Human–Computer Interaction and Music

Isabel Barbancho, Alejandro Rosa-Pujazón, Lorenzo J. Tardón and Ana M. Barbancho

1 Introduction

In this document, we offer some insight into the potential of combining advanced interaction paradigms with sound and music for the development of innovative interactive audio applications. Therefore, in this chapter we present novel ways to interact with the music by means of using advanced natural human computer interfaces. Furthermore, the basics of specific applications which are currently being developed will be briefly presented. A motion-based paradigm is considered in the context of music signal processing to provide an innovative and immersive experience in audio and music applications.

1.1 Review of Previous Works

Human Computer Interfaces have long moved beyond the conventional setting of a single user sitting in front of a desktop computer. Nowadays, the latest technological advances in the fields of motion tracking or speech recognition have allowed for the definition of more complex, enriching interaction metaphors. Also, the development and proliferation of low-cost off-the-shelves devices, such as Microsoft's *Kinect* camera or the Nintendo *Wiimote*, make it possible to provide more intuitive ways of interaction with a given computing device (Villaroman et al. 2011).

An adequate fusion of auditory and visual cues is critical in the conception of natural human computer interfaces (Shivappa et al. 2010), since vision and hearing are the primary senses used by humans to comprehend and interact with the world. Thus, in order to create a fully immersive experience, it is necessary to carefully

I. Barbancho (✉) · A. Rosa-Pujazón · L. J. Tardón · A. M. Barbancho
Department of Ingeniería de Comunicaciones, Universidad de Málaga, Málaga, Spain
e-mail: ibp@ic.uma.es

R. Bader (ed.), *Sound—Perception—Performance*, Current Research in Systematic Musicology 1, DOI: 10.1007/978-3-319-00107-4_16,

design the presentation of audiovisual information. As such, music is an integral part of the auditory modality. Moreover, musical interaction itself can constitute the basis and main focus of a certain interface for a wide assortment of applications: music learning (Wang and Lai 2011), musical instrument creation/simulation (Jorda 2010), music-guided rehabilitation (De Dreu et al. 2012), etc.

One of the most well-known types of applications regarding musical interaction is that of videogames. Games such as *GuitarHero* (GuitarHero 2011), *SingStar* (SingStar 2004) or *RockBand* (RockBand 2012) are some of the most prominent and popular applications of this kind, especially on western markets. *Guitar Hero III: Legends of Rock* became the first single game ever to surpass $1 billion in sales (Craft 2009). This proves the economic relevance of this genre. However, an analysis of 20 releases showed that these rarely respect the creative component of actually playing music (Grollmisch et al. 2009). Mostly, the player has to rhythmically trigger buttons according to predefined sequences. This mostly requires dexterity and swift reflexes, whereas the possibilities to actively participate in music creation are only marginal.

Some rip-offs of the most popular titles exist in the open source world with titles like *FretsOnFire* (FrestsOnFire 2006), *UltraStar* (UltraStar 2010). It is remarkable that the editing of the songs (alignment of notes and lyrics with the songs) for these titles has to be done manually, this is why *UltraStar* features a built in editor. For commercial titles, game studios use proprietary editors and formats. There are only a few attempts to use automatic music processing in these kinds of games. In (Barbancho et al. 2009) a description on how to generate game packages for *FretsOnFire* is presented. *AudioSurf* (AudioSurf 2008) or audio processing as proposed in (Migneco et al. 2009) are examples of action games based on automatic, real-time analysis of music features.

Nevertheless, it would be definitely unfair to judge music games as being nonmusical. They do foster the interest of children and adolescents in music, some gamers are even encouraged by these games to learn a real instrument. For example, the Beatles Rock Band Edition (*BeatlesRB* 2009) allows up to three singers to sing in harmony and rewards the players with high scores if they succeed. Such features are somewhat useful for self-study voice education (*Mikestar* 2009). Furthermore, recent studies seem to confirm that these kinds of applications can actually help children develop musical skills and knowledge (Gower and McDowall 2012). Thus, despite their limitations, they can serve as a gateway to musical concepts for potential users, especially with regards to rhythm.

Musical interaction has recently become a hot topic. Antle et al. (Antle et al. 2008) proposed a system to connect body movements to output sounds. The experiments conducted showed that learning processes could be improved through the use of such interaction metaphors. Further research was performed regarding the use of tangible interaction in order to manipulate the pitch, volume and tempo of ongoing tones (Bakker et al. 2011). Another study (Holland et al. 2010) made use of a haptic vibrotactile device set (the Haptic Drum Kit) to aid the learning process of rhythm and rhythmic patterns.

A good example of musical exploration and creation using physical body movements as an interface can be found in the works by Khoo et al. (Khoo et al. 2008). Their study depicts a system capable of mapping motion features onto acoustic parameters (such as frequency, amplitude, tempo, or time signature) for real-time expressive rendering of a given piece of music. At the same time, visual feedback was projected accordingly on a screen. Therefore, this system allowed users to interact with music in an intuitive and explorative way, lowering the barriers towards the understanding and appreciation of music features. A similar interaction metaphor was considered in (Castellano et al. 2007). In a similar fashion, it is possible to modify the visual patterns presented by means of speech or sung voice, making the voice "visible", as shown in the works conducted by Levin and Lieberman (Levin and Lieberman 2004), or directly interacting with a virtual character (Taylor et al. 2005; Mancini et al. 2007).

Musical interaction can act as a strong motivator, but it can also fulfill an important role as a way to allow users to acquire or expand their knowledge on music theory. In general terms, the mechanics and abstract concepts of music are not usually known to most lay people. Furthermore, in order to learn the different aspects of music theory it is necessary to devote a considerable amount of time to such purpose. On the other hand, visitors to physical museums are often overwhelmed by the vast amount of information available (Karimi et al. 2012). In this regard, musical interaction allows for a learning-by-action exploration, lowering the barriers of the inherent abstract nature of many of these concepts and making the global experience much more accessible and enjoyable.

1.2 Chapter Organization

As previously stated, the aim of this text is to portray how innovative user-computer interfaces can be used to provide new experiences and forms of interaction with music. More concretely, Sect. 2 presents a brief overview of the technologies available for the design and implementation of interaction paradigms revolving around human body motion tracking. After the different alternatives have been identified, the chapter will cover the option chosen to implement the natural human–computer interface as well as the justification behind this decision and a succinct review of the capabilities for motion tracking of the Open Natural Interaction (*OpenNI*) framework. Then, Sect. 3 will present two applications which use motion-tracking to allow the user to modify some musical features in the sounds played. In addition, an overview of alternative motion-based interaction paradigms will be described in order to more widely show the potential of these interaction techniques in musical applications. Finally, Sect. 4 will briefly present the conclusions gathered from this work.

2 Motion Tracking: A Camera-Based Approach

2.1 Technologies for Motion Tracking

In the next lines, a brief summary of the most commonly used motion capture technologies is given. As it is not the scope of this chapter to exhaustively cover the distinct technological aspects behind these technologies, only the most prominent and relevant aspects will be summarized. For further details, the following works should be reviewed: Welch and Foxlin (Welch and Foxlin 2002), Baratoff and Blanksteen (Baratoff and Blanksteen 1993), Perry et al. (Perry et al. 1997).

Traditionally, the most commonly used alternatives for motion tracking can be summarized into the following five categories, depending on the main technology employed: optical, electromagnetic, acoustic, inertial and mechanical sensors.

- Optical trackers in general show high update rates and short lags. The data acquired are usually quite accurate, and electromagnetic noise and room temperature have little to no effect on them. However, they suffer from the line of sight problem: any obstacle between sensor and source will create an occlusion which can seriously degrade the overall performance of the tracking system. Ambient light and infrared radiation can also adversely affect the performance. As a result, the environment must be carefully designed to reduce these interferences as much as possible.
- Electromagnetic trackers have been quite popular, but they can give inaccurate measures. They usually show latency problems and they are also very sensitive to the presence of large amounts of metal and conductive materials in the surrounding area or other electromagnetic fields, such as those that would be generated by large computer equipment, displays, etc. They have some important advantages, however, such as not requiring direct line-of-sight between the trackers themselves and the source, and they usually have a small and ergonomic size.
- Acoustic sensors lie somewhat in-between the previous ones. They have a restricted workspace volume and require direct line-of-sight (although they are not as affected by this as optical trackers). Time-of-flight trackers usually have a low update rate, and phase-coherence trackers are subject to error accumulation over time. Additionally and more importantly, both types are affected by temperature, pressure changes and the humidity level of the environment. Furthermore, the effects of echoing and the presence of other nearby sonic sources can have very negative effects on their measurement.
- Inertial systems are already available in chip form. Inertial sensors are completely self-contained. There are no line-of-sight requirements, no emitters to install, and no sensitivity to interfering electromagnetic fields or ambient noise. They also have very low latency (typically a couple of milliseconds or less) and high sampling rates. The main disadvantage of inertial trackers is that they are

very sensitive to drifting errors, since they do not actually measure position and/or orientation directly, which makes them ineffective when absolute measures are needed.
- Mechanical trackers offer many advantages, such as almost complete environment independence, and potentially highly accurate, fast and low latency measurements. Unfortunately, mechanical trackers are very cumbersome. They can be bulky, heavy, and severely limit the motion of the user, giving rise to ergonomic issues.

As a conclusion, there is no optimal solution when it comes to making a choice. The advantages and disadvantages of each technology must be carefully assessed according to the requirements of the task at hand.

We have decided to use a camera-based system. A camera-based tracking system can be seen as a passive subtype of the optical tracking technology, since the tracking points of interest will not actually emit light by themselves, instead the system will rely on the light reflected on natural surfaces. We have followed this approach because it offers an off-the-shelf, inexpensive solution that minimizes intrusiveness (Gleicher and Ferrier 2002), constituting a good solution to implement high precision motion tracking. Nevertheless, while humans can easily identify motion from a recorded sequence telling from their own experience in the real world, computers face some difficulties (Gleicher and Ferrier 2002), for mapping a 3D world into a 2D movie involves an important loss of information. In order to circumvent such limitation, we have opted to use a Z-camera, that is, a camera that is capable of capturing the 3D scene, by means of a depth map which assigns to each pixel a value related to the distance between the camera and the point in the 3D scene which was projected onto that pixel. In particular, a Microsoft *Kinect* sensor is used.

2.2 The OpenNI Framework

Kinect is a composite device consisting of an infrared (IR) projector, an IR camera and a RGB camera. The infrared camera projects a pattern of infrared points which are then used to triangulate points in space (Smisek 2011), thus allowing to determine the depth of the objects in the scene. In November 2010, the company *Primesense* released their own open source drivers, as well as a middleware motion tracking module called *NITE*, both integrated as part of the *OpenNI* framework. By using the *OpenNI/NITE* framework, it is feasible to achieve advanced user identification and motion tracking capabilities. This section aims to briefly present the key features to consider in the development of our interface using a *Kinect* device.

The purpose of the *OpenNI* framework is to provide the means to develop human–computer interfaces that achieve "natural interaction", that is, an interaction based on human senses, especially on human hearing, motion and vision. In

order to actually accomplish this objective, it is necessary to resort to hand and body gestures and motion tracking, speech and voiced command recognition, etc. *OpenNI* offers a flexible framework to make use of natural interaction devices (such as 3D sensors) and the middleware needed to control those devices.

The layer view of the *OpenNI* framework is shown in Fig. 1. It includes complex components, such as Production Nodes, Production Chains or Capabilities. However, it is not the aim of this section to thoroughly cover the possibilities of the *OpenNI* API. Let it suffice to say that a Production Node is a component that outputs a certain type of data to be used by either other Production Nodes or the application itself. Examples of such components would be an Image Generator (which generates colored image-maps) or a Depth Generator (which produces a depth map). In the particular case of the *Kinect* sensor module, both the RGB data and depth image data are accessible through the use of Production Nodes, where examples are shown in Fig. 2.

One of the key features of the *OpenNI/NITE* framework is the Production Node called Scene Analyzer. This component analyzes a depth image produced by a lower-level node to look for figures in it. Thus, the foreground, background, the floor plane, as well as human figures are identified (PrimeSense 2011a). The data are then processed into a labeled depth map as output. *NITE* processes this data to generate a label map that allows for specific user tracking in the scene, using a user segmentation algorithm (PrimeSense 2011b). This label map holds information regarding the detection of human shapes, in such a way that each pixel is associated

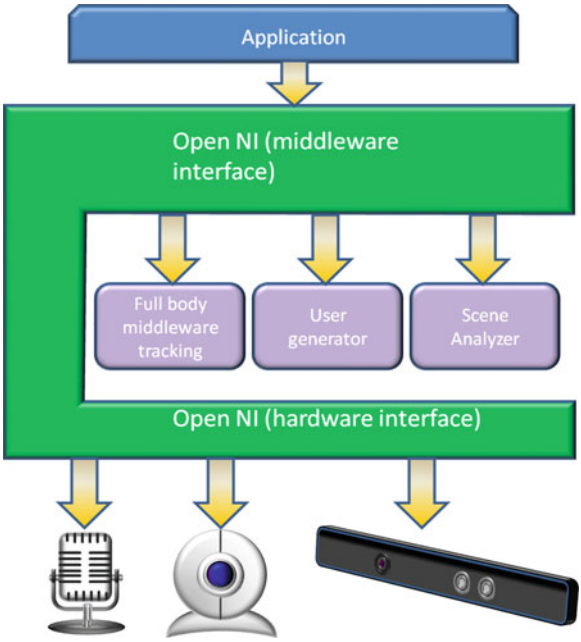

Fig. 1 An example of the abstract layer view of *OpenNI* (based on image in (PrimeSense 2011a))

Fig. 2 Example of Kinect's production nodes output, depth image in rainbow scale (*left*) and RGB image (*right*)

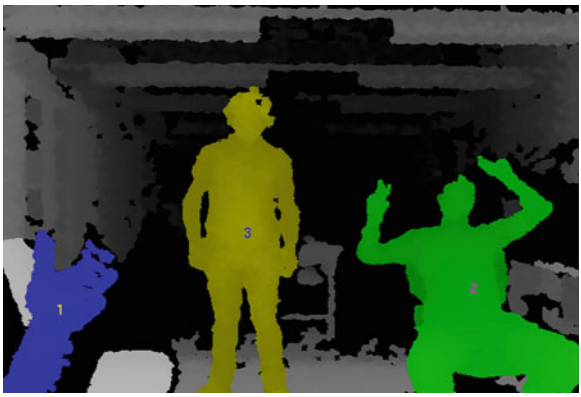

Fig. 3 User segmentation labelled depth image, identifying three different users from non-user pixels (background)

with a label which in turn indicates whether that pixel corresponds to the background or to one of the figures identified. An example can be found in Fig. 3.

Finally, the most critical feature for the implementation of a human–computer interface for musical interaction applications could be considered to be *NITE's* skeletal tracking production node (PrimeSense 2011b).

NITE's skeletal model makes use of a total of 15 skeletal nodes (of 24 possible nodes supported by the *OpenNI* API). The correspondence of such nodes with human joints can be seen in Fig. 4. This production node processes the labeled depth image generated by the user segmentation node to calculate the skeletal joints position and orientation values for the fifteen nodes of one of the users identified in the previous step. Note that joint position coordinates are far more stable than joint orientation ones, which are actually quite noisy. Thus, it is rather preferable to rely on joint position measures to prevent incorrect measures.

After a sufficiently stable model of the user's joint positions has been found, it is possible to implement different kinds of interactive models based on the motion detected for each of the nodes.

In the next section, we will cover the implementation of specific examples of musical interaction by means of skeletal motion tracking and musical signal processing.

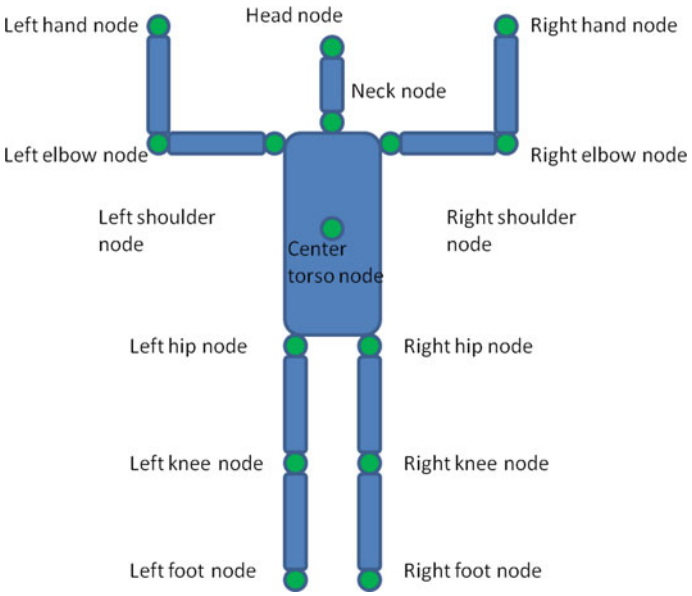

Fig. 4 *NITE's* skeletal model

3 Interaction with Music: Examples of Applications

So far, we have discussed the advantages and drawbacks of different tracking technologies and we have briefly skimmed over some of the possibilities that the *OpenNI/NITE* framework opens for the use of the *Kinect* camera in an application with stress in music interaction. However, we have not yet covered any specific example. This section provides examples of actual applications developed using a natural interaction interface based on *Kinect*. More specifically, this section will present two of such applications: first, an application where the user's hand movements induce pitch shifting in a sound source being reproduced, and secondly, an application that detects a drum-hitting-like gesture as a cue to actually play a drum-like sound. Finally, a collection of other possible paradigms for musical interactive applications is portrayed.

3.1 Pitch-Shifting by Motion Tracking

Pitch shifting is the process of changing the pitch of a given piece or sample without affecting its duration or speed. This process, along with time-stretching, is used, for instance, to match pitches and tempos of two sound excerpts that are to be mixed, to correct the intonation of instruments or singers, or to produce special effects such as increasing the range of a given instrument or adding a chorus-like

effect to a single singer performance (by mixing transposed copies of the original monophonic voice).

The most straight-forward way to implement pitch shifting would be to simply resample the clip to be processed, that is, to rebuild the original continuous waveform and sample it again at a different rate. However, this process also changes the duration of the signal. This means that if the signal were to be played at the original sampling frequency, the audio clip would sound faster or slower. In order to keep the duration unchanged, it would be necessary to time-scale the input signal by the same factor as it is going to be oppositely scaled by the resampling process. This stage can be implemented by using a time-stretching algorithm that does not introduce changes to the original pitch, such as the synchronized overlap-add method (Zölzer 2002). The order in which these two operations are applied can be interchanged (example in Fig. 5).

Another alternative is to use a phase vocoder to apply a pitch transposition transformation to a representation of the audio data based on a time–frequency model. One possible implementation of the phase vocoder multiplies the input audio signal by a sliding window which is nonzero for only a finite period of length N samples. This divides the input audio waveform in chunks or frames, which are transformed to the frequency domain by using the Fast Fourier Transform (FFT). This process is called Short-Time Fourier Transform (STFT), and it is mathematically represented by the equation Eq. 1.

$$X(n,k) = \sum_{m=-\infty}^{+\infty} x(m)w(m-n)e^{\frac{-j2\pi km}{N}} = |X(n,k)|e^{j\phi(n,k)} \quad (1)$$

Equation 1: Short-Time Fourier Transform for time–frequency representation.

In this equation, x(n) represents the digital audio samples, w(n) is the sliding window chosen, and X(*n, k*) represents the short-time spectra of each frame.

After the short-time spectrum has been calculated, it is possible to apply an operation or transformation by either manipulating the magnitude or the phase of each sample at each of the N frequency bins given by the FFT. Once the desired operation has been performed, the data is transformed back to the time-domain by using an Inverse FFT (IFFT), and then the processed output signal is conformed by adding and overlapping all the different frames previously calculated.

The summation of the IFFTs of all the *X*(*n, k*) allows the synthesis of the transformed signal back to the time domain. This implementation of the phase vocoder is refered to in (Zölzer 2002) as Direct FFT/IFFT approach. An alternative

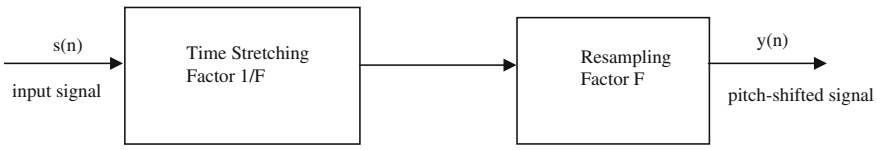

Fig. 5 Pitch shifting by resampling and time-stretching

way to implement phase vocoders is that of the filterbank or "Sum of Sinusoids" approach (Zölzer 2002).

When partitioning the signal into frames it is important to select an appropriate window. In particular, we have used a Hanning window in our implementation, and the windowing process is applied with an overlapping factor of 87.5 %, using a frame and FFT size of 1,024 samples. For each $X(n, k)$ calculated, real frequencies represented by each of the 1,024 bins are extracted (taking into account the phase $\varphi(n, k)$ and the delay introduced by the overlapping windowing process). Then, the frame is pitch-shifted by multiplying each frequency value by the corresponding pitch-shifting value. Afterwards, the IFFT is calculated using the same magnitude values as the original frame, but the phase values for each sample at each frequency bin are extracted from the transposed frequency values. Thus, the synthesis process yields an effectively pitch-shifted signal (an example can be found in Fig. 6).

After the selection and implementation of the pitch-shifting scheme, the next stage is the integration of such a process into a human motion tracking system to actually perform the pitch-shifting effect. To this end, we designed and programmed a simple virtual environment integrating the *OpenNI/NITE* API to get access to user motion data detected by a *Kinect* device in real time. The virtual environment and the application were programmed in C/C++ using a set of different libraries and frameworks: OpenAL for audio management, OGRE3D for graphics, OpenCV to manage the bitmaps extracted from the *Kinect* device, and *OpenNI/NITE* to implement natural interaction.

A screenshot of the virtual environment developed is portrayed in Fig. 7. Each of the fifteen nodes of the user's tracked skeleton was represented visually by spheres with a metallic texture. The information extracted from the depth image and the user segmented image was also used to create two small billboards to offer additional information to the user.

Users interact in the virtual world by using both their left and right hands. More specifically, during the start-up, the application loads a previously selected audio file. Left hand motion is used to start playing the audio or to pause the playback. This action was implemented by performing a simple "push-button-like" gesture to toggle the playing state of the song. Thus, playback switches between on and off whenever the user's left hand is moved forward so that the distance between the hand and the torso becomes larger than a certain amount over the Z axis, towards the camera.

While the music is playing, the user has the possibility of altering its pitch in real time by simply raising or lowering his right hand. In particular, if the right hand is kept approximately at the same height (Y axis) as the user's head height, then the song is played without any alterations. If the right hand is raised or lowered with respect to the reference height, then the pitch is shifted, increasing or decreasing respectively, according to the right hand's position. Pitch transposition is limited to a maximum of 12 semitones with respect to the original value. The highest pitch shift is applied when the user's right hand Y coordinate is approximately 50 cm larger than the head's one. Similarly, the lowest pitch shift possible

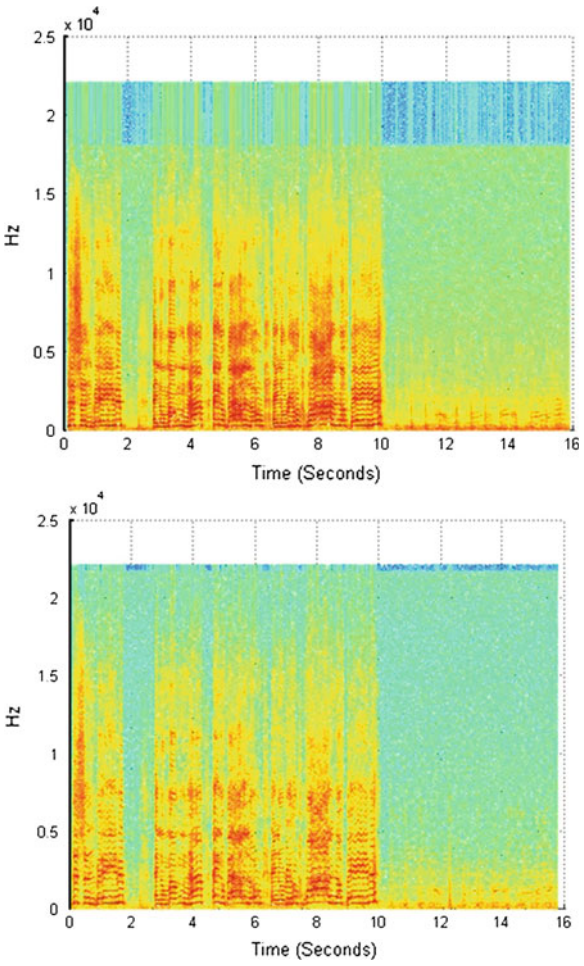

Fig. 6 Spectrogram of a vocal audio excerpt (*up*) and its pitch-shifted version (*down*). A 3 semitones shift is applied

is performed when the hand is about 50 cm smaller than the user's head height. In intermediate positions, the pitch shifting introduced is continuously distributed along the interval of semitones considered (−12, 12) proportionally to the relative height of the hand with respect to the head. Such behaviour is represented by Eq. 2, which portrays the factor F by which the audio excerpt's pitch is shifted (*relative Height* is measured in mm in this equation). Figure 8 illustrates an example of this procedure.

$$F = 2^{\frac{relative\ Height}{41{,}67 \times 12}} \tag{2}$$

Equation 2: Transcription of tracked movement into a pitch-shifting factor.

This application has been conceived from a pedagogical point of view. By correlating pitch rising and lowering with similar rising and lowering movements

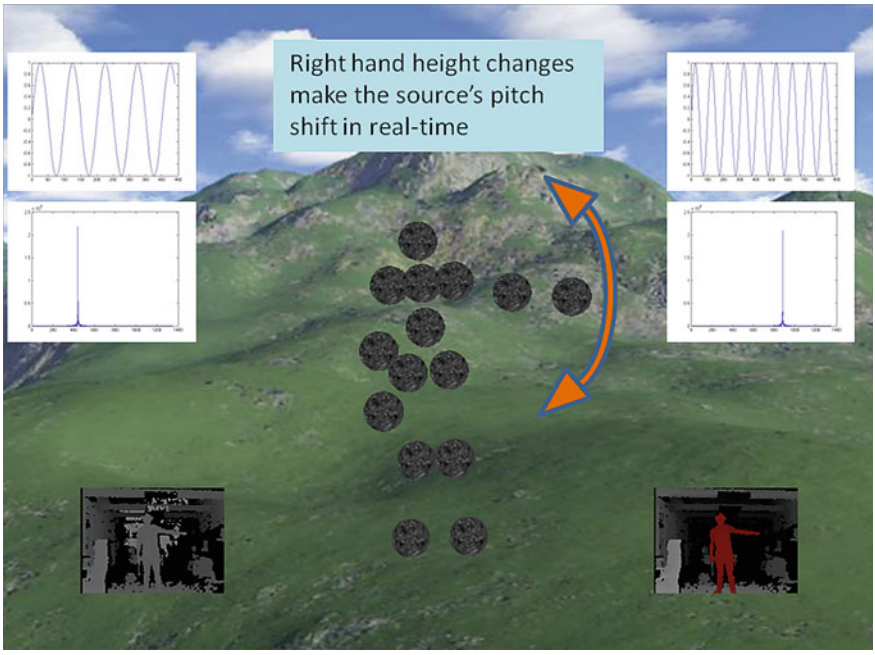

Fig. 7 Screenshot of the virtual environment developed for the pitch-shifting application

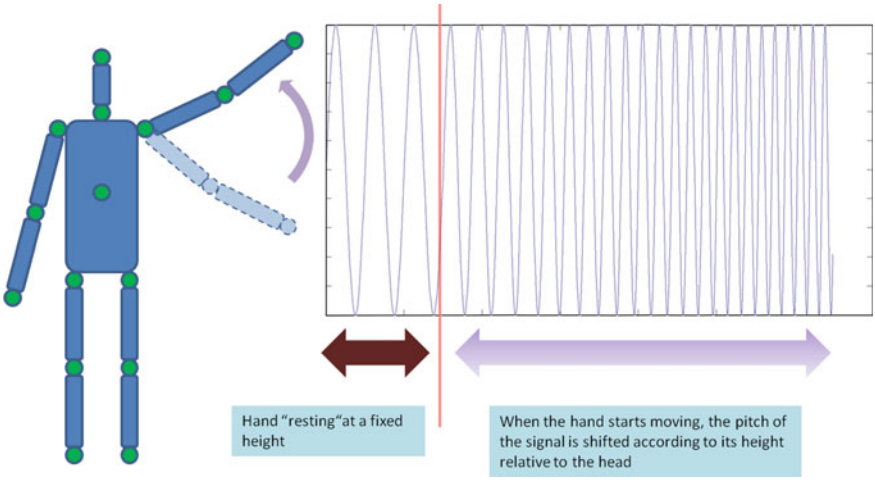

Fig. 8 Example of pitch-shifting of a single tone. When the hand moves from the blurred position to the end position, the pitch shift factor changes according to the height of the hand at each instant

of the hand, it becomes easier to understand the musical concept of pitch, especially for children.

The idea of coupling body motion with the generation of sound effects can be a useful tool for music learning, as previously indicated in the introduction, making the experience more enjoyable and explorative, and, at the same time, acting as an additional motivator towards the learning process.

3.2 Kinect-Battery: Playing Drums with Kinect

In the previously presented application, motion tracking was used in combination with music signal processing to achieve real-time pitch transposition. In this second application, we present a system that is capable of emulating a simple drum-like instrument by capturing user hand motion.

A first approach to the implementation of a simple air-drum would be to geometrically define a virtual drum in a given position in space. Since the *Kinect* 3D sensor actually provides coordinates for each position joint within its field of view, it could be possible to define the position and dimensions of a virtual drum within that space. Nevertheless, several usability drawbacks can be found in this approach. First of all, the most glaring problem when the virtual drum is located at a fixed position is that the drum would be "invisible" in the real world. Thus, unless the user is wearing a head-mounted display device, the user will need to keep watching an external display to find the drum. This can be easily overcome by requiring the user to stay at a particular position. Another issue is that the drum should be resized according to the height of the user, to ensure that children and adults alike can use it comfortably.

While these are definitely minor issues, they can pose an unnecessary inconvenience for user comfort. Therefore, an immediate refinement to the previous approach would be to virtually attach the virtual drum to the user; this can be achieved by placing the drum in a fixed relative position with respect to the user. This modification removes the previous issues and does not restrain user movements.

Figure 9 shows a screenshot of an application running this implementation. The framework of the application is the same one used in the pitch-shifting application. The virtual drum is represented by a simple geometric shape. Whenever the user moves either hand downwards and "hits" the upper top of the drum, a drum sound is played (see Fig. 10 for a schematic representation of this model). It is important to notice that the virtual drum is obviously intangible in the physical world. Therefore, the user might force the system to keep playing a sound by constantly waving the hand in short up-and-down moves near the top of the drum, creating a very unrealistic and unnatural sound output. To prevent this, both hands have to be raised a certain height before being capable of triggering a new sound when "hitting" the top surface of the virtual drum. The speed of the downwards movement is also taken into account to modify the intensity of the sound played

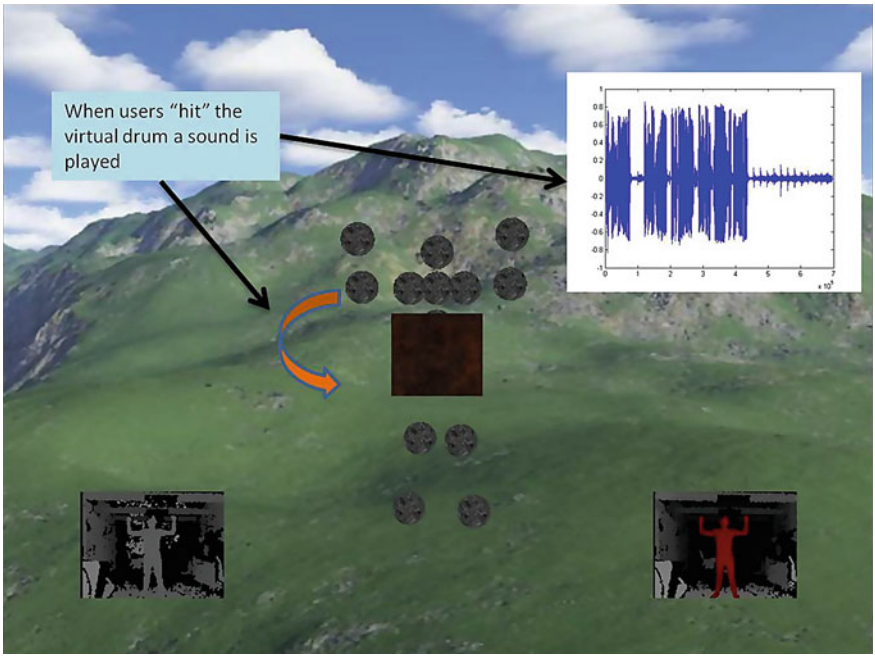

Fig. 9 Screenshot of the virtual environment for the drum simulator

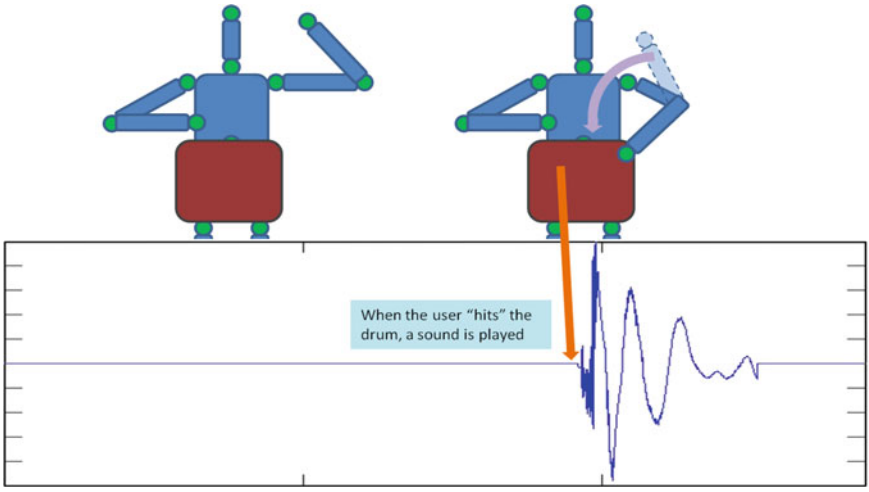

Fig. 10 Illustration of the "hit-drum-to-play" model

accordingly. A drum kit functionality can be easily implemented by creating several virtual drums distributed around the user.

A second approach to the implementation of the air-drum would rely strictly on user's motion to infer when the user "intends" to hit the drum. More specifically, the system would follow the movement of the hands, and whenever a drum-hitting gesture was detected, it would play the corresponding sound. In this second case, there would be no virtual drum, as the triggering of the sound would not depend on the exact position of user's hands but on the identification of drum-hitting gestures.

There are several reasons why this approach constitutes a more interesting approach. First of all, it removes the need for "invisible" drums. Furthermore, noisy samples of the hands' position can be more problematic when using the first approach, since it is more reliant on the precision of the user when "hitting" the drum.

The second reason why resorting to a drum-hitting recognition model can be a more enticing prospect is that the 3D sensing technology used to track user movement has some limitations in terms of accuracy, delay, etc. In particular, the delay between user movements and user's avatar transcription of the motion tracked could be as high as 200 ms. This is a hurdle that cannot be easily overcome with the former approach. Notwithstanding, it is feasible to look for features in the user tracked motion that can offer some insight on whether the user intends to perform a drum-hitting gesture or not. Thus, these features can be used to compensate for the delay introduced by the application. Moreover, by this way it is possible to predict when the user is going to perform a drum-hitting gesture according to a reference of previously performed gestures and the corresponding features extracted.

There are several ways in which a drum-hitting motion can be characterized. In the following lines, we present some possible features that can be used for the recognition of drum-hitting-like gestures in a system with meaningful delay. To illustrate these ideas, we will consider that the user performs drum-hitting gestures in a somewhat exaggerated way. Note that short moves are very hard to track with the 3D sensing technology considered.

- **Descending Acceleration**. A feature that is clearly linked to a downwards drum-hitting gesture is that of velocity and acceleration. More specifically, a negative acceleration peak along the Y axis will be expected to be found. At the start of the downwards movement, the acceleration over the Y axis should have a negative value; at the time when the drum would be actually hit, velocity should ideally become zero for that instant (so acceleration should have become positive instants before). Finally, since it would not be natural to end the movement exactly at the drum-hitting point, it stands to assume that in a proper drum-hitting gesture, the hand would ascend again after performing a beat. Thus, at the "end" of the gesture, the acceleration over the Y axis should become positive. This is a simple feature that can be used (along with a certain threshold and some constraints to the X, Y and Z coordinates to avoid false positives) to discern whether the user actually performs a drum-hitting motion or not. Furthermore, it is feasible to use this very same feature to predict whether the user intends to perform such a gesture: if the user makes a downwards movement with a sufficiently large value of speed/acceleration, it is very likely

that a drum-hitting gesture is to be performed. Thus, by setting a certain threshold on the measured acceleration/velocity of the user hand, it is possible to predict when the user intends to "play the drums". Therefore, the system can give a response before the actual motion is fully detected, helping to mitigate the effects of the delay of the tracking device (see Fig. 11).

- **Linear Prediction of Position**. Another way to address the problem of the delay is to use a linear predictor to estimate the evolution of user's motion according to the behaviour observed in the samples already captured. This would allow the system to actually predict the position of the user's hand and identify potential drum-hitting gestures prior to the actual completion of such gestures. Such identification would be accomplished, for example, by fitting the samples to a function that characterizes the gesture with sufficient accuracy. In this case, it is also important to account for the fact that different users might tend to perform their gestures at different velocities, so the information in the database should cover a wide enough assortment of gestures performed at different speeds or be properly parameterized to prevent this.

These features can be used by themselves to detect whether a drum-hitting gesture has been performed or not. Despite this, it is more likely to get better results in the gesture-recognition task when combining the outputs given by these features. This can be achieved by training a Bayesian classifier or by resorting to other machine learning methods.

It is important to notice that the data sampled should be appropriately normalized in order to minimize the dependence of the motion data tracked on the actual size of each user. Even for users similarly sized, the proximity to the 3D sensing device will also affect the actual range of values that will be tracked.

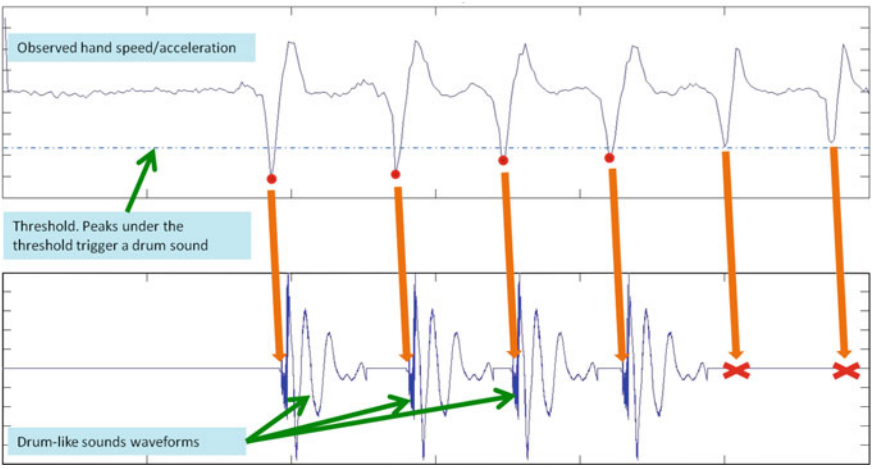

Fig. 11 Example of gesture detection using descending velocity (or acceleration): if there is a peak of downward speed faster than a given threshold, a drumbeat is played

One simple way to overcome this limitation is to use the torso-to-head distance as a reference to normalize the data tracked. This simple reference removes the dependence of the data sampled on the proximity to the sensor and mitigates the effects of user size variability in the size of potential users.

Another issue, especially with regard to avoiding false positive detection, is that certain constraints should be applied to gesture detection features for those cases in which it is obvious that no significative gestures can be performed. Examples of such constraints can refer to the mean velocity of the movement or the coordinates of the location of the hand at the beginning of the motion.

One final aspect to consider is that of a drum-kit simulation. We previously stated that, with the first approach discussed (virtual battery simulation in 3D space), it would be difficult to implement a drum-kit due to the precision necessary to "hit" the space confined by the invisible virtual drums. In this second approach, a problem is the delay introduced by the system in the detection of a gesture. Since the gesture recognition measures proposed do not depend on the actual position on the XZ plane, a simple way to discriminate between different drums would be to use the coordinates on the X and Z axes whenever a gesture is detected to select the corresponding drum sound to play. While this partially brings back some issues regarding precision when "hitting" an invisible 3D space, this solution is by far more robust in this case. First of all, discriminating between positions in the XZ plane is far simpler than detecting when the user "hits" a specific 3D volume. Secondly, and more important, since a sound is played when a gesture has been detected, as long as a gesture is correctly performed, a sound will play. Precision errors might result in having the wrong drum sound played, but the system would respond. On the contrary, with the first implementation, "missing the drum" implies that no sound would be played, which would be more frustrating to the user. Furthermore, with the second approach, if the user wants to virtually play drums, he/she will need to perform gestures that are reasonably similar to the actual gestures a musician would make when playing the drums.

Like the pitch-shifting application presented in the previous section, this application can also be seen from a pedagogical point of view. In particular, it can be used to teach rhythmic patterns in basic music stages. But perhaps the most enticing application of such a system would be as an accessible way for novice users to musical expression and performance. Albeit limited, the proposed application helps to lower barriers for lay people in musical terms, and allows for a reasonable simulation of a drum-like instrument. Also, it removes most of the hurdles related to instrument maintenance and overuse, as well as offering an explorative and intuitive gateway to the world of musical performance.

3.3 Other Interaction Paradigms

So far we have presented in detail some aspects regarding specific applications for real-time integration of human motion and musical responses. However, there are

many other enticing and appealing applications and interaction paradigms that allow for the integration of motion cues and music signal processing. In this section, some examples of interaction paradigms that can be used to achieve new forms of musical expression will be briefly presented.

3.3.1 The Virtual Director: An Example of Tempo Modification with Hand Motion

The title of music director is normally used in many symphony orchestras to designate the principal conductor and artistic leader of the orchestra. The role of the director is to oversee the overall musical performance, supervising and guiding the musical performance of the orchestra. The director guides the musicians with the motion of his hands and arms. Thus, a simple approximation to a virtual director application can be easily achieved with a body motion tracking system, as shown in Fig. 12.

More concretely, it is relatively easy to use skeletal joint data to discriminate over a given range of tempo. For example, to discriminate among *andante*, *adagio* and *presto*, the space around the user could be divided into three regions according to height: a lower region (for example, from slightly over the waist and downwards), an upper region (from the shoulders upwards), and a middle region (approximately between shoulders and waist). Therefore, if the user moves his/her hands in the lower region, tempo would be set to *adagio*, *andante* in the middle region, and *presto* for the upper region. Thus, by detecting that the user is performing a waving motion similar to the archetypical waving motion used by music directors, it is feasible to implement a basic version of a music director simulator. More complex implementations could be achieved by extending the range of tempo considered, or by analyzing the waving speed instead of or in addition to the height of the hands, as shown in Fig. 13.

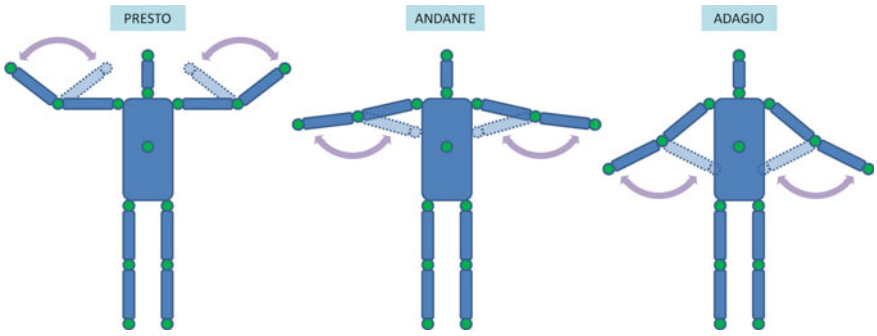

Fig. 12 Schematic representation for a simple implementation of a virtual director by means of a height-based system for tempo selection

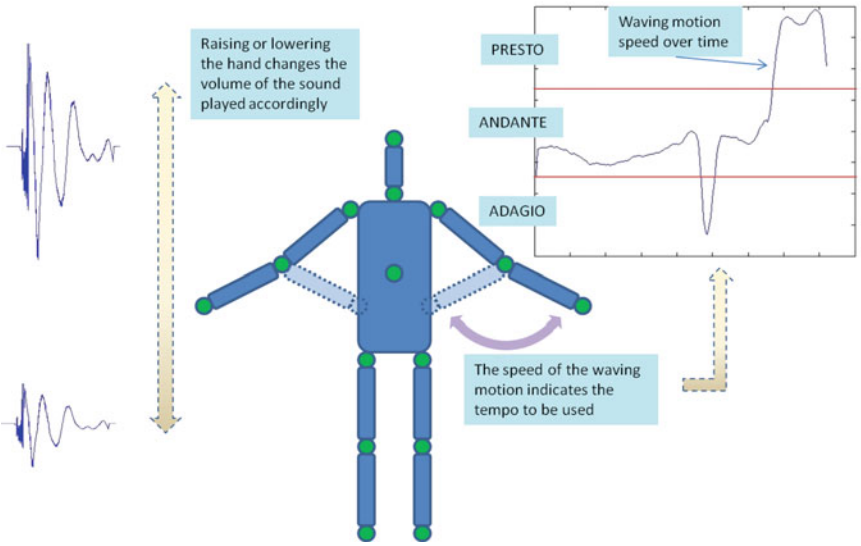

Fig. 13 An alternative interaction paradigm for a virtual director

3.3.2 Instrument Simulation

Previously, we have discussed the implementation of a simple drum-like instrument, but a similar approach can be followed for the implementation of other types of instruments. The xylophone and other percussion instruments could be easily simulated using a system similar to the one previously proposed. Obviously, the skeletal model obtained from *Kinect's* depth map would not be useful to implement an "air-piano". However, by processing the depth and RGB images simultaneously, it would be possible to find the position of each finger to implement such application. Of course, the computational cost would be larger than in the previous examples. Also, in this case a visual reference in the real world would be necessary since users cannot be expected to correctly hit the keys of an invisible piano accurately.

3.3.3 Advanced Pitch-Shifting/Correction

An evolution of the previously presented pitch-shifting application could combine the idea of pitch shifting by motion with an onset detector. In particular, this application could follow the next scheme: the user records a performance of a musical piece. Then, this sample is processed and segmented into the notes actually played. Afterwards, the application shows the timeline of the song divided by the note-segments detected (for example, using a pentagram, as shown in Fig. 14). Then, the user could modify the pitch of each identified note-segment

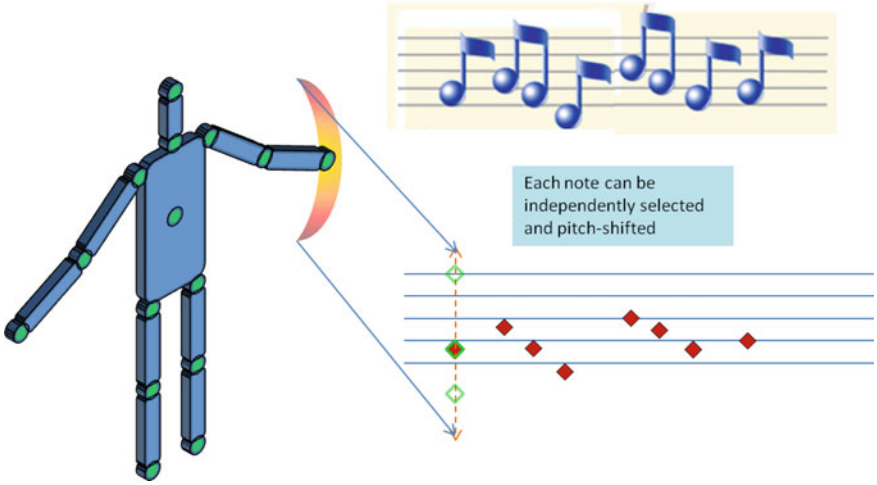

Fig. 14 Portrayal of an advanced pitch-shifter

along the pentagram, for example using the left hand to select the notes to be pitch-shifted, and the right hand to raise or lower the pitch. Again, this can be a useful application for pedagogical purposes, especially in teaching the basics of music theory.

3.3.4 Real-Time Motion-Based Composition

Motion-based interfaces can provide the means for real-time expressive rendering of musical pieces, lowering the barriers of music theory for a naïve user. This can be easily achieved, for example by allowing the user to create music by selecting notes and placing them on a score (in this case, the motion interface becomes mainly an additional compelling element). Also, the previously commented pitch-shifting application can be considered a real-time composition-by-motion scheme. A more interesting system could be based on combining motion detection with an automatic score generator.

4 Conclusions

In this chapter, a brief overview of the capabilities of advanced human computer interaction for the design and development of musical interactive applications has been drawn. More specifically this chapter has focused on the combination of human motion with music signal processing techniques in order to achieve an assortment of effects or applications. After summarizing the main advantages and

disadvantages of the different technological alternatives for the implementation of human motion-tracking systems, we have turned our attention to the Microsoft *Kinect 3D* sensor given its availability, low-cost, non-intrusiveness and high performance features.

We have shown details of two example applications that make use of a 3D sensor depth camera to implement a real-time motion-based application for pitch shifting and a drum simulator. However, as evidenced by the bibliography and the alternative examples shown in the previous sections, the interaction possibilities are nearly endless.

In view of the work developed, the main advantage of the applications created or described is their high level of accessibility, since no prior musical knowledge is required for the user to actually interact with the sounds played. As previously stated, this makes for an innovative and explorative way to better understand the concepts behind music, sound, or sound effects. Taking into account this observation, it is easy to get to the conclusion that this kind of applications can be potentially useful for music learning purposes, diminishing some of the hurdles concerning music theory by means of a more "tangible" visualization of theoretical concepts.

Furthermore it is expected that this type of added interactivity will also make the experience more compelling and fulfilling. This should encourage the application of advanced technologies and devices for music related tasks, including entertainment and learning.

Acknowledgments This work was supported by the Ministerio de Economía y Competitividad of the Spanish Government under Project TIN2010-21089-C03-02 and Project IPT-2011-0885-430000 and by the Ministerio de Industria, Energía y Turismo of the Spanish Government under project TSI-090100-2011-25.

References

Antle, A. N., Droumeva, M., & Corness, G. (2008). Playing with the sound maker: Do embodied metaphors help children learn? *Proceedings of the 7th International Conference on Interaction Design and Children (IDC '08)* (pp. 178–185).

AudioSurf. http://www.audio-surf.com/, release Feb 2008. Last accessed 14 June 2012.

Bakker, S., van den Hoven, E., & Antle A. N. (2011). MoSo tangibles: Evaluating embodied learning. *Proceedings of the fifth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '11)* (pp. 85–92).

Baratoff, G., & Blanksteen, S. (1993). Tracking devices. Technical Report, Human Interface Technology Laboratory. http://www.hitl.washington.edu/scivw/EVE/I.D.1.b.TrackingDevices.html. Last accessed 15 May 2012.

Barbancho, A. M., Barbancho, I., Tardon, L. J., & Urdiales C. (2009). Automatic edition of songs for guitar hero/frets on fire. *IEEE International Conference on Multimedia and Expo (ICME 2009)* (pp. 1186–1189). June–July 2009, New York.

Beatles Rock Band. The. http://www.thebeatlesrockband.com/, release Sept 2009. Last accessed 14 June 2012.

Castellano, G., Bresin, R., CamurriI, A., & Volpe, G. (2007). Expressive control of music and visual media by full-body movement. *Proceedings of the 7th International Conference on New Interfaces for Musical Expression (NIME'07)* (pp. 390–391).

Craft, K. (2009). Guitar hero III passes $1b. http://www.edge-online.com/news/activision-guitar-hero-iii-passes-1b. Last accessed 14 June 2012.

De Dreu, M. J., van der Wilk, A. S. D., Poppe, E., Kwakkel, G., & van Wegen, E. E. H. (2012). Rehabilitation, exercise therapy and music in patients with Parkinson's disease: A meta-analysis of the effects of music-based movement therapy on walking ability, balance and quality of life. *Parkinsonism and Related Disorders, 18*(1), 114–119.

FretsOnFire. http://fretsonfire.sourceforge.net/, debut release Aug 2006. Last accessed 14 June 2012.

Gleicher, M., & Ferrier, N. (2002). Evaluating video-based motion capture. *Proceedings of Computer Animation 2002 (CA'02)* (pp. 75–80).

Gower, L., & McDowall, J. (2012). Interactive music video games and children's musical development. *British Journal of Music Education, 29*, 91–105.

Grollmisch, S., Dittmar, C., & Gatzsche, G. (2009). Concept, implementation and evaluation of an improvisation based music video game. *IEEE Consumer Electronics Society's Games Innovation Conference* (pp. 210–212). London.

Guitar Hero. http://hub.guitarhero.com/, last release Feb 2011. Last accessed 14 June 2012.

Holland, S., Bouwer, A., Dalgleish, M., & Hurtig, T. (2010). Feeling the beat where it counts: Fostering multi-limb rhythm skills with the haptic drum kit. *Proceedings of the Fourth International Conference on Tangible, Embedded, and Embodied Interaction (TEI'10)* (pp. 21–28).

Jorda, S. (2010). The reactable: Tangible and tabletop music performance. *Proceedings of the 28th of the International Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA'10)* (pp. 2989–2994).

Karimi, R., Nanopoulos, A., & Schmidt-Thieme, L. (2012). RFID-enhanced museum for interactive experience. *Multimedia for Cultural Heritage, Communications in Computer and Information Science, 247*(3), 192–205.

Khoo, E. T., Merritt, T., Fei, V. L., Liu, W., Rahaman, H., Prasad, J., & Marsh, T. (2008). Body music: Physical exploration of music theory. *Proceedings of the 2008 ACM SIGGRAPH Symposium on Video Games (Sandbox '08)* (pp. 35–42).

Levin, G., & Lieberman, Z. (2004). In-situ speech visualization in real-time interactive installation and performance. *Proceedings of the 3rd International Symposium on Non-Photorealistic Animation and Rendering* (pp. 7–14).

Mancini, M., Bresin, R., & Pelachaud, C. (2007). A virtual head driven by music expressivity. *IEEE Transactions on Audio, Speech and Language Processing, 15*(6), 1833–1841.

Migneco, R., Doll, T., Scott, J., Hahn, H., Diefenbach, P., & Kim, Y. (2009). An audio processing library for game development in flash. *IEEE Consumer Electronics Society's Games Innovation Conference* (pp. 201–209). London.

Mikestar. http://www.mikestar.com/de/skore/start, debut release Oct 2009. Last accessed 14 June 2012.

Perry, L. D. S., Smith, C. M., & Yang, S. (1997). An investigation of current virtual reality interfaces. *ACM Crossroads Student Magazine, 3*(3), 23–28.

PrimeSense (2011a). OpenNI: User guide. Available: http://www.openni.org/Documentation.aspx. Last accessed 15 May 2012.

PrimeSense (2011b). Prime sensor NITE 1.3 algorithms notes. Available: pr.cs.cornell.edu/humanactivities/data/NITE.pdf. Last accessed 15 May 2012.

Rock Band. http://www.rockband.com/, debut Wii version release Sept 2008. Last accessed 14 June 2012.

Shivappa, S. T., Trivedi, M. M., & Rao, B. D. (2010). Audiovisual information fusion in human–computer interfaces and intelligent environments: A survey. *Proceedings of the IEEE, 98*(10), 1692–1715.

SingStar. http://www.singstargame.com/, debut release May 2004. Last accessed 14 June 2012.

Smisek, J. (2011). 3D with kinect. *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)* (pp. 1154–1160).

Taylor, R., Torres, D., & Boulanger, P. (2005). Using music to interact with a virtual character. *International Conference on New Interfaces for Musical Expression* (pp. 220–223).

UltraStar. http://www.ultrastar.pl/about, release Oct 2010. Last accessed 14 June 2012.

Villaroman, N., Rowe, D., & Swan, B. (2011). Teaching natural user interaction using OpenNI and the Microsoft Kinect sensor. *Proceedings of the 2011 Conference on Information Technology Education (SIGITE '11)*. ACM (pp. 227–232).

Wang, C. Y., & Lai, A. F. (2011). Development of a mobile rhythm learning system based on digital game-based learning companion. *Lecture Notes in Computer Science, 6872*, 92–100.

Welch, G., & Foxlin, E. (2002). Motion tracking: No silver bullet, but a respectable arsenal. *IEEE Computer Graphics and Applications, 22*(6), 24–38.

Zölzer, U. (Ed.). (2002). *DAFX digital audio effects*. Wiley: New York.