Virtual Air Guitar

Matti Karjalainen¹, Teemu Mäki-Patola², Aki Kanerva³, Antti Huovilainen¹, and Pekka Jänis¹

¹Helsinki Univ. of Tech., Laboratory of Acoustics and Audio Signal Processing, Espoo, FIN-02015 HUT, Finland
²Helsinki Univ. of Tech., Telecommunication Software and Multimedia Laboratory, Espoo, FIN-02015 HUT, Finland

Correspondence should be addressed to Matti Karjalainen (matti.karjalainen@hut.fi)

ABSTRACT

A combination of hand-held controllers and a guitar synthesizer is called here the "Virtual Air Guitar". The name refers to playing an "air guitar", i.e., just acting the playing with music playback, and the term virtual refers to making a playable synthetic instrument. Sensing of the left hand position is used for pitch control, the right hand movements for plucking, and the finger positions of both hands for other features of sound production. The synthetic guitar algorithm supports electric as well as acoustic sounds, augmented with sound effects and intelligent mapping from playing gestures to synthesis parameters. The realization of the virtual instrument is described and sound demos are made available.

1. INTRODUCTION

Synthesis of musical instruments and position-sensing controllers designed for virtual reality can make interesting playable virtual instruments. In this study we developed an instrument that we call the "Virtual Air Guitar". The idea was to combine "air guitar playing", i.e., just acting the playing of an imaginary instrument with music playback¹, and a guitar synthesizer including sound effects, thus making a virtual instrument, playable with sensors that follow hand movements.

There were two main motivations to the present study:

(a) to study the expressiveness of synthetic guitar playing with hand-following sensors and (b) to design an easy-to-play attraction for a science center exhibition.

The entire virtual air guitar is composed of several subsystems: (a) a guitar synthesizer with sound effects and audio reproduction, (b) a user interface consisting of hand-held sensors (as well as possible foot controllers), and (c) software to map user interface signals to expressive playing of the synthetic guitar. While some of the hardware and software components are available off-the-shelf, we wanted to explore many of the subsystems more freely with new innovations.

The guitar synthesizer is realized as an extended Karplus-Strong type of synthesis algorithm. For electric guitar sound it is fairly easy to calibrate the algorithm

¹Air guitar playing means typically mimicking of rock guitarists, their often acrobatic choreography (http://www факт-индекс.com/a/ai/air_guitar.html). An Annual Air Guitar World Championship contest is held in Oulu, Finland (http://www.omvf.net/2004/ilmakitara.php).
Fig. 1: Basic structure of the single-delay loop filter and excitation mechanism for implementing the Extended Karplus-Strong string model.

to yield a desired sustain and spectral properties. For good attack a wavetable can be sampled through inverse filtering of a real played sound by the calibrated string model. For good acoustic guitar sound the calibration of the model is more involved, because a body model is also required.

Audio effects are needed for realistic electric guitar sounds. Reverberation as room simulation is useful also for the acoustic guitar. The electric version of the virtual air guitar includes a distortion unit, a delay unit, and a reverb unit as software algorithms integrated with the guitar synthesizer.

For the user interface we have experimented two approaches: (a) in a cave-like virtual room using data gloves and position tracking, multi-wall stereo-vision video projection, and multi-channel sound reproduction, and (b) with small-sized hand-held control devices. Hand-tracking by video image analysis will also to be checked. The main control parameters required in all cases are the left hand position for pitch control and the right hand movement sensing for plucking of strings. In case (b) the left-hand pitch control is realized by measuring the propagation delay of high audio frequency pulses from a small loudspeaker to an electret microphone in the left-hand controller, thus estimating the distance of the left hand to the body or the right hand of the player. The right hand plucking movement is measured by an acceleration sensor chip. In addition to pitch and plucking controls it is desirable to use fingers in one or both hands to control such features as selection of strings and chords as well as string damping by the right hand operations.

For easy playing of the virtual instrument a number of algorithms are under development to map the controller signals to synthesis parameters, sound effects, and possible accompaniment of a background playback file.

Many artificial intelligence algorithms can be utilized to map simple hand movements to performances mimicking qualified musicians. This is particularly useful in the science center demo version of the system.

2. THE GUITAR MODEL

In this section we describe the synthetic guitar model, particularly a Stratocaster type of an electric guitar, as well as sound effects processing.

2.1. Electric guitar synthesizer

Physics-based models for real-time synthesis of the guitar have been studied actively, see for example [1]–[11]. Many popular models are based on the Extended Karplus-Strong (EKS) concept, see particularly [2] and [6]. It is a computationally efficient digital waveguide algorithm for modeling the guitar string as a single-delay loop filter structure with parametric control of fundamental frequency and losses in the filter loop.

Figure 1 depicts the basics of this model structure. A wavetable (often a set of selectable wavetables) stores an excitation waveform that may be a synthetic signal or often extracted from a real recorded guitar pluck by inverse filtering with the EKS filter structure. A timbre control filter, shaping the high-frequency damping vs. boosting, follows the wavetable. This excitation is further passed through a pluck position control, making a comb filter effect that results from summing up the two traveling waves in the string, depending on in which point the string is excited. The signal is then fed through the single-delay feedback loop which is the most fundamental part of the string model. Finally the signal is integrated so that the output is proportional to the force at the bridge.

In the electric guitar, which is the case discussed in this paper, the signal is captured by a magnetic pickup.
From the string loop

\[ y(n) \]

\[ P_s(z) \]

\[ I(z) \]

\[ a(n) \]

Fig. 2: Filter model for a magnetic pickup in an electric guitar.

Vertical polarization

Horizontal polarization

Sympathetic couplings from other strings

Vertical polarization

Out

Sympathetic couplings to other strings

\[ S_h(z) \]

\[ m_o \]

\[ 1 - m_o \]

Fig. 3: Dual-polarization string model.

For example in a Stratocaster type of solid-body model there is a selector for three pickups: bridge, middle, and neck pickup. The magnetic pickup [12] is sensitive to the string velocity in vertical direction, and it introduces another comb filtering effect in a way similar to the plucking point filtering. Figure 2 depicts a filter model for the pickup [6]. Delay is the time of wave propagation from pickup to bridge and back. Block LP is an optional filter to adjust the comb filtering details. Block \[ P_s(z) \] implements the spectral shaping characteristics due to electrical RLC-type lowpass and the finite effective width of the pickup sensing of the string vibration. Filter \[ I(z) \] is an integrator to yield the string velocity.

Figure 3 shows a full synthesis model for a dual-polarization guitar string [6]. The model is developed primarily for the acoustic guitar, but is applicable to the electric one when properly interpreted. The wavetables store excitation waveforms that capture the plucking and the body effects. While in an acoustic guitar the body response can be commuted to the plucking wavetable [13, 14], in the solid-body electric guitar the body effect is of minor importance only. Pluck shaping and plucking point filters are as discussed above. Sympathetic couplings between strings play a minimal role in the solid-body electric guitar, so they can be omitted.

The magnetic pickups in the electric guitar are sensitive to the vertical movements of the string only. Yet a dual-polarization string model is useful for simulating the slight beating effects found in the partials of string vibration. When the string is plucked, there is a combination of vibration components of both polarizations. The polarization state will change, however, in the course of time so that the pickup output exhibits more or less beating in the envelope of each partial. This phenomenon can be simulated by summing dual-polarization model components, although this is not a physically correct way of realizing the beating. Since the beating is typically not a very strong effect in the electric guitar, it is not a critical yet a desirable feature in sound synthesis.

2.2. Calibration of the guitar model

In this paper we discuss a case of modeling and synthesizing a solid-body guitar that is a hand-made copy of the Fender Stratocaster. A fairly accurate synthesis model was achieved using the procedure described below. Each string is calibrated separately in the following way:

- The string is plucked about 1 cm from the bridge in vertical direction by a hard sharp pick. This approximates an ideal pluck so that the comb filtering due to plucking point effect is moved to high enough frequencies. The response is recorded from
the bridge pickup with the tone and volume control potentiometers turned clockwise.

- The decay time constants of 10–20 lowest partials are analyzed. This can be done using procedures proposed for example in [7, 15]. Here we band-pass filtered each partial and fitted a line to dB-scaled decay curve for the fret 1 and 12 fingerings. The period and depth of beating in the envelope of each partial was also approximated, if prominent.

- A first-order lowpass loop filter (Fig. 1) with two control parameters, DC gain and a high-frequency control parameter, can be fitted to the string model in the two cases of fingerings, for example by the algorithm proposed in [7]. Due to relatively regular behavior in the electric guitar, the loop filter parameters were adjusted manually, instead of an automatic procedure, to give proper decay times. For other fingerings the loop filter parameters are interpolated linearly at runtime according to fret position. In practice we have divided the loop filter into two cascaded filters, one for nominal damping values inherent for the string as a function of fretting position and another one for controlling extra damping due to right or left hand operations.

- Approximately correct values are set for the polarization submodel string length difference for proper beating rate and mixing ratio for beating depth. Because these effects need to be only roughly approximated, the parameters are adjusted manually to yield a perceptually good result. These parameters are also interpolated according to fretting position.

- Fifth-order Lagrange interpolation is used for loop delay control [16] to obtain click-free variation of string length and good approximation of unity magnitude response up to about 10 kHz (sampling rate of 44.1 kHz) in order not to affect the loop gain.

- The single-coil Stratocaster pickup model used here is a lowpass plus comb filter type of a digital filter. It is designed to match the pickup response spectrum recorded above. Figure 4 shows a measured spectrum and a single-coil pickup response spectrum in comparison when the pickup used was modeled as a cascade of two 4th order Butterworth lowpass filters with cutoff frequency of about 10 kHz plus the comb filtering. Notice that a specific pickup responses depend on many factors such as pickup type, electric control circuitry and settings, electric loading, and magnet pole distance from the string.

- The calibration procedure described above yields proper sound for ‘ideal’ plucking if an impulse is used as the excitation wavetable to the model. Better sounding attacks can be obtained if real instrument recordings are inverse filtered by the string and pickup models, and the residual signal is properly truncated and used as a wavetable. Different plucking and excitation sounds can be stored in different wavetables and the pluck shaping filter can be used for fine-tuning of the timbre.

The Stratocaster guitar synthesizer consists of six strings, each one calibrated separately, with controls for pitch (string length), plucking point, pickup position, plucking wavetable selection and triggering, and plucking spectral shaping for smoother or sharper attack. The model is implemented as a patch in BlockCompiler [17, 18], and it can be exported to different other software platforms such as Mustajuuri [19] and pd [20].
2.3. Guitar effects

The solid-body electric guitar without any sound effects is perceived in general quite dull in timbre. Sullivan [3] demonstrated that for a simple synthetic guitar model the addition of distortion and feedback can make relatively convincing classic distortion guitar effects.

In rock guitar playing, the sound of the guitar is processed with a variety of effects to obtain the final sound. In particular, the recorded sound of the guitar is typically fed through a tube amplifier. When driven at high volumes, this amplifier as well as the loudspeaker introduce non-linear distortion and overtones to the sound, which is pleasing to the human ear. The Virtual Air Guitar (VAG) is designed primarily for rock music, so the system must simulate a typical rock guitar set-up as well as the sound of the guitar itself.

A typical setup consists of a compressor, a preamplifier, a power amplifier, speaker cabinets, an equalizer, a reverb unit, and possibly a delay unit. While we can use any external effect devices and amplifiers with our virtual air guitar, we wanted to combine all elements to software that runs on a single computer. In the Mustajuri audio software we have used the CAPS plug-in suite [21], a collection of open source audio plug-ins using the LADSPA, a common application programming interface for audio processors on the Linux platform.

In this study we paid special attention to the distortion effect produced in tube preamplifiers and developed a simulation model that runs on the BlockCompiler software. We describe this tube distortion simulation in detail below. Other effects used are not discussed here.

2.3.1. Tube amplifier stage

The circuit diagram for a typical tube preamplifier stage is shown in Fig. 5. The voltage \( V_{\text{pk}} \) between grid and cathode as well as the voltage \( V_{\text{pk}} \) between plate and cathode determine the plate current. The plate resistor \( R_p \) converts this current to voltage. The capacitor \( C_o \) acts as a highpass filter to block the DC voltage from the next stage input. The resistor \( R_i \) prevents parasitic high-frequency oscillations and limits the input current to the grid of the tube when \( V_{\text{pk}} \) approaches or exceeds zero volts. Resistor \( R_k \) biases the cathode positive compared to grid (which is normally at ground potential). The capacitor \( C_k \) acts as a lowpass filter to keep signal current from affecting the cathode voltage. For more information about triode amplification stages, see [22].

Because the \( V_{\text{pk}}, V_{\text{pk}} \) to \( I_p \) transfer function of an electron tube is nonlinear and asymmetric, the cathode voltage changes somewhat depending on the average signal level. It produces dynamically varying distortion on note attacks as the bias shifts. The authors feel that this is one of the reasons many guitarists describe tube amps as sounding more “dynamic” or “lively”. This effect was therefore deemed desirable to be duplicated.

Another source of time-variance in the circuit is the input capacitor \( C_i \) (same as previous stage’s output capacitor \( C_o \)). Because the grid current rises rapidly as the grid to cathode voltage approaches zero, the normally constant voltage over \( C_i \) is changed. This nonlinearity is only to one direction and causes a shift in the input DC voltage, biasing it negatively. If the biasing is severe enough, it will cause the stage to go into cutoff, making the signal cut in and out — an effect known as blocking distortion [23]. Blocking distortion can be avoided by increasing the input resistor \( R_i \) to limit the amount of grid current.

2.3.2. Model of the tube amplifier stage

Fully modeling the amplifier stage would require solving a system of non-linear differential equations which is problematic and CPU intensive. However, if the plate load is assumed to be constant and resistive, then plate current and voltage are coupled and a curve \( V_p(V_{\text{pk}}) \) can be calculated. This curve depends only on grid to cathode voltage \( V_{\text{pk}} \), power supply voltage \( V_+ \) and plate resistor \( R_p \), and of course tube type. If \( V_+ \) and \( R_p \) are fixed constant, the curve can directly be measured by shorting the cathode in Fig. 5 to ground and varying \( V_{\text{pk}} \). A curve for tube 12AX7 with \( V_+ = 250 \) V and \( R_p = 100 \) k\(\Omega \) is shown in Fig. 6.

\[ V_p = \frac{R_p}{R_p + r} \cdot V_{\text{pk}} \]

where \( r \) is the cathode resistor as shown in Fig. 5.
A small error still remains because $V_k$ changes with input level. This can be safely ignored as the variation in $V_k$ (hundreds of millivolts) is very small compared to $V_p$ variation (approximately 200 V peak-to-peak). Note that this does not mean that $V_k$ variation can be completely ignored as the input level is of similar order of magnitude.

The $V_{pk}$ to $I_{pk}$ curve can be measured in a way similar to plate voltage curve. It was found to be approximately exponential in shape. Fitting a curve to the measured data gave

$$I_{pk} = e^{7.75 V_{pk} - 18.3}$$

(1)

In simulation, the $V_i$ to $V_{gk}$ relationship is more useful than the $V_{pk}$ to $V_{gk}$ relationship. If the input impedance is assumed to be constant and resistive, this means solving the implicit equation

$$\frac{V_i - V_k}{R_i} = I_{pk}(V_{gk})$$

(2)

and then computing

$$V_p = V_{pk}(V_{gk})$$

(3)

These can be combined into a single function $F_{\text{tube}}(V_{gk}, R_i)$ for use in a digital model. This function can be pre-calculated to avoid solving of an implicit equation at runtime.

### 2.3.3. Digital model of the tube stage

With a further assumption that the input DC blocking capacitor $C_1$ can be replaced by a normal first order high-pass filter (a reasonable assumption as blocking distortion is usually avoided as much as possible in amplifier design) Fig. 7 now shows a digital model of the tube amplifier stage.

![Digital model of the tube stage](image)

**Fig. 7: Digital model of the tube amplifier stage.**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_p$</td>
<td>250 V</td>
</tr>
<tr>
<td>$R_i$</td>
<td>100 kΩ</td>
</tr>
<tr>
<td>$HPF_o$</td>
<td>31 Hz</td>
</tr>
</tbody>
</table>

Table 1: Parameters common for all stages.

The $LPF_{in}$ block models the lowpass filtering caused by $R_i$ and tube miller capacitance [24]. The cutoff frequency is $1/(2\pi R_i C_{pg}(1 + G_{gkb})$ where $C_{pg}$ is the plate to grid parasitic capacitance (typically around 1.7 pF [25]) and $G_{gkb}$ is the gain of the tube stage (around 60).

$HPF_o$ is the interstage DC blocking filter. Its cutoff frequency depends on the output capacitor value and the next stage input resistance.

The gain block $R_k / R_p$ provides the biasing for the tube, while $LPF_k$ models the lowpass filtering caused by $C_k$. A unit delay is inserted between the output of $LPF_k$ and tube input to make the model realizable. The extraneous phase shift caused by the unit delay is not a problem as $LPF_k$ cutoff frequency is very low (tens to hundreds of Hz) and gain at high frequencies is low.

### 2.3.4. Model for a complete amplifier

Model for a complete guitar amplifier consists of three gain stages followed by an equalizer and cabinet simulator. The power amplifier is not simulated. Table 1 shows parameters that are common for all stages and Table 2 shows individual stage parameters. These parameters were derived from an AX-84 High Octane guitar amplifier preamp circuit [26].

Guitar amplifier equalizers are usually purely passive and the controls have complex interactions. However, the resulting response shape is usually more or less a ‘V’-shape curve. A parametric mid-cut EQ can be used to get acceptable results.

Guitar amplifiers are practically always used with dedicated guitar speakers. Their response is often far from
Fig. 8: Frequency response of Celestion Vintage 30 speaker in 4x12" cabinet.

flat and the response shape strongly colors the signal. Certain amplifiers and music styles are often associated with specific speakers and cabinets (for example the 4x12" "Marshall stack" is almost de-facto standard in some rock styles). Furthermore, the fast decrease in response after 5 kHz attenuates the harsh sounding high frequencies that will result from distortion. Figure 8 shows the frequency response of a typical guitar speaker.

A simple solution for speaker simulation is to convolve the signal with a recorded impulse response. While producing very good results, this technique is computationally intensive to be feasible. Therefore the speaker is often simulated with an IIR filter bank that mimics the peaks and notches. To replicate the response accurately, a large number of filters is required which is again computationally intensive. Furthermore, matching a set of IIR filters to a certain response is not trivial. We have therefore opted to use a minimum-phase reconstruction of a recorded impulse response. This has the advantage that any speaker and cabinet combination can be simulated easily. The method loses all time and phase information, but because speakers are often close-miked, it is unlikely to be a problem. Informal listening has shown that using a 200 tap minimum-phase FIR filter (at 44.1 kHz sample rate) is indistinguishable from the original close-miked impulse response for typical guitar speakers.

### Table 2: Parameters for amplifier stages.

<table>
<thead>
<tr>
<th>Stage</th>
<th>$R_s$/kΩ</th>
<th>$L_P F_{in}$/Hz</th>
<th>$L_P F_b$/Hz</th>
<th>$R_b$/Ω</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage1</td>
<td>68</td>
<td>22570</td>
<td>86</td>
<td>2700</td>
</tr>
<tr>
<td>Stage2</td>
<td>250</td>
<td>6531</td>
<td>132</td>
<td>1500</td>
</tr>
<tr>
<td>Stage3</td>
<td>250</td>
<td>6531</td>
<td>194</td>
<td>820</td>
</tr>
</tbody>
</table>

Fig. 9: Three levels of mapping of user input to sound model parameters in the Virtual Air Guitar user interface.

3. **USER INTERFACES**

Playing air guitar is like playing rock guitar without the actual physical instrument or even without any musical skills. Therefore, the Virtual Air Guitar is not a professional musical instrument, but more of an entertainment device. It follows, then, that the user of an air guitar does not want an intricate and expressive instrument that requires years to learn to play, but something that allows them to rock on with their favorite music.

The air guitar is by definition a very non-physical instrument, and so its user interface is controlled by monitoring the user’s hand movements. The physical parameters of the extended Karplus-Strong model are not very intuitive even for a real guitarist, and much less so for an air guitarist. Thus, the interface must map the parameters to intuitive gestures used in air guitar playing.

For the most complex implementation, Fig. 9 shows how input data from the user is mapped through various modules to the sound model. First, the user’s hand movements (and finger movements, depending on implementation) are tracked by the Input module. It sends out position data to the Gesture Recognition module, which identifies various playing gestures from the data it receives.

At the simplest, the gestures control the Sound Module directly. However, they can also be made to trigger musical phrases from the Phrase Database, which contains information on what actions are performed on the guitar, in which order and at what time to pass them on to the Sound Model Controller. The Controller receives actions...
Karjalainen et al Virtual Air Guitar

and converts them into control data that the Sound Model understands.

Some actions, such as a pull-off, depend on the current state of the guitar. When the controller is told to do a pull-off, it knows which string is playing and on which fret, and it triggers the same string more silently from a lower fret. The most complex mapping also has a Musical Artificial Intelligence module, which attempts to interpret the musical context from the data and select appropriate playing styles according to them.

3.1 Input hardware options

We are developing the air guitar user interface on three different interface hardware approaches. For the science center exhibition version of the Virtual Air Guitar we have two alternative choices for input hardware. Depending on which approach will result in a more convenient interface, we will use either computer vision or specialized hardware built for the guitar. For testing and developing different interaction approaches we use a virtual reality (VR) system that accompanies data gloves and a magnetic motion tracker. The VR system offers the most expressive user input but is obviously difficult to port outside the laboratory. The three hardware options are discussed below in subsections 3.2–3.4.

3.2 Magnetic tracker and data gloves

In addition to offering expressive input with several degrees of freedom, the virtual reality system promotes quick testing of new interaction methods. We can easily experiment combinations of artificial intelligence (AI) and gesture recognition input extraction methods on it.

The virtual reality system that we use is a Cave-like virtual room, called EVE [27]. Figure 10 shows a snapshot of playing the Virtual Air Guitar in the virtual room. The virtual reality interface hardware for the Virtual Air Guitar consists of two 5DT data gloves combined with a MotionStar magnetic sensor attached to both gloves. The location and orientation of the data gloves (magnetic sensors) are read at a rate of 100 Hz by the MotionStar magnetic tracker. The location accuracy of the tracker is in the order of one centimeter; the orientation accuracy is few degrees. The 5DT data gloves are simple and inexpensive. They measure the finger flexure through bending of optical fibers and return one integer value for each finger.

The input data from the tracker and the gloves is passed to an SGI Onyx 2 Infinite Reality main frame running the virtual room. The Onyx 2 has four graphics pipelines of which two are used to produce stereographic three-dimensional (3D) graphics on the three walls and the floor of EVE. The graphics are viewed through Crystal Eyes shutter glasses to produce real 3D imagery of the virtual world. In the case of the air guitar there is currently no other visualization than a static background scenery. However, we plan to make an interactive crowd simulation as discussed in section 5.1.

3.3 Optical tracking of hand movement

Using optical tracking for control parameter extracting consists of capturing video data of the user and then processing the data to extract the desired features. Every frame of the video is a single image of the user at a specific moment in time. Computer vision algorithms are used to analyze the individual images to extract information such as the location of some part of the user’s body. Usually temporal cohesion can be utilized for faster analyses. This means that the analyses algorithm takes advantage of information from previous images (animation frames). For instance, the location of some part of the user does not change much in between of consecutive
frames. Thus, the new location is first searched from the neighborhood of the previous one.

For a motion-intensive optical tracking system it is desirable to have a camera with a high update rate. This ensures steady control flow and prevents aliasing effects such as missing of fast movements. Most web cameras capture only 30 frames per second. However, some new cameras offer 60 frames per second. Of course special cameras offer even much higher rates. However, for possible continuation of the air guitar project we wished to build the system from parts that even a normal home computer owner could obtain for reasonable price. We also do not want the system to be dependent of highly special hardware. We obtained a Philips ToUcam PRO II web camera, which captures up to 60 frames per second.

The resolution of the captured video is not as important as the update rate. To minimize latency and to guarantee fast image analyses we use a 320*240 pixel resolution for the video capture.

At the moment of writing we have made only preliminary tests with the computer vision interface. We have tested primitive blob tracking using the free EyesWeb computer vision software [28]. The approach simply thresholds the image so that only areas of bright predefined colors remain visible. Then the X/Y location of the centers of these areas (blobs) is evaluated by counting the contribution of every pixel of the blob. We have not yet used the computer vision for controlling the guitar model. However, the virtual reality interface can be made to mimic the computer vision interface by producing only 2D location data.

We have a couple alternatives for what the user wears to produce the colored blobs to the images. One alternative is that the user holds a short mallet with a colored ball in his left hand and a plastic colored circle as an oversized plectrum in his right hand. Second alternative is that the mallet and the plectrum are replaced by colored stickers on the back of the user’s hands. Third alternative is that the user wears simple fabric gloves of a certain color. The gloves seem to offer the solution that is visible no matter how the hands are turned.

The sound model of the guitar with six strings and the effects chain consume so much computing power that we will run the computer vision software on a different computer, which will control the Mustajuuri sound software through a local net. The virtual reality interface software already controls the sound software in this way.

It seems likely that because the control software needs to implement the Mustajuuri software’s messaging system we will use a Linux computer for the computer vision instead of a Windows computer. Thus, we will move away from EyesWeb and use a computer vision library called PureData [20].

3.4. Special control devices

Special hand-held control sticks or gloves are an inexpensive alternative to the hand-tracking systems above. Using proper techniques for hand and finger positions and movements can make a relatively simple controller that is useful for example in the science center application. Below we will describe the development of sticks (see prototype in Fig. 11) that apply acoustic and mechanical principles to hand-tracking.

3.4.1. Acoustic positioning of player’s hand

There are two conceptually different approaches to wireless measurement of a distance of two small sensors. The choice is that of selecting a medium, i.e., whether to use electromagnetic or acoustic waves. Electromagnetics is successfully applied for example in the musical instrument Theremin, where distance is measured by sensing the capacitance between an antenna and the hand of the player. Another approach with electromagnetics would be to measure the phase difference between the sent and
received pulses. With these approaches the measurement could be done without acoustic interference, no matter how loud a band would surround the player of the virtual air guitar. A drawback is the need of RF electronics in the implementation.

Acoustic measurement of distance by propagation delay of sound has the advantage that it can be implemented with merely a loudspeaker and a microphone (and possible amplifiers) connected to a computer, leaving the signal processing to DSP. This is attractive since the virtual air guitar has in any case audio interfaces and DSP implemented. The use of sound however leaves the measurement to be done in a noisy environment, especially if there are for example percussionists next to the virtual air guitar.

In order to keep electronic and audio equipment minimal and simple, the measurement of propagation delay can be done using audio frequencies. To achieve an adequate SNR the sound pressure level needs to be high enough, thus making the measurement sound signal potentially audible. This is prevented with the use of only highest audio frequencies. If the common 44.1 kHz sampling frequency is used, the applicable frequency range in the measurement is limited to about 18–20 kHz. The lower limit is to keep a relatively strong signal, that is needed, inaudible, and the high limit is due to the anti-aliasing filters of the A/D- and D/A-converters.

The desired operating range of the distance measurement is 0.1–1.0 m, which corresponds to 0.3–3.0 ms time at the speed of sound. With 44.1 kHz sampling frequency the range translates to about 13–130 samples. While one sample interval corresponds to 7.5 mm and the minimum guitar fret spacing is about 10 mm, sub-sample accuracy of the distance measurement is highly desirable.

### 3.4.2. Time-delay estimation

Time delay estimation (TDE) is a common signal processing task with many proposed algorithms [29]. For example the maximum likelihood estimate of the delay between a signal and its delayed replica embedded in noise is the argument that maximizes the properly weighted cross correlation function. In a digital implementation, if the ratio of the sampling frequency to the signal frequency is high enough, this yields an accuracy of one sample interval. Sub-sample accuracy can be obtained with simple interpolation, e.g. by parabolic fit. In our case, however, the measurement signal is almost critically sampled and the resulting sampled cross correlation function doesn’t in general attain its peak in the vicinity of the underlying continuous function and a relatively complex interpolation algorithm is needed. The phenomenon is described in [30].

We found by experiments that a small off-the-shelf dynamic headphone loudspeaker and an electret microphone connected to computer soundcard yielded a good enough signal-to-noise ratio for our purposes. If only speech and typical background noise of a public space are considered, the strongest interfering signal in the 18–20 kHz range are the fricative $f$, $s$, $sh$, etc., when spoken close to the microphone. In practice the worst situation where the system should be operable occurs when the loudspeaker-microphone distance is 1 m, the microphone is off the main axis of the loudspeaker, and someone is pronouncing a loud $f$-phoneme at a distance of 70 cm from the microphone.

### 3.4.3. Envelope follower algorithm

To avoid the computational cost of the cross-correlation function, we devised a simpler algorithm which seeks...
Fig. 14: Output of acceleration sensor for a rhythmic plucking movement (solid line) and threshold for pluck event detection (thick dotted line).

the peak of the received signal envelope of a modulated pulse. The block diagram of the envelope follower type of TDE is illustrated in Fig. 12. A bandpass-filtered impulse is sent as the measurement signal, and the received signal is bandpass-filtered again, rectified (absolute value), and lowpass-filtered, yielding a smooth pulse envelope. The peak of the envelope is found by seeking the first zero-crossing of its derivative after the maximum. The estimate is further refined for sub-sample accuracy by interpolating the zero-crossing position with linear interpolation, see Fig. 13.

In the prototype control sticks shown in Fig. 11 we have used an electret microphone (Sennheiser KE 4-211) as an acoustic receiver that is sensitive enough and almost omnidirectional at 18–20 kHz. The measurement pulse is transmitted through a small headphone driver that has good response up to 20 kHz and is almost omnidirectional (within 5 dB for angles ±60°). We have found that many small earplug-type headphones have drivers that can be used as transmitters for this purpose to yield high enough received level.

3.4.4. Sensing of right-hand movements

Sensing of the right hand movements can be done in a simple way by using acceleration sensors that are available as microchips. We have applied the VTI Technologies chip SCA320-CDCV1G [31] that senses the acceleration perpendicular to the plane of the chip and delivers 0.15 V/g, when driven by a +5 V supply voltage.

The acceleration signal is first bandpass filtered to frequency range of 0.3–20 Hz. An example signal obtained for a rhythmic plucking movement of the right hand stick (see Fig. 11) is plotted in Fig. 14. Peaks of the envelope above a given threshold (thick dotted line) are used to send a pluck control event to the guitar synthesizer with amplitude proportional to the height of the peak.

Simple strategies for damping/muting of a string are: (a) no damping of previous vibration during a new pluck, (b) fast muting of previous note while activating a new one, and (c) damping when a negative acceleration peak is found. More advanced damping control can be realized by extra sensors (finger controls) in the control sticks.

4. PLAYING SUPPORT SOFTWARE

A real guitar, electric or acoustic, offers several different playing techniques. Basically all of the techniques alter only a few sound parameters: pitch, amplitude, damping and excitation. Our guitar sound model offers the same parameters for real time control. Excitation type, be it plectrum or finger, is altered by changing excitation sample data sent to the guitar sound model. The other parameters are sent to the guitar model using our control message language.

The three different input hardware approaches that we have taken offer different amount of input parameters to be mapped into guitar control parameters. The computer vision interface gives just the X/Y location of both hands. We can extract an estimate for hand velocity and acceleration from the consecutive samples of the location data. However, we do not have orientation information of the hands or the finger flexure information which we have in the virtual room interface. Currently our special control interface extracts the distance between the user’s hands and the acceleration of the user’s right hand. It does not detect finger movements either. Of course we could build also control keys for the left hand’s fingers but we haven’t yet attempted this.

The fewer parameters the input hardware offers the more of the control parameters of the sound model need to be automatically generated. The virtual reality system offers enough input parameters for implementing expressive free control but the computer vision interface needs artificial intelligence (AI) methods for producing expressive playing techniques.

The science center exhibition version of the Virtual Air Guitar should offer entertainment for a few minutes. It cannot require much practice before the user is able to produce pleasing results. Still, it should offer enough interaction so the user feels that he/she causes the behavior of the instrument instead of perceiving the instrument to be automatic.
4.1 Gesture recognition

No matter which input hardware is used, the raw input data from the input hardware is sent to the gesture recognition module for gesture extraction, see Fig. 9. The module then attempts to recognize meaningful air guitar gestures, which are then used for controlling the sound model either directly or through a phrase database or AI.

At first, the location data from both hands is stored in a buffer with time stamps. From the location data the gesture recognition block creates a velocity vector buffer. This buffer stores the difference vectors between consecutive locations in the location buffer. Before storing, the vectors are scaled with the time difference between the location samples resulting in velocity vectors representing the movement at each moment in the buffer in meters per second. In the virtual reality version of the air guitar, also the orientation vectors and finger flexures of both hands are stored into buffers.

As a base operating element for the gesture recognition, we made a simple algorithm that measures the change of direction along a given vector. The location data contains noise making the consecutive samples possibly switch direction couple of times at the peak points. We removed this by counting the dot product of several consecutive velocity vectors from the buffer with the given movement direction vector. The algorithm then defines that a direction has been changed only when three or more consecutive dot products are of the same sign after previous changing of the sign. With 100 Hz sample rate this results in 30 ms reaction delay from the direction change before the control event gets sent. However, as there is no tactile feedback of the moment when the hand should hit the strings, this does not bother. Also, if we would, for instance, trigger a pluck right after the direction change, it would come too early. With a 30 ms delay together with the other delay factors of the VR system the pluck is perceived to come at the right time.

To recognize plucking and strumming movements we use the direction change operator for the right hand’s movement comparing it with the direction of the up vector. When the hand changes direction from upwards to downwards and at the same time exceeds a threshold velocity, we extract a pluck gesture. Pluck and strum are identical from the recognition point of view. Their difference is defined by the playing mode. For instance, if the user is performing a predefined phrase, the phrase database defines if the trigger is a pluck or a strum.

Also vibrato is recognized with the direction change operator. The velocity of the left hand’s movement is measured along the vector running from the right hand to the left hand. The tempo of the left hand’s movement can be directly mapped to the frequency of the vibrato. If this is not used, the vibrato is simply switched on using the default frequency of 4 Hz.

The pitch of the guitar sound is defined by the distance of the user’s hands. The pitch goes either continuously by making all the producible pitch values between the maximum and the minimum distance, or it produces only the discrete notes from some scale. For instance, we made a test version that produced notes from a pentatonic scale. Combining the rough sound of the electric rock guitar with Chinese-sounding music scale produced an instrument with a weird feeling. There was an indefinable feeling that something is not right in the world of that guitar.

In the initial version of the virtual reality interface we used closing of the right hand for triggering a pluck. The guitar-like wag movement of the right hand is naturally more intuitive and more air-guitar-like. Also, the accelerometer and computer vision interfaces cannot extract finger flexure information. Thus, we changed the trigger to the intuitive strum motion of the right hand. However, as the pitch of the guitar sound is defined by the distance between the hands, waving of the right hand fluctuates the distance measure. This can be compensated by taking a few seconds long average of the right hand’s location. If both hands move to the left or right at the same time (the user moves his whole body by stepping left or right) the average location of the right hand needs to be shifted immediately.

Also the average of the right hand’s location proposes some problems for the distance measure of the hands. Some users trigger long strong notes by letting the right hand rotate down and behind the user after the pluck. The movement is so slow that it shifts also the average location significantly. We have not yet established a good two-sensor solution for this. Either we should recognize the long strokes and not let them affect the average or we should use an extra location sensor to mark the location of the user’s body. The distance measure would then be evaluated as the left hand’s distance to the body sensor instead of between the hands.

With the virtual reality input hardware we have implemented a free playing mode that gives the user direct control also of hammer-on, pull-off, slide and bend playing techniques. The guitar is always in a mode where the
pitch is not continuous but quantized to fret positions on the strings. Hammer-on is then possibly triggered if the middle finger of the left hand is bent while the right hand initiates a pluck. If the user now bends the ring finger of his left hand as the sound is still playing, a hammer-on is triggered. The hammer-on triggers a new more silent sound one fret line higher on the string used for producing the current sound.

Similarly than the hammer-on, a pull-off is triggered in the opposite bending order. If both, the right finger and the middle finger of the left hand are bent while a string is plucked/sounding and then the ring finger is lifted, we initiate a pull-off. The pull-off triggers a new more silent sound one fret line lower on the same string. A new hammer-on can be triggered after a pull-off by bending the ring finger again.

Slide is done in the same spirit than hammer-on and pull-off. When the user bends his left hand's middle finger, the slide mode is activated. If he closes also the ring finger, a hammer-on is triggered, but if he instead quickly slides the hand closer or further, a slide is done. The slide sends an event on each passing of a virtual fret line sending a silent trigger and rescaling the delay buffer of the sound model.

Bending is done by twisting the left hand around the imaginary neck of the air guitar. The forward direction vector of the knuckles is used for deciding how much to bend. Normally the hand is level and the vector points away from the user or a bit upwards. When the vector rotates more than a threshold in a predefined amount of time, the bend is activated. The orientation of the vector is stored as the bend begins. The amount of bending is then scaled to one note if the vector turns totally upwards.

Bending, hammer-on, pull-off, and slide are directly controllable only on the VR version of the Virtual Air Guitar as the other input methods do not offer finger state information or hand orientation information. On the other interfaces these effects are also realizable, but they are initiated by the AI.

4.2. Guitar control language

To add more expressiveness and liveliness to the playing, we added a new layer of abstraction to the interface: the ability to play musical phrases. Instead of mapping the input data directly to the parameters of the sound model, the user triggers prepared sequences of control parameter data. The system follows an existing musical piece and triggers notes from it according to the input data. Since playing air guitar is not so much about having the correct playing technique but rather more about getting into the feeling of a particular song, using phrases frees the user from having to concentrate on getting the right notes out of the guitar.

To create these phrases, there must be a notation for this, a way of representing what a real guitarist would do when playing. However, neither musical notation nor sound model control data is very useful for this purpose. On one hand, musical notation lacks in expressive information and the mechanical completeness required for computer processing, because the musicians are expected to interpret the score and add their own expression to it. And on the other hand, it would not be prudent to write down every change to each of the sound model’s parameters, as this would result in hundreds, even thousands of lines of data for a simple passage. Therefore, an abstraction between these two is required, a language for representing guitar playing techniques that can be processed by the sound model controller.

For this purpose, we have developed a language for describing musical passages played on a rock guitar for the Virtual Air Guitar. Because the scope of the language is very specific, it can be made very efficient and intelligent.

The guitar control language defines events with parameters, much like the MIDI language. Each event is assigned a time code for when it is supposed to be run, and the event has some parameters that define what it does and how. Unlike MIDI, however, the air guitar language is designed just for this one instrument, and the events contain are directly related to actions of a real guitarist playing a real guitar. For example, there is a ‘pluck’ event that corresponds to plucking a string on the guitar, and its parameters define which string is plucked, which fret is selected from the fretboard (i.e., the pitch), how strong the string is plucked, and when it is muted.

To compile a list of required events, we first needed to find out what real guitarists do when they play their instrument. In their article, Erkut et al. discuss some of the expressive parameters of the classical guitar [8]. Furthermore, Rossing discusses what a guitar actually sounds like from a physical perspective in his book [32]. Table 3 lists the most important guitar playing techniques in rock music.

These playing techniques are what the guitar sound
Table 3: The most important guitar playing techniques in rock music.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Description</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pluck string</td>
<td>Plucking a string</td>
<td>Sound</td>
</tr>
<tr>
<td>Place fret</td>
<td>Pushing a string towards the fretboard at a specific fret</td>
<td>(no sound)</td>
</tr>
<tr>
<td>Vibrato</td>
<td>Wobbling a string up and down with the finger that is pressing it at the fret</td>
<td>Pitch change in a sine wave (only upward)</td>
</tr>
<tr>
<td>Grace note bend</td>
<td>Pushing the string upwards, takes a short time</td>
<td>Note that has a bend up at the beginning</td>
</tr>
<tr>
<td>Measured bend</td>
<td>Pushing the string upwards, takes longer and has boundary between two notes</td>
<td>Note, smooth change to another higher note</td>
</tr>
<tr>
<td>Slide</td>
<td>Sliding fingers along the fretboard, usually going to a specified fret from a few frets above or below</td>
<td>Smooth pitch change in quants related to frets</td>
</tr>
<tr>
<td>Grace note hammer-on</td>
<td>Hitting a finger on the fretboard (usually preceded by a pluck), (for right hand tapping also), takes a short time</td>
<td>Grace note upwards, also damped sound</td>
</tr>
<tr>
<td>Measured hammer-on</td>
<td>Hitting a finger on the fretboard (usually preceded by a pluck),</td>
<td>Two rapid notes, the second higher, also damped sound</td>
</tr>
<tr>
<td>Grace note pull-off</td>
<td>Pulling a finger off the fretboard and re-plucking the string with it, while keeping a second finger on a lower fret, takes a short time</td>
<td>Grace note downwards, also damped sound</td>
</tr>
<tr>
<td>Measured pull-off</td>
<td>Pulling a finger off the fretboard and re-plucking the string with it, while keeping a second finger on a lower fret, takes a short time</td>
<td>Two rapid notes, the second lower, also damped sound</td>
</tr>
<tr>
<td>Mute</td>
<td>Lifting a finger from the fretboard but still touching the string</td>
<td>Cutting off the sound from a string</td>
</tr>
<tr>
<td>Palm mute</td>
<td>Performing a pluck while holding side of the right hand on top of the string</td>
<td>In conjunction with pluck, produces short rhythmic sound</td>
</tr>
<tr>
<td>Pick scrape</td>
<td>Pulling the plectrum along the strings towards the end of the neck</td>
<td>A screeching, falling sound</td>
</tr>
<tr>
<td>Harmonics</td>
<td>Touching a string over a fret bar</td>
<td>A bell-like sound</td>
</tr>
</tbody>
</table>

In addition to the playing techniques, the language must also be able to define timing. Not all of the playing techniques can be accurately tracked with all input mechanisms, and so some can only be triggered automatically by the musical phrase. For example, bending the strings is difficult to monitor by optical tracking due to lack of resolution and 3D depth. The phrase can still contain a bend, though, it is simply played automatically at a specified time after a trigger has been received.

Table 4: Events in the guitar control language.

<table>
<thead>
<tr>
<th>Event</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pluck</td>
<td>string to pluck, fret to select, time to mute strength</td>
</tr>
<tr>
<td>Hammer-on</td>
<td>fret to select, strength</td>
</tr>
<tr>
<td>Pull-off</td>
<td>fret to select after pull-off, strength</td>
</tr>
<tr>
<td>Bend</td>
<td>fret to bend to, time to bend in</td>
</tr>
<tr>
<td>Slide</td>
<td>fret to slide to, time to slide in</td>
</tr>
<tr>
<td>Slide on</td>
<td>direction, speed</td>
</tr>
<tr>
<td>Slide off</td>
<td>none</td>
</tr>
<tr>
<td>Palm mute on</td>
<td>none</td>
</tr>
<tr>
<td>Palm mute off</td>
<td>none</td>
</tr>
<tr>
<td>Vibrato on</td>
<td>speed, amplitude</td>
</tr>
<tr>
<td>Vibrato off</td>
<td>none</td>
</tr>
<tr>
<td>Pick scrape</td>
<td>speed</td>
</tr>
<tr>
<td>Mute</td>
<td>string to mute</td>
</tr>
</tbody>
</table>

Timing in the control language is expressed in a notation of measures, beats and ticks, a system commonly used by sequencer software. Each event has a time code such as 12:2:480, which would mark the note to play at 480 AES 117th Convention, San Francisco, CA, USA, 2004 October 28–31
Page 14 of 19
ticks after the second beat of the 12th measure.

With the control language, it is possible to build sequences that the Virtual Air Guitar then advances through along with the user. When a pluck gesture is recognized, the phrase database returns all the events in the sequence up to the next trigger, which can be another pluck. As an example, let’s say there is a sequence with a pluck event, a vibrato event, and another pluck event. When a user trigger is recognized, the phrase database returns the first pluck and vibrato events, and passes them on to the guitar controller to be played out at their correct time. It then waits for the next trigger to come, and when it does, it sends the final pluck event along.

Our tests included an interface for playing along existing musical pieces with this system. An entire piece is written in the control language, and the system advances to the next pluck event when it receives a trigger from the user. We found there are problems in knowing where in the sequence the user is at a given time. The user can activate notes, but they cannot choose which note to activate. If they lose their position, they cannot replay a note, and they have no indication of what note is the next one in sequence, or where in the sequence they are. The user can only play more notes at some arbitrary rhythm, but this does not help them regain the position.

Based on these findings, we implemented the option to reset the sequence by not playing anything for a few seconds. This made recovery from errors easier, but it did not prevent them. It can be speculated that some graphical indication of the sequence position could be helpful. For the science center exhibition version it is important to use well known songs. If the user does not know how the song goes it is almost impossible to guess the right relative durations of the notes and the playing becomes fruitless.

A second important discovery was that users easily get confused by the sequence automatically generating what sounds like a new pluck, even if it isn’t. This happens, for example, when there are hammer-ons and pull-offs in the sequence. Usually, guitarists use these techniques to play faster than simply plucking with the right hand would allow. It follows that air guitarists cannot move their right hand fast enough either, so we decided to have the system play these rapid hammer-ons and pull-offs automatically. This resulted in confusion, because sometimes the user hears one note, sometimes two when they play a pluck gesture.

While we have not yet tested it, we can hypothesize that different users are likely to have different views on how a certain well-known piece should be played on the air guitar. For example, some people may want to play the hammer-ons and pull-offs with a plucking motion, while others would use their left hand as real guitarists do. Others may confuse a song’s bass line as part of the guitar riff, and attempt to play along that as well. Finally, people are very particular about how their favorite songs should sound like, and even small glitches may ruin the experience.

Keeping in mind all the difficulties described above, playing along existing songs may not be as entertaining as it may seem at first. The user can simultaneously feel that they do not have enough control and that they have too much of it as well.

4.3. Musical artificial intelligence

For even more control over the final sound, a layer of artificial intelligence can be added between the input data and the musical phrases. Instead of following a single phrase, the gesture recognition system could be made to trigger short phrases when it recognizes certain gestures. For example, a simple pluck could call a phrase that triggers a note, applies vibrato to it to make the sound more interesting, plays a slide down at the end, and mutes the sound afterwards. Another phrase could trigger a sequence of notes so fast that it would be difficult to play using only the right hand as a trigger. And instead of controlling the pitch of the sound directly using the left hand position, the system could quantize pitches to match a certain chord.

By raising the intelligence of the system this way, the user can be allowed to play complex passages without needing to put considerable effort into mechanically challenging musical skills, and also without the need to recognize minute hand movements and finger actions on the guitar strings. Furthermore, by limiting ourselves outside of existing songs, we avoid the pitfalls of having to accurately reproduce existing music. The two play modes described previously are extended by three additional modes that both limit the user to a specific playing style and allow them more complexity within that style.

4.3.1. Rock solo mode

A rock solo play mode is introduced as an intelligent extension of the free playing mode described earlier. The user no longer controls the pitch directly, like in the pentatonic quantization in Section 4.1, but the gestures are
run through a system that generates the pitch by an intelligent random process instead. We know the playing style is a rock solo, because we define it to be so. The user hears an accompaniment track that plays a simple eight-bar, three-chord rock rhythm. This restricts the user into a certain playing style as well as a certain chord pattern, but these are not as important as is playing the actual solo.

By using a Hidden Markov Model system [33], we can calculate a note that is most likely to be played if the user moves his/her left hand by a certain amount, given the previous notes, the chord from the rhythm pattern, and knowledge of the playing style. Bends, slides and vibrato can be added at appropriate positions of the pattern. Many rock guitarists use a combination of plucks, hammer-ons and pull-offs to play high notes very rapidly in solos. Since this is rather difficult for an air guitar player to do, the system can automatically generate notes quantized to the rhythm pattern if the user simply moves his/her right hand rapidly enough. However, we must keep in mind that triggering notes when the user does not expect them can lead to confusion.

4.3.2. Chord strumming mode

The chord strumming mode differs from the rock solo in that a recognized trigger gesture plays several or all six strings in a strum (raking the fingers across the strings from top to bottom) instead of just one string. The speed of the hand also affects the speed of the strum. In this playing style, vibrato and bends are not used much, and so the AI does not have to reproduce them.

Strumming is used for creating a chord backing to a song, not a solo. It is more important to be able to select appropriate chords instead of the exact pitch of a string. Thus, the strumming mode recognizes left hand movements as chord selection. Chords are selected by a similar Hidden Markov Model process as pitch in the rock solo mode.

4.3.3. Chord picking mode

The acoustic picking mode is similar to the strumming mode, only with the difference that each movement of the right hand triggers one string instead of all six. The strings are played in succession, so the first string is plucked first, then the second, and so on. The pluck order can be varied by using different phrases or randomization.

4.3.4. Rock riff mode

Rock riffs are short, easily memorable rhythmic patterns. In the rock riff playing mode, the user can play riffs with two strings with a fifth interval (the higher string is played with a note seven semitones above the lower string) that is used commonly in riffs. Pitch is selected in a similar manner to the rock solo mode.

Essential to this playing mode is the ability to mute the strings, or even perform palm mutes. The user must have the ability to perform some action to mute the strings. Thus, this playing mode is difficult to implement with optical tracking, as it may not offer enough information for recognizing actions such as closing the left hand into a fist, or turning it at the wrist.

5. FUTURE WORK

VR and computer vision systems always introduce some latency. It is important to know how much latency can be allowed for different control paradigms and how it affects the control. We have studied these issues with user tests for continuous sound gesture control interfaces [34, 35].

With the virtual air guitar we can study the latency issue for guitar control and from the point of view of different levels of AI supporting the playing.

The gesture recognition and artificial intelligence modules can be further developed by conducting tests on a number of air guitarists playing along existing songs. With the knowledge of what is happening in the song and data from the players acquired by motion tracking, it could be possible to extend the AI to add expression according to certain gestures, and even predict what the user wants to play before they have done so.

5.1. Virtual crowd

As mentioned, the virtual reality system allows stereographic three-dimensional (3D) visualization that surrounds the user. The virtual world is projected on the walls of a three meters wide cube. EVE has three back-projected walls and a floor, for which one the image is projected from above. The subjects wear shutter glasses that filter different images through for both eyes letting the subject see three-dimensionally.

We intend to make a dynamic crowd simulation using the 3D visualization. We will make visual and audible surroundings for the player. The player will be standing on a virtual rock stage with a large virtual audience surrounding the stage. The audience will consist of photographed texture animations taken of different people. These textures will be placed on billboard polygons in
the virtual world, one per audience member. Thus, the crowd will consist of hundreds to few thousand people replicated from about ten different individuals.

In addition to the visual animation the crowd will be audible and will react dynamically. The audio will consist of recorded samples such as constant noise and yells. When the player comes to the stage the audience will go wild with noise and move a lot. When the user starts to play, the crowd falls more silent and moves less. When the user stops playing the crowd goes wild again. Also after intensive passages the crowd becomes louder for a while.

When we will make a version of the crowd for the science center exhibition, it will most likely be only aural. Crowd on a monitor or even on video projector will not offer much immersion compared to the immersive surrounding crowd that can be produced in the virtual room.

6. SUMMARY AND CONCLUSIONS

In this paper we have described the development of a playable virtual instrument called Virtual Air Guitar (VAG). It consists of hand-held controllers and a guitar synthesizer with sound effects. Three different user interfaces for controlling the instrument are experimented: data gloves used in a virtual room, optical tracking of hand movements, and special control sticks using acoustic and acceleration sensing of hand movements. The control parameters are mapped to synthesis control parameters in various ways: from direct control of pitch and plucking to artificial intelligence based advanced control.

A simple version of the Virtual Air Guitar will be applied as a demonstration system in a music-related exhibition in the Heureka Science Centre, Vantaa, Finland. From a research point of view, an advanced version of the Virtual Air Guitar will be used as a case study on the controllability and expressivity of virtual musical instruments.

Sound demonstrations and information related to the Virtual Air Guitar available at:

http://www.acoustics.hut.fi/demos/VAG

7. ACKNOWLEDGMENTS

This work is supported by the IST/ALMA project (ALgorithms for the Modelling of Acoustic interactions, IST-2001-33059) and the Academy of Finland projects SA 104934 (CLAPS) and SA 53537. We are grateful for the interest of the Heureka Science Centre in the development of the virtual instrument and for the help of VTI Technologies in acceleration sensors. Special thanks are due to Hannu Pulakka who converted the tube distortion algorithm to BlockCompiler.

8. REFERENCES


