Engineering a realistic real-time conducting system for the audio/video rendering of a real orchestra

Jan O. Borchers
Computer Science Dept.
Stanford University
Stanford, CA 94305-9020, USA
borchers@cs.stanford.edu

Wolfgang Samminger
Telecooperation Dept.
University of Linz
4040 Linz, Austria
wolfgang.samminger@liwest.at

Max Mühlhäuser
Telecooperation Group
Darmstadt University
64283 Darmstadt, Germany
max@informatik.tu-darmstadt.de

Abstract

This paper describes the first multimedia system that allows users to conduct a realistic electronic orchestra. Users 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 that it is able to play back the audio and video recording of the orchestra at variable speed while avoiding pitch changes or other artifacts in the playback. These speed changes can be executed interactively in real time while the recording is being played back.

The system has been deployed as an exhibit that has become a major attraction of a large Vienna-based music exhibition center.

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 a particularly good example: People frequently enjoy acting as if they were conducting a classical piece that is being played back from a CD. Naturally, they are really simply conducting alongside the fixed recording, without any way to actually influence the performance. The experience would be much more realistic if the user could indeed *control* the orchestra playing. And (in order to really feel like Karajan) the user should be able to not just hear, but also see the orchestra playing. The goal of this work was to create such a system.

2. Requirements

The above scenario suggested the following basic requirements:

Conducting Device: The system requires a baton-like device to let the user control 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 currently conducted volume (derived from gesture size),

tempo (from gesture speed), and any desired emphasis of certain instrument sections (from the direction in which the user conducts).

Time-Stretching Algorithm: The audio/video recording of the orchestra needs to be played back at varying speeds. To give users enough control, playback speed should be variable between 50% and 200% of the original tempo. However, different playback speeds have to avoid any noticeable artifacts (such as a change in audio pitch), and changes in playback speed need to be possible at any time while the music is playing.

User Interface: Apart from displaying the orchestra, the system also has to offer a means to let the user select a language and a piece to conduct, and to display instructions and other messages.

In addition, there were several requirements due to the fact that the resulting system was to be deployed in the HOUSE OF MUSIC VIENNA [1], a major music museum and exhibition center in Vienna, Austria, that opened its doors to the public in June 2000:

Audio Quality: Since the system was to feature a well-known orchestra (the Vienna Philharmonic), audio quality of the time-stretching component had to be high enough to satisfy the expectations that both potential users and the orchestra had.

Usability: As a system to be deployed in a public museum, the software 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 fulfil other requirements that are particularly important in this class of systems, such as a non-technical, innovative appearance, universal accessability, and robustness [4].

3. Usage Scenario

The following scenario describes the user experience that the system was to convey:

As an exhibit, the system was designed to be set up in a room by its own which was designed to create an environment of the Vienna Philharmonic's concert hall. Users enter this room, find wall-size facsimiles of the available pieces on wall tapestries, traditional note stands, and a red velvet conductor's podium. In front of them, a large rear video projection shows the orchestra, softly rehearsing, waiting for a conductor to become active (see Fig. 1).

When a user picks up the infrared baton and presses the button on it (as indicated on the idling orchestra

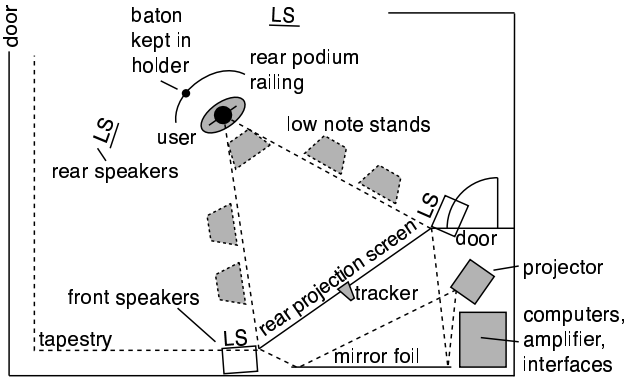

Figure 1. Overview of the *Personal Orchestra* exhibit space.

screen), a first screen appears, and by moving the baton up and down, the user controls a highlight on-screen to select one of the available languages. The selection is activated by pressing the button on the baton. The subsequent screen explains how to conduct, and offers a similar mechanism to select one out of four well-known pieces, or to learn more about the exhibit.

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, when they raise to congratulate the conductor with applause from the audience, or until they have decided that the user keeps conducting too badly (see below).

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

Users are prevented from conducting too slowly or too quickly—not only is time-stretching or time-compressing the orchestra audio with sufficient quality only possible to a certain extent, but the Vienna Philharmonic also would not have liked the idea that exhibit visitors could make them play arbitrarily fast (and look arbitrarily silly in the process). However, an error dialog box informing users about their mistake would have ruined the immersive experience. Instead, we invented a more natural and realistic error message: If the user "teases" the orchestra too much by conducting very quickly, slowly, or stopping completely, the orchestra reacts in the most natural way—they stop playing, and one player gets up to complain about

the conductor's skills. The system includes suitable tolerance rules for the orchestra (currently 8 beats of conducting too quickly or slowly), 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 design patterns* for interactive exhibits [10]. 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.

We have reproduced the overall graph of the pattern language in Fig. 2 to convey an idea of the design patterns that were considered for this project. 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 pattern 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.

HCI design patterns have recently received increasing attention [11]. While it is beyond the scope of this paper to further discuss this approach, it is explained in detail in [10].

4.2. System architecture

Figure 3 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 emphasis 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. The protocol is described further below.

4.3. Input technology

We used Don Buchla's *Lightning II* infrared baton system [12]. It translates input from the infrared-emitting, battery-operated baton, received by a tracker

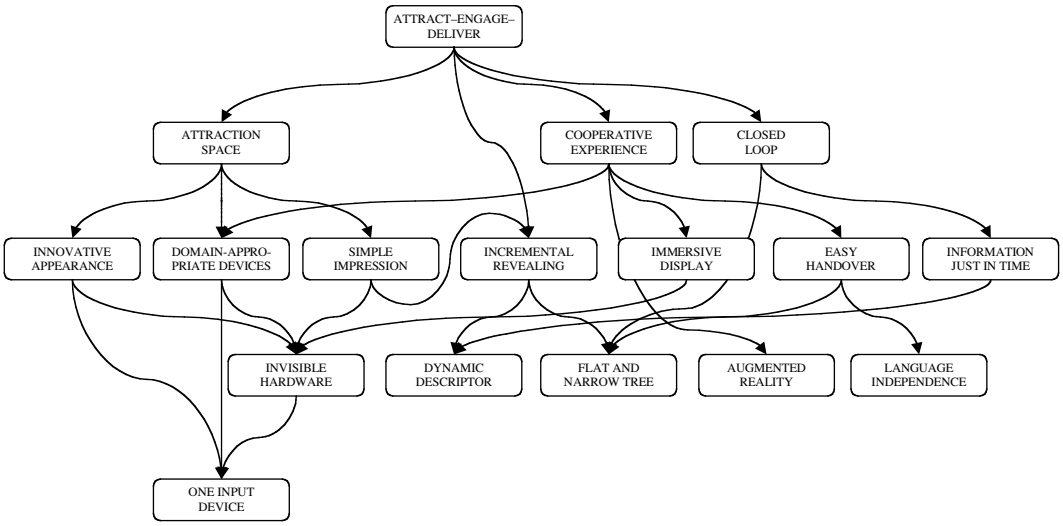

Figure 2. The HCI design pattern language for interactive exhibits [10].

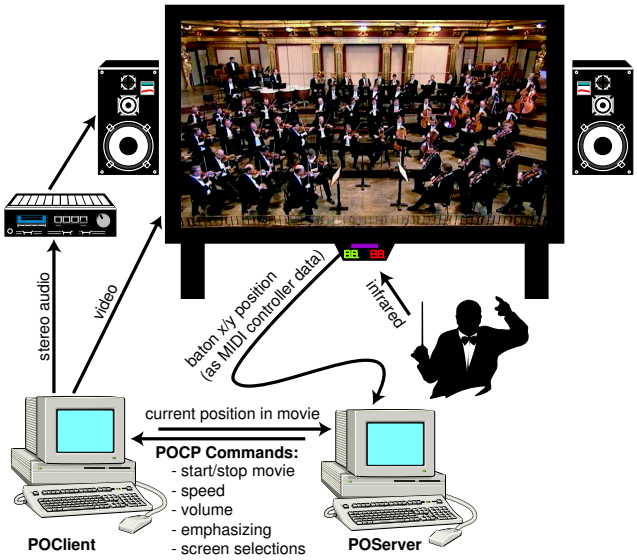

Figure 3. *Personal Orchestra* **system architecture.**

mounted below the screen, into clean MIDI controller signals representing x/y baton coordinates with a resolution of 7 bit each. A third, binary controller 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 marked manually as the "beats" in the music, using a simple utility we developed to take time-stamped key press inputs from a user tapping alongside a piece being played back.

The current playback speed is then adjusted so that the orchestra always follows the conductor. There are two major problems with this, however:

First, a conductor may be conducting at the same speed as the orchestra plays, still 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 speeds up, for example, a part of the current beat has not been played yet when the next, first conductor beat gesture at the higher tempo arrives: the orchestra has to "catch up" with the conductor (Figure 4). To re-synchronize with the conductor, playback speed has to be increased above the target (measured) new conducting rate for a while, until movie and conductor are at the same time in their piece, then it has to level back off to the actual new conducting speed, according to the following formulae:

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

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

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 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 over-estimate 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 against such errors. We left this parameter as a variable that can be changed before running the system, to simplify adjustments in everyday use.

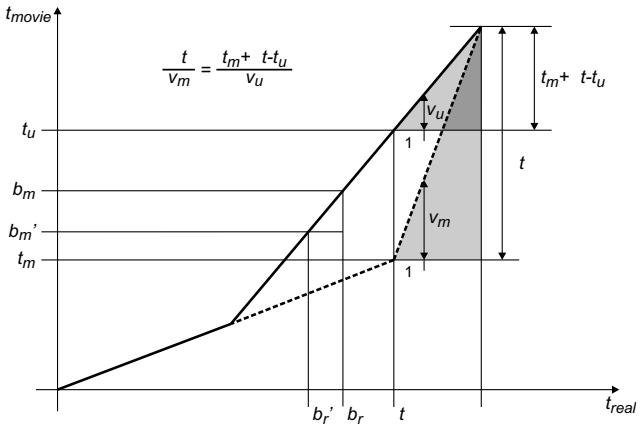

Figure 4. When a conductor speeds up, the orchestra has to play faster than the new tempo to get back in phase.

A similar low-pass filter was implemented for volume control. In order 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 $[0; 1]$:

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

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

Figure 5. The *Personal Orchestra* exhibit in the HOUSE OF MUSIC VIENNA**.**

4.5. High-quality interactive audio/video time-stretching

A broadcast-quality 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. The challenge now was to adjust the speed of, or *time-stretch,* the orchestra movie being played back.

Time-stretching video is simple; most multimedia libraries easily handle changes in playback speed by repeating 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 uncritical with our scenery.

The audio track, on the other hand, creates a problem since of course, just changing the speed of an audio recording being played back will also change its pitch (an effect known from choosing a different speed on an analog record player). It is relatively easy to avoid this by Fourier-transforming the audio signal: in frequency space, it is possible to change the duration of a signal without changing its frequency. After inverse Fourier transformation, the resulting audio signal can be played in a time-stretched version at the same pitch. Another way to look at the same process is *granular synthesis*, which essentially cuts the audio signal into small packets of ca. 50ms and then repeats or leaves out packets to adjust tempo.

Unfortunately, these simple methods create noticeable artifacts in the audio signal. Typically, slowed-down Fourier-transformed versions will exhibit a strong reverberation component since all parts of the audio signal will have been prolonged equally. Granular synthesis needs to mix several signals into each other to avoid artifacts at packet borders, and even then fast attack sounds, if they happened to fall into a packet that is repeated, would sound twice.

Still, various algorithms exist that time-stretch audio in real-time. However, while some of these can, for example, speed up ("time-compress") recorded speech [13] with high intelligibility, these algorithms produce insufficient sound quality when applied to polyphonic, musical audio signals.

Essentially, audio data needs to be preexamined, even depending on music type, and time-stretched offline to take care of these special cases, which takes a multiple of the original playing time to create good audio results. Naturally, it is only a matter of time before affordable hardware can do these computations in real time. However, at the time Personal Orchestra was implemented (1999–2000), the ratio of processing time to signal length was still far away from real-time performance (32:1 on a Pentium III/450 MHz processor).

For that reason, we intended to pre-time-stretch all our audio channels at various speeds, and then, for each channel in parallel, crossfade between its pre-stretched versions to change playback speed. That way, time-stretching would take place only once during development for each channel and speed required, taking as much time as necessary to produce the best possible audio quality. During playback, all to be done would be to determine the new tempo required, and smoothly crossfade all four audio channels from their current tempo track over to their newly selected one within a few milliseconds. (This crossfade is necessary to avoid the audible clicks that the audio waveform discontinuities during an immediate track switch would create.)

However, this would have 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 one 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) [14].

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

Figure 6 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, tracks 2–14 the full-orchestra stereo audio track, ordered from lowest to highest pitch (in increments of $\sqrt{(2)}$, 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, and after 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. A server machine dealt with ges-

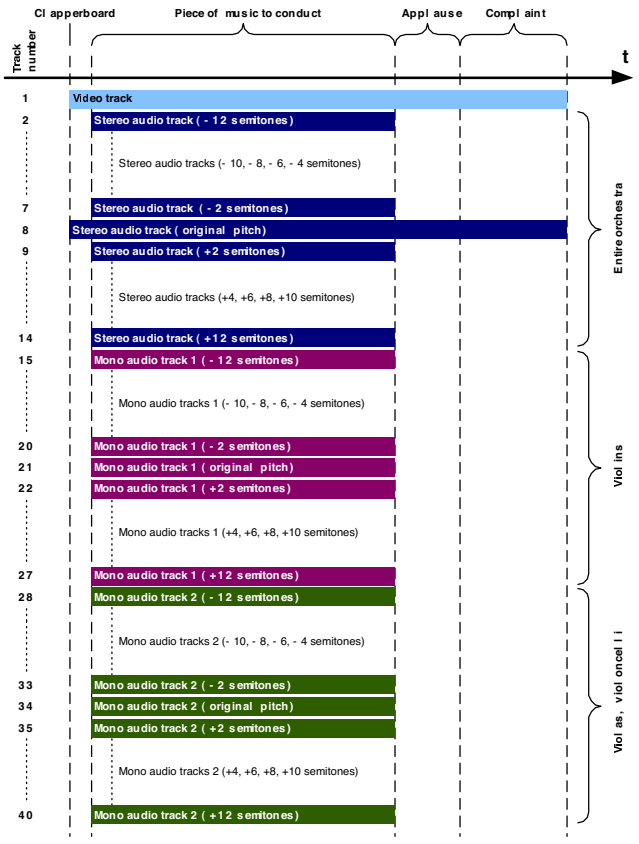

Figure 6. Layout of audio and video data streams in a single movie file.

ture recognition and tempo/volume/emphasis computation, while a client machine used this information to render the audio and video of the orchestra at high quality.

Our Personal Orchestra Control Protocol (POCP) defined the messages that the gesture recognition component (POServer) could exchange with the audio/video rendering component (POClient). It was designed to be lightweight, and because of the fairly low bandwidth this control channel required, it was feasible to make the protocol human-readable in order to simplify 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 1/600s) 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 5 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, Figure 7 shows a sample communication between POServer and POClient.

After launching the application, POClient loads the idle loop automatically. The first commands from POServer then tell POClient to jump to the beginning of that loop, and play it at 5% of full volume. POClient then reports its current movie position continuously, until POServer determines that the idle movie has reached its end, upon which POServer tells POClient to start again from the beginning.

When the user presses the baton button, POServer tells POClient to stop the idle loop and switch to the language selection screen, starting with the default language (German) selected. In this example, the user briefly highlights English as language, but then goes back to German (LNG DE) and presses the baton button. POServer next tells POClient to show the piece selection screen highlighting the second piece (because the user held the baton at the initial height corresponding to that selection when pressing the button an instant ago).

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).

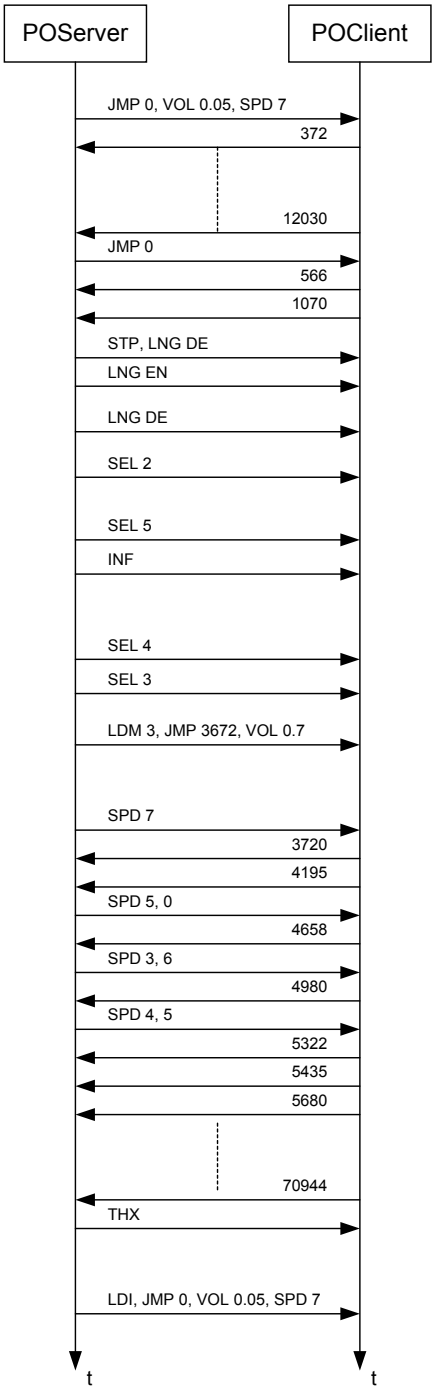

Figure 7. Sample communication through the Personal Orchestra Control Protocol (POCP).

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), 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 to conduct, 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, 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 either in the orchestra complaining when the conducting is too bad for several beats in a row, or with the end of the piece, with the orchestra raising and a big round of applause from the invisible audience behind the conductor. The state diagram of the system is shown in Figure 8.

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 716x288 pixels) for each piece are contained in the combined QuickTime movie file and are streamed directly off the hard disk. The maximum data rate of the material is around 3.6 Megabytes per second, leading to each musical

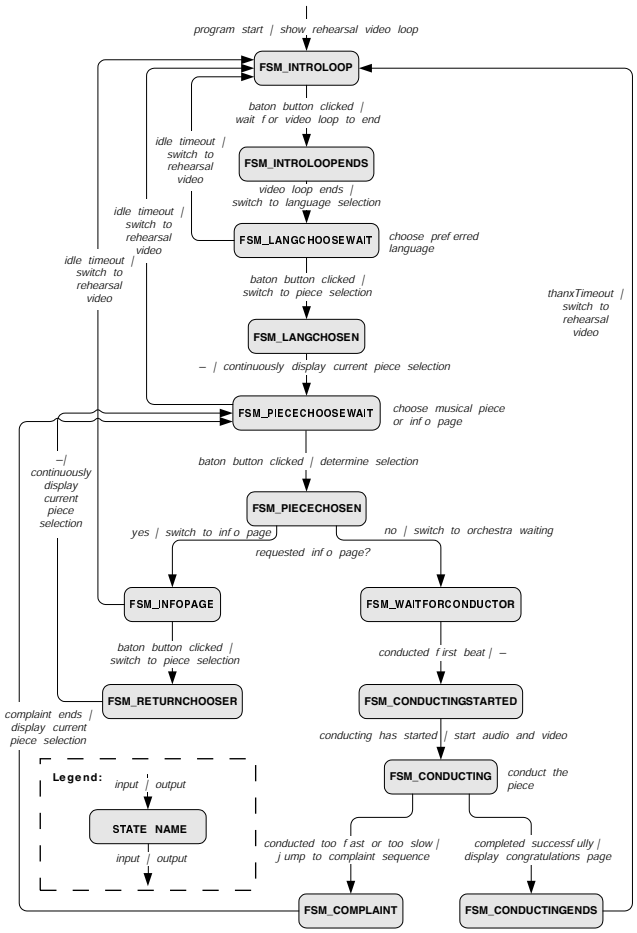

Figure 8. The Finite State Machine of *POServer*.

piece (between 3 and 5 minutes) being represented by a QuickTime movie file with a size between 600 Megabytes and 1 Gigabyte.

Video is projected via a rear-projector attached to POClient; audio is fed from the same machine into a high-end speaker setup with 2 front and 2 rear speakers and a sub-woofer 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. In an initial round, qualitative observations showed that very few users managed to conduct a complete piece successfully. On the other hand, our "error message" of the orchestra complaining turned out to be a major attraction of the system for users, who would frequently try to intentionally provoke a complaint from the orchestra. However, it often happened that the orchestra would stop and complain while the user was just getting acquainted with the system, or shortly before finishing a piece. Both situations were very frustrating for users. We therefore increased the tolerance of the orchestra by fine-tuning our parameter sets, and introduced safety zones at the first and last few seconds of each piece to avoid those frustrating error situations. We also learned that users 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 (in both languages) into the idle loop encouraging the user to pick up the baton and push its button.

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. 97% of all users managed to do basic conducting gestures recognized by the system. 93% 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. 60% of all users managed to finish conducting a piece without errors, 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). 93% voted the exhibit to be a top three exhibit in the HOUSE OF MUSIC VIENNA.

6. Portability

More recently, experiments by other research groups have shown Apple's recently released Mac OS X operating system to deliver surprisingly good audio performance [17]. We have ported our system to this new platform, and found that this helped eliminate most of the remaining audio artifacts when switching tracks. Also, initial tests show that its preemptive multitasking and multiprocessor support balance processing power well enough to let us run the entire system on one dual-processor instead of two single-processor machines.

This simple port was largely possible because of two design choices: first, because we used Java and QuickTime as high-level APIs, which are available in nearly

unchanged form under this more modern operating system, there was little need to change the underlying code apart from the low-level MIDI input routines. Second, because POCP is a standard TCP-based protocol, we were able to simply run POClient and POServer on the same machine with sufficiently low overhead.

7. RELATED WORK

There is a large body of research in systems that follow human conducting. We will only present those efforts most relevant for comparison here.

Max Mathews' *Radio Baton* [2] was among the first systems to provide a conducting experience. It uses the movement of one or more batons emitting radio frequency signals above a metal plate to determine conducting gestures. A MIDI file is played back in sync with these movements. Conducting is restricted to the space above the metal plate.

In Realtime Music Solution's *Virtual Orchestra* [3], a commercial system by F. Bianchi, D. Smith, J. Lazarus and S. Gabriel two technicians watch the movements of a conductor and adjust playback parameters of a computer cluster accordingly in real time. The system has been used successfully in many commercial productions, but produces synthesized sound only and does not include a video of the orchestra.

The *WorldBeat* interactive music exhibit [4] contains a *Virtual Baton* feature to let users conduct a classical piece using an infrared baton. It reacts very directly and realistically, using gesture frequency, phase, and size to adjust tempo and dynamics. It detects the upbeat at the start of a piece, and detects syncopic pauses in mid-play. Conducting again controls playback of a MIDI score. The conducting feature is based on earlier work by Guy Garnett et alii. [5].

Satoshi Usa's *MultiModal Conducting Simulator* [6] uses Hidden Markov Models and fuzzy logic to track gestures with a high recognition rate of 98.95–99.74%. It plays back a MIDI score, with matching tempo, dynamics, staccato/legato style, and an adjustable coupling of the orchestra to the conducting.

Marrin's *Digital Baton* measures additional parameters beside baton position, such as pressure on parts of its handle, to allow for richer expression [7]. Her *Conductor's Jacket* [8] uses sixteen additional sensors to track muscle tension and respiration, translating gestures based on these inputs to musical expressions. It uses a MIDI-based synthesizer to create the resulting musical performance.

Tommi Ilmonen's *Virtual Orchestra*, demonstrated at CHI 2000, is one of the few systems that also feature graphical output; however, it renders the orchestra synthetically as 3-D characters. Audio output is again MIDI-based [9].

Thus, all these systems share one or more of the following characteristics, rendering them unsatisfying for us:

- 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: using synthesized sound generation such as MIDI playback instead of processing the actual audio recording of a piece makes it virtually impossible to create a system with 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.

8. FUTURE WORK

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

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[15] 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 possility of audio artifacts (but see below).

Usability improvements. Some minor usability problems that have surfaced over time need to be addressed. For example, we intend to supply English subtitles to the (Austrian) complaints from the orchestra, and improve the selection screen at the beginning with an animated video showing how to conduct, since this will grasp the user's attention more than the current graphical and textual explanations (which, as observations revealed, visitors hardly ever read.)

Continuous tempo adjustment. The latest advances in algorithms and computing power have now made continuous real-time interactive audio time-stretching an increasingly realistic alternative (see, for example, [16]). We are currently working with those researchers to explore the potential of such an approach. The benefits would be that, at any time, the orchestra could match the required speed precisely instead of having to choose the closest existing track, and that artifacts from switching tracks could be eliminated altogether.

New input device. We have been working on a more robust and readily available replacement for the *Lightning II* infrared baton system to make it easier for us and others to continue research and experiments in this domain.

9. SUMMARY

Personal Orchestra is the first system to let users control an actual 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.

10. 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] HOUSE OF MUSIC VIENNA web site, http://www.hdm.at/, 2000.

[2] Mathews, Max V.: The Conductor Program and Mechanical Baton, in Max V. Mathews and J. R. Pierce, eds.: *Current Directions in Computer Music Research,* MIT Press, Cambridge, 1991.

[3] Pogue, David: The dangers of the Digital Orchestra. *New York Times Direct,* Apr 5, 2001.

[4] Borchers, Jan: WorldBeat: Designing a baton-based interface for an interactive music exhibit. *Proc. CHI'97*, pp. 131–138, ACM, Atlanta, 1997.

[5] Lee, Michael, Garnett, Guy and Wessel, D.: An Adaptive Conductor Follower. *Proc. ICMC 1992*, ICMA, San Jose, 1992.

[6] Usa, S. and Mochida, Y.: A conducting recognition system on the model of musicians' process. *Journal of the Acoustical Society of Japan*, **19**(4), 1998.

[7] Marrin, Teresa: Possibilities for the Digital Baton as a general-purpose gestural interface. *Proc. CHI'97*, ACM, 1997, pp. 311–312.

[8] Marrin Nakra, Teresa: Inside the Conductor's Jacket: analysis, interpretation and musical synthesis of expressive gesture. *PhD thesis, Massachusetts Institute of Technology,* 2000.

[9] Ilmonen, Tommi: The Virtual Orchestra performance. *Proc. CHI 2000*, ACM, 2000.

[10] Borchers, Jan: *A pattern approach to interaction design.* 264 pages, John Wiley & Sons, New York, 2001 (http://www.hcipatterns.org/).

[11] Borchers, Jan and Thomas, John C. : Patterns—what's in it for HCI? *CHI 2001 Ext. Abstracts*, ACM, 2001.

[12] Don Buchla: Lightning II MIDI Controller. http://www.buchla.com/

[13] Griffin, D. W. and Lim, J. S.: Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing,* ASSP-**32**(2):236-243, April 1984.

[14] The PROSONIQ MPEX Near-Lossless Time Scaling Technology, http://www.prosoniq.net/html/mpex.html.

[15] Maybeck, Peter S.: *Stochastic Models, Estimation, and Control, Volume 1.* Academic Press, New York, 1979, http://www.cs.unc.edu/~welch/kalman/maybeck.html.

[16] Bonada, Jordi: Automatic technique in frequency domain for near-lossless time-scale modification of audio. *Proc. ICMC 2000*, ICMA, Berlin, 2000.

[17] MacMillan, K., Droettboom, M., and Fujinaga, I.. Audio Latency Measurements of Desktop Operating Systems. *Proc. ICMC 2001*, ICMA, Havanna, 2001.