this catching up happens is chosen, the more the orchestra creates the impression of being "slow to catch up" — it does not respond immediately to a tempo change, but rather over time, and it takes the orchestra longer to get back in sync with the conductor. The advantage is that short tempo jitter by inexperienced conductors does get filtered out; the orchestra is more "benign" and tolerant toward such errors. We left this parameter as a variable that can be changed at runtime.

A similar low-pass filter was implemented for volume control. To make the system more tolerant against conducting glitches, abrupt changes in volume are prevented by calculating the desired new volume vol' with the following formulae out of the previously calculated volume vol and the volume vol_u conducted by the user, where the values of volume are in the range $[0; 1]$:

$$vol' = vol + \frac{vol_u - vol}{2}, \; if \; |vol_u - vol| > 0.1$$

$$vol' = vol,\ else.$$

4.5 High-quality interactive audio/video time stretching

A broadcast-quality DigiBeta (digital betacam) video camera fixed to a position resembling the view of the conductor recorded the orchestra playing various pieces without a conductor. Its output was converted to AVID, a computer-compatible digital video format. Microphones throughout the orchestra recorded the various instrument sections onto ADAT digital audio tape. One challenge was how to adjust, or *time-stretch*, the playback speed of the orchestra movie.

Time-stretching video is fairly straightforward; most multimedia libraries can handle changes in playback speed by holding or dropping frames. As long as these variations do not drop below animation frame rates (around 12 fps), and as long as there is no extreme movement that would create jerkiness at higher-than-normal speed (which is not the case with an image of an orchestra sitting and playing), the change of video playback speed creates no critical artifacts. While non-standard playback speed creates unnatural movements (such as with respect to gravity — objects falling at slower than normal speed), this was also not critical with our scenery.

The audio track, on the other hand, is a more complex problem since changing only the playback speed of an audio recording will also change its pitch (an effect easily demonstrated by choosing a different speed on an analog record player).

Various algorithms exist that time-stretch audio in real time. However, at the time this system was being designed, there were no algorithms available that could run in real time while preserving sufficient sound quality of the original recording.

Thus, we instead opted to prestretch all our audio channels at various speeds and then, for each channel in parallel, crossfade between its prestretched versions to change playback speed. Since the processing is done offline, we could take as much time as necessary to produce the best possible audio quality. During playback, we simply determined the new required tempo and smoothly crossfaded all four audio channels from their current tempo track over to their newly selected one within a few milliseconds. The crossfade is necessary to avoid audible clicks that would occur from audio waveform discontinuities caused by immediate track changes.

Unfortunately, this approach introduced different time coordinate systems for each audio track. To avoid this, and benefit from the system support of a single movie file with one video and multiple audio tracks, we *pitch-shifted* the audio between -1 and $+1$ octave, in 2-half-tone steps of a factor of $2 \cdot \sqrt[12]{2}$. Playing back, for example, an audio recording that has been pitch-shifted down 1 octave at double speed returns the original pitch at double tempo. This way, we were able to integrate all pitch-shifted audio tracks with the video track, and a tempo change simply meant fading over to the appropriate audio track and simultaneously changing playback speed of the entire movie to bring that audio track back to its original pitch. We used Prosoniq's commercial high-quality pitch-shifting software package *TimeFactory*, which utilizes the proprietary *MPEX* algorithm (Minimum Perceived Loss Time Expansion/Compression) [1].

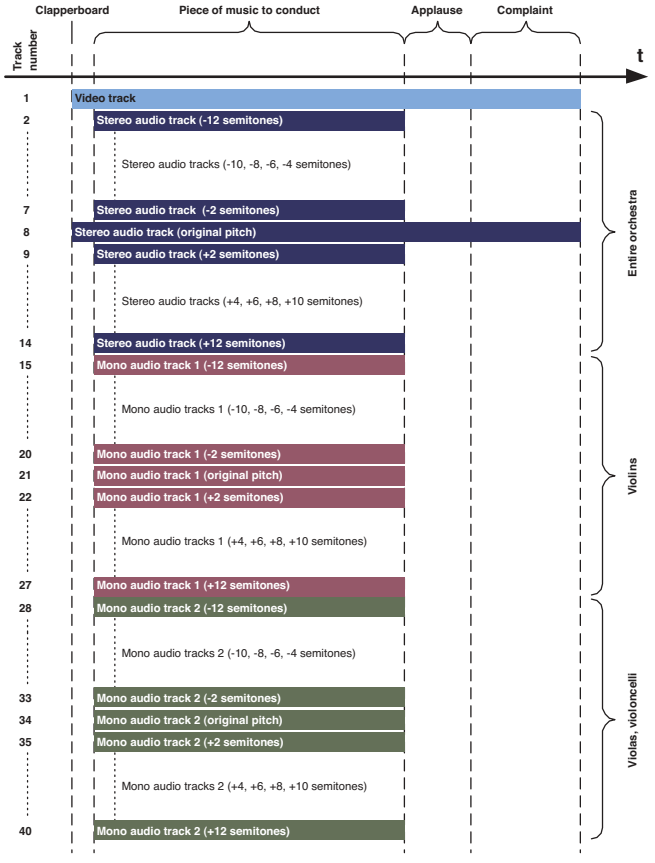

Fig. 4. Layout of audio and video data streams in a single movie file

The emphasis between different instrument sections was simply implemented by pitch-shifting our four recorded and premixed instrument section channels separately and mixing them according to emphasis during playback in real time.

Figure 4 shows the structure of a typical audio/video data file, representing one musical piece, and stored as a QuickTime movie. Track 1 carries the video, and tracks 2–14 carry the full-orchestra stereo audio track, ordered from lowest to highest pitch (in increments of $\sqrt[12]{2}$, i.e., semitones). Tracks 15–27 and 28–40 carry audio data, in the same pitch order, for the two instrument sections that can be emphasized. The initial part of the movie only contains a few seconds of data in the video and the original stereo audio track, showing the orchestra mounting their instruments (synchronized using a clapperboard in the original raw footage) until the moment the orchestra begins to play. The main part of the movie shows the orchestra playing and contains data in the video and all audio tracks. At the end of the piece, only the video and original stereo track continue, containing the applause scene, followed by a separate piece of footage showing the complaint scene. The system jumps to this last scene when the conductor performs too badly.

4.6 Protocol

As shown in the system architecture, the entire system was distributed between two machines connected over a local network. The POServer handles gesture recognition and

tempo/volume/emphasis computation, while POClient uses this information to render the audio and video of the orchestra.

Our Personal Orchestra Control Protocol (POCP) defines the messages that the gesture recognition component (POServer) could exchange with the audio/video rendering component (POClient). It was designed to be lightweight and thus we could afford to make the protocol human-readable, with the advantage of simplifying development, debugging, and maintenance. The following commands were specified:

SPD i j: Set playback speed to integer value i (1 = 50% ... 13 = 200% of normal speed; see the audio track setup for the actual speeds), and, if given, set instrument emphasis to section j (0=none, 1=timpani, 2=horns left, 3=horns right, 4=violoncelli, 5=violins, 6=celli; 0 is the default; currently only 0, 5, and 6 are used.)
VOL f: Set overall volume to f $\in [0, 1]$.
JMP i: Jump to integer position i (in 1/600 s) in the movie.
STP: Stop playback immediately.
LDM i: Load movie with index i, display first frame. Our sample exhibit features four pieces: 1 = Radetzky March, 2 = Blue Danube, 3 = Eine Kleine Nachtmusik, and 4 = Anna Polka.
LDI: Load introductory QuickTime movie, display its first frame.
SEL i: Display screen i (used during piece selection and to show information screen, currently five screens are used).
INF: Same as SEL 5, showing information screen.
THX: Display "Thank You" screen after conducting piece successfully.
LNG (DE|EN): Switch to German (DE) or English (EN) as current interface language, and show the current screen in the new language.

To illustrate the use of the POCP protocol, Fig. 5 shows a communication scenario between POServer and POClient.

This scenario is described as follows:

Upon startup, the POClient loads and plays the idle loop automatically. The POServer monitors the current playback position in the idle loop and sends commands that tell POClient to continuously loop the idle movie (LDI).

When the user presses the baton button, POServer tells POClient to display the language selection screen. The user selects German (LNG DE), at which point POServer tells POClient to show the piece selection screen highlighting the second piece (chosen because this item corresponds to the current height of the baton).

The user moves the baton to the bottom; POServer sends a SEL 5 message, and POClient highlights the lowest selection (the link to the information screen). By pressing the baton button, the user selects that link and POClient is instructed to display that screen (INF).

After reading that page, the user presses the baton button again (anywhere on the screen), leading back to the selection of a piece (SEL 4). The user finally picks piece number 3 (SEL 3).

The corresponding QuickTime movie is loaded (LDM 3), and POClient jumps to where the conductable video in that movie starts (JMP 3672, determined from the configuration text file) and sets volume to 70% (VOL 0.7). The user begins

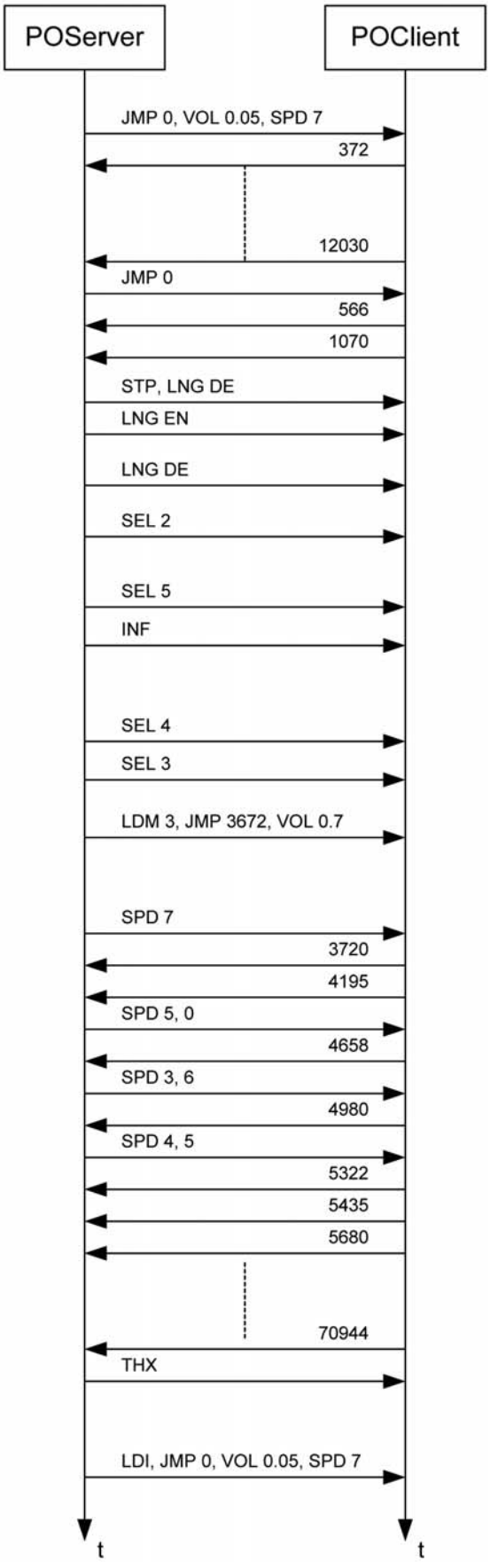

Fig. 5. Sample communication through the Personal Orchestra Control Protocol (POCP)

to conduct, and POServer recognizes the first gesture and starts the video at normal tempo (SPD 7).

In order for POServer to synchronize the audio/video playback using tempo changes (SPD 5.0 – SPD 3.6 – SPD 4.5 – . . .), POClient continues to send the current position in the movie (3720, 4195, 4658, . . .).

In this example, the user manages to complete the piece without the orchestra complaining and POServer detects a movie position that indicates the piece is over and instructs POClient to switch to the final page (THX), which congratulates the user. After several seconds, the system returns to the idle loop (LDI, JMP 0, VOL 0.05, SPD 7) and is ready for the next user.

4.7 Navigation

As indicated earlier, *Personal Orchestra* plays a movie loop of the orchestra rehearsing until a user picks up the baton and presses the button on it. The orchestra disappears, and the user selects his favorite language and piece by moving the baton up and down and pressing the button.

After selecting a piece, the orchestra appears again, waiting for the user to start conducting. The conducting ends with either the orchestra complaining when the conducting is too bad for several beats in a row or the end of the piece, at which point the orchestra rises and a big round of applause from the invisible audience behind the conductor can be heard. The state diagram of the system is shown in Fig. 6.

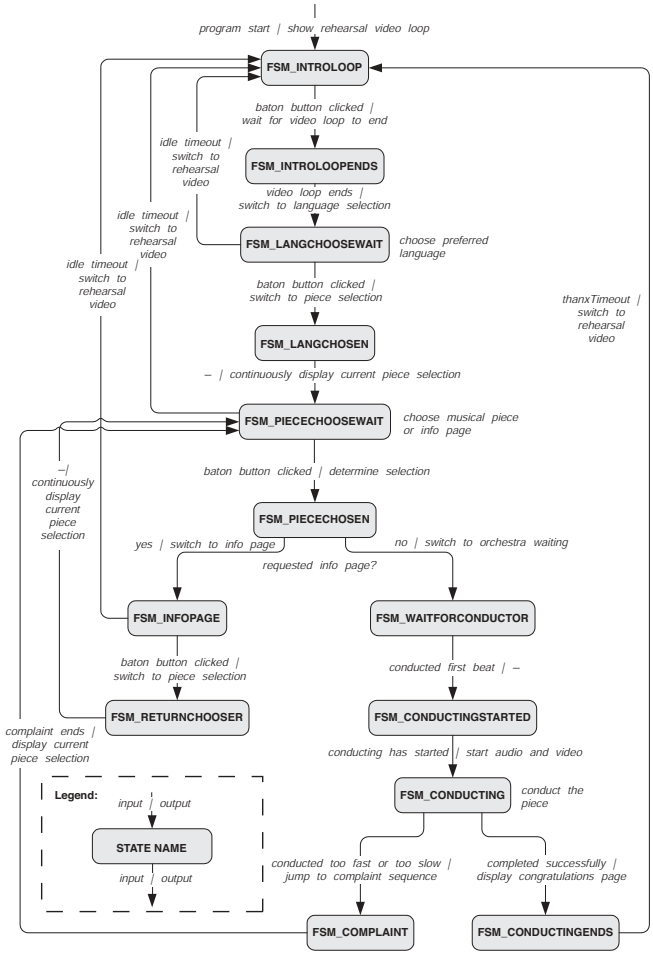

Fig. 6. The finite state machine of *POServer*

4.8 Hardware and software

The client and server software was implemented in Java. After initial experiments with Microsoft Windows and its DirectX/DirectMedia interfaces, we decided to use two Apple Power Macs G4/500 running Mac OS 9 and QuickTime, since they provided a more appropriate multimedia environment for our particular development and exhibition needs. All audio tracks and the compressed video (using QuickTime's Sorenson video codec at a resolution of 716×288 pixels) for each piece are contained in the combined QuickTime movie file and streamed directly off the hard disk. The maximum data rate of the material is around 3.6 MB per second, leading to each musical piece (between 3 and 5 min) being represented by a QuickTime movie file with a size between 600 MB and 1 GB.

Video is projected via a rear projector attached to POClient; audio is fed from the same machine into a high-end speaker setup with two front and two rear speakers and a subwoofer to enable sound locating as well as creating an audio ambiance that fills the room.

5 Evaluation

5.1 User observations

In addition to user feedback during the iterative design and prototyping of *Personal Orchestra*, we conducted several studies of visitors using the exhibit in the first few weeks after the opening of the HOUSE OF MUSIC VIENNA. Our initial, qualitative observations showed that the orchestra would stop and complain while the user was just getting acquainted with the system or shortly before finishing a piece. We introduced safety zones at the first and last few seconds of each piece to avoid these frustrating error situations. We also discovered that users often did not read the signage at the exhibit and instead just looked at the idle loop, wondering what to do. We therefore rendered a single sentence into the idle loop encouraging the user to pick up the baton.

After those improvements, we did another study where we observed and then interviewed 30 random users between 9 and 67 years, with a wide variety of reported educational, musical, and computing backgrounds. The average user tried to conduct 2.4 pieces. Average usage time was 5.9 min. Ninety-seven percent of all users managed to do basic conducting gestures recognized by the system. Ninety-three percent of all users realized that they could control tempo, 77% that they could control volume, and 37% that they could control emphasis of instrument sections. Sixty percent of all users managed to finish conducting a piece without errors, while 27% did so on their first attempt. On a scale of "good", "mediocre", and "bad", 81% judged audio quality to be "good", the remaining 19% "mediocre". Video quality was judged "good" by 75%, "mediocre" by 21%, and "bad" by 4% (one user). Ninety-three percent voted the exhibit to be a top three exhibit in the HOUSE OF MUSIC VIENNA.

6 Related work

There is a large body of research in systems that follow human conducting [3,6,9,11]. However, these systems share one or more of the following characteristics, rendering them unsatisfactory for our purposes:

- They are mostly designed to interpret professional conducting styles, which does not match the skills of our target user group of museum visitors.
- While many of them focus on optimizing their gesture recognition, they do not pay the same attention to their output quality. Instead, they use synthesized sounds such as MIDI, which makes it virtually impossible to re-create the unique sound of a specific, renowned orchestra such as the Vienna Philharmonic playing in their Golden Hall.
- These systems mostly do not provide a natural video rendition of the orchestra playing – a critical feature of the experience we wanted to provide.

7 Current and future work

There are several research topics that we are currently working on, building on our experience with the Personal Orchestra project.

7.1 Real-time audio stretching of arbitrary factor

As outlined in Sect. 4.5, the Personal Orchestra system relies on prestretched audio tracks, which introduces a number of limitations. Most notably, the system is limited in the number of available playback speeds as well as the rate at which adjustments can be made (currently once per second). It is desirable to have a broad range of playback speeds, continuously adjusted to match the user input. Such a feature is only feasible with a real-time time-stretching algorithm; with the advances in technology since the original design, it is now possible to perform such processing on consumer hardware without compromising audio quality.

There are essentially two main classes of audio time-stretching algorithms used for real-time applications: time domain and frequency domain. Time domain techniques, while less computationally intensive, unfortunately work well for only small scaling factors. Frequency domain techniques are much better suited to the Personal Orchestra despite their significantly higher processing requirements.

Most frequency domain algorithms are based on the *phase vocoder*, which utilizes the short-time Fourier transform to convert data into the frequency domain; computations are performed on the signal's *spectrum* to convert it to the desired timebase [7]. The idea behind the phase vocoder is to estimate the spectrum of the time-stretched signal based on the spectrum of the original signal. In doing so, we must compute both the amplitude and phase of the new spectrum. The amplitude is taken directly from the input signal spectrum. Estimating the phase is more tricky as it is computed from the difference among successive input signal frames.

Time-stretched audio signals produced by the classical phase vocoder exhibit significant reverberation artifacts. To solve this problem, we chose to implement an improvement presented in [8] called the phase-locked vocoder. The basic phase vocoder algorithm computes the phase for each output frequency bin in the spectrum independently, without regard to its relationship with neighboring frequencies. Since such a relationship exists for real-world signals, the output audio exhibits the above-mentioned reverberation effect.

Laroche introduces a technique called rigid phase locking to address this issue. Rigid phase locking attempts to preserve the structure of a signal by performing the phase estimation at peaks in the spectrum only. The phases of neighboring frequencies surrounding the peaks are "locked" to these peaks, and thus the structure of the audio signal is preserved. Another way of looking at this is that the amplitude envelope of the signal in the time domain is preserved.

7.2 Improved baton hardware

The current baton system is relatively expensive (around $2000), with even just the baton itself already in the $150 range. Instead, it would be desirable to have one that is cheap and easy to produce. This is particularly important in a museum environment, where the baton could be easily broken, lost, or otherwise misplaced. Tethering the baton is an undesirable option because it detracts from the overall user experience. One possible solution currently being investigated is the use of a simple infrared LED powered by a battery. However, a signal produced by such a simple device may require a more complex tracking device to distinguish the baton input from other external light sources.

7.3 Improved tempo following

The current system only adjusts tempo once every conducted beat, at the downward turning point. We are working on an algorithm that tracks and compares multiple points along the beat trajectory with the expected position at the tempo currently used. The result is a Kalman filter [10] that predicts tempo, amplitude, and size of the gestures on an ongoing basis. How often these predictions are then used to actually change tempo is up to the playback algorithm, taking into account that tempo changes (with the current method) involve track changes that include the possibility of audio artifacts.

8 Summary

Personal Orchestra is the first system that allows users to control an audio and video recording of a real orchestra in real time, using natural conducting gestures. The main technical achievement is the real-time time-stretching architecture, while the overall design is a result of applying a user-centered design pattern language for interactive exhibits. The system also features a very realistic reaction when the user fails to conduct well enough, which has become a major part of the attraction of this exhibit. The system has been successful as a public exhibit that is experienced by over 300 visitors of the HOUSE OF MUSIC VIENNA each day.

The HOUSE OF MUSIC VIENNA received the *2002 Austrian Museum of the Year Award*. The jury particularly praised its

exemplary use of new media to deliver high educational value in a modern and entertaining experience. The award was presented on 14 November 2002.

Acknowledgements. We would like to thank the HOUSE OF MUSIC VIENNA, in particular Stefan Seigner, Robert Hofferer, Christian Bauer, and Dominik Nimptschke, the Vienna Philharmonic Orchestra, and Ingo Grüll for their support in this ambitious project.

References

1. Prosoniq MPEX near-lossless time scaling technology. http://www.prosoniq.net/html/mpex.html
2. HOUSE OF MUSIC VIENNA (2000) http://www.hdm.at
3. Borchers J (1997) Worldbeat: designing a baton-based interface for an interactive music exhibit. In: Proceedings of the CHI '97 ACM conference on human factors in computing systems, Atlanta, GA, 22–27 March 1997. ACM Press, New York, pp 131–138.
4. Borchers J (2001) A pattern approach to interaction design. Wiley, New York. http://hcipatterns.org
5. Buchla D (2003) Lightning II MIDI controller. http://www.buchla.com/
6. Ilmonen T (2000) The virtual orchestra performance. "The art of the desktop" invited session participant. In: Extended abstracts of the CHI 2000 ACM conference on human factors in computing systems, The Hague, The Netherlands, 1–6 April 2000. ACM Press, New York, pp 203–204
7. Laroche J (1998) Time and pitch scale modification of audio signals. In: Kahrs M, Brandenburg K (eds) Applications of DSP to audio and acoustics. Kluwer, Dordrecht, pp 279–309
8. Laroche J (1999) Improved phase vocoder time-scale modification of audio. IEEE Trans Speech Audio Process 7:323–332
9. Mathews MV (1991) The conductor program and mechanical baton. In: Current directions in computer music research. MIT Press, Cambridge
10. Maybeck PS (1979) Stochastic models, estimation, and control, vol 1. Academic, New York. Available at: http://www.cs.unc.edu/∼welch/kalman/maybeck.html
11. Nakra TM (2000) Inside the conductor's jacket: analysis, interpretation and musical synthesis of expressive gesture. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA
12. Rudolf M (1995) The grammar of conducting. Wadsworth, London