HEAD IN SPACE: A HEAD-TRACKING BASED BINAURAL SPATIALIZATION SYSTEM

Luca A. Ludovico, Davide A. Mauro, and Dario Pizzamiglio
LIM - Laboratorio di Informatica Musicale
Dipartimento di Informatica e comunicazione (DICo)
Universitá degli Studi di Milano, Via Comelico 39/41, I-20135 Milan, Italy
{ludovico,mauro}@dico.unimi.it
http://www.lim.dico.unimi.it

ABSTRACT

This paper discusses a system capable of detecting the position of the listener through a head-tracking system and rendering a 3D audio environment by binaural spatialization. Head tracking is performed through face recognition algorithms which use a standard webcam, and the result is presented over headphones, like in other typical binaural applications. With this system users can choose an audio file to play, provide a virtual position for the source in an euclidean space, and then listen to the sound as if it is coming from that position. If they move their head, the signal provided by the system changes accordingly in real-time, thus providing a realistic effect.

1. INTRODUCTION

3D sound is becoming a prominent part of entertainment applications. The degree of involvement reached by movies and video-games is also due to realistic sound effects, which can be considered a virtual simulation of a real sound environment.

In one of the definitions of Virtual Reality, simulation does not involve only a virtual environment but also an immersive experience (see [1]); according to another author, instead of perception based on reality, Virtual Reality is an alternate reality based on perception (see [2]). An immersive experience takes advantage from environments that realistically reproduce the worlds to be simulated.

In our work, we are mainly interested in audio aspects. Even limiting our goals to a realistic reproduction of a single audio source for a single listener, the problem of recreating an immersive experience is not trivial. With a standard headphones system, sound seems to have its origin inside the listener's head. This problem is solved by binaural spatialization, described in Section 3, which gives a realistic 3D perception of a sound source S located somewhere around the listener L. Nowadays, most projects using binaural spatialization aim at animating S keeping the position of L fixed. Thanks to well known techniques, such a result is quite easy to achieve. However, for an immersive experience this is not sufficient: it is necessary to know the position and the orientation of the listener within the virtual space in order to provide a consistent signal [3], so that sound sources can remain fixed in virtual space independently of head movement, as they are in natural hearing [4].

As a consequence, we will introduce a head-tracking system to detect the position of L within the space and modify the signal delivered through headphones accordingly. The system can now verify the position of S with respect to L and respond to his/her movements.

At the moment, audio systems typically employ magnetic head trackers thanks both to their capability of handling a complete 360° rotation and to their good performances. Unluckily, due to the necessity of complex dedicated hardware, those systems are suitable only to experimental or research applications. But the increasing power of home computers is supporting a new generation of optical head trackers, based primarily on webcams.

This work proposes a low cost spatialization system which only relies on resources available to most personal computers. Our solution, developed with MAX/MSP, is based on a webcam head-tracking system and binaural spatialization implemented via convolution.

The paper is structured as follows. First we will provide a short review of related literature and similar systems. Then the basic concepts about binaural spatialization techniques will be introduced. Finally we will describe the integration of a head-tracking system via MAX/MSP externals - namely the multi-platform, real-time programming environment for graphical, audio, and video processing used to implement our approach - and the real-time algorithms involved in the processing of audio and video streams.

2. RELATED WORKS

We want to present here other similar approaches and projects which served as a basis in the development process. Some concepts, such as "binaural spatialization" will be introduced in the following.

- Binaural Tools: A MAX/MSP patch from the author of CIPIC database that performs binaural panning using Head Related Transfer Function (HRTF)

measurements. The panner takes an input sound file and convolves it with a measured sound response recorded from a selectable angle and elevation. Output can optionally be recorded to a sound file. The program was created based on some parts of Vincent Choqueuse's binaural spatializer for Max/MSP [5]. We started from these works to develop our approach. They are inspiring as they do not use external libraries and rely solely on MAX capabilities. This approach has also some drawbacks. For example, in order to perform spatialization efficiently, other techniques could be used but they should be expressly implemented.

- Spat∼: A Spatial Processor for Musicians and Sound Engineers [6]. Spat∼ is a real-time spatial processing software which runs on the Ircam Music Workstation in the MAX graphical signal processing environment. It provides a library of elementary modules (pan-pots, equalizers, reverberators, etc.) linkable into a compact processor integrating the localization of sound events together with the manipulation of room acoustical quality. This processor can be configured for various reproduction formats over loudspeakers or headphones, and controlled through a higher-level user interface including perceptual attributes derived from psychoacoustical research. Applications include studio recording and computer music, virtual reality or variable acoustics in rooms. The stability and quality of this library could be useful to redesign some structures of our spatializer and achieve better quality and performances.

- bin_ambi: A Real-Time Rendering Engine for Virtual (Binaural) Sound Reproduction [7]. This library is intended for the use with Miller Puckette's open source computer music software Pure Data (PD). The library is freely downloadable and can be used under the terms of GNU General Public License. It provides a simple API easy to use for scientific as well as for artistic projects. In this implementation there is a room simulation with 2 sound objects and a listener. One direct signal and 24 early reflections are calculated and rendered per sound object. The sound rendering based on mirror sources provides models for the early reflections. Each reflection will be encoded into the Ambisonics domain (4th order 3-D) and added to the Ambisonics bus. The listener rotates the whole Ambisonics field, the Ambisonics decoder renders the field into 32 discrete signals of 32 virtual loudspeakers. All 32 speaker signals will be filtered by its HRFT in relation to the left and to the right ear (binaural decoding). Interpolation is one of the critical points of such applications. We can choose an approach like the one proposed here that could give a better interpolation and sound quality but increases the computational complexity of the system.

- 3D-Panner [8]: A SuperCollider-based spatialization tool for creative musical applications. The program spatializes monaural sounds through HRTF convolution, allowing the user to create 3D paths in which the sound source will travel. In 3D Panner the user can easily create unique paths that can range from very simple to very complex. These paths can be saved independently of the sound file itself and applied to any other monaural source. During playback, the sound source is convolved with the interpolated HRTFs in real-time to follow the user-defined spatial trajectory. This project is inspiring for our work because we plan to introduce new features, such as moving sound sources, and we need a way to describe and handle trajectories.

3. BINAURAL SPATIALIZATION

Binaural spatialization is a technique that aims at reproducing a real sound environment using only two channels (like a stereo recording). It is based on the assumption that our auditory system has only two receivers, namely the ears. If it is possible to deliver a signal equal (or nearly equal) to the one which a subject would receive in a real environment, this will lead to the same perception. Our auditory system performs various tasks to obtain a representation of the acoustic environment; most of them are based on the physical parameters of the signal of interest and are called "cues" [9][10].

Binaural spatialization can be achieved through various processes, such as: equalizations and delays, or convolution with the impulse response of the head (HRIR). The latter approach is the one we have followed in our work. In order to obtain these impulses, many experiments involving the use of a dummy head [1] have been made (see i.e. [11]), thus creating databases of impulse responses. Most of them use a fixed distance (usually 1 meter) from S to L, which constitutes a potential limitation.

4. INTEGRATING A HEAD-TRACKING SYSTEM INTO MAX

In our work, we choose to adopt faceAPI, namely an optical face tracking system developed by Seeing Machines [12] that provides a suite of functions for image processing and face recognition encapsulated in a tracking engine. It is a commercial product - freely usable only for research purposes - that implements a head tracker with six degrees of freedom. It can be seen as a "black box" which grants access to tracking data through a simple interface oriented to programming tasks. Basically the engine receives frames from a webcam, processes them and then returns information about the position of the head with respect to the camera.

MAX/MSP is an integrated platform designed for multimedia, and specifically for musical applications [13]. This graphical real-time environment can be successfully used by programmers, live performers, "traditional" musicians, and composers. Within the program objects are also represented like "black boxes" which accept input through their

[1] A dummy head is a mannequin that reproduces the human head.

inlets and return output data through their *outlets*. Programs are built by disposing these entities on a canvas (the *patch*) and creating a data flow by linking them together through *patchcords*.

MAX provides developers with a collection of APIs to create external objects and extend its own standard library [14]. The integration of the head tracker requires to create a base project for MAX (we used the so called "minimum project") and then add references to faceAPI to start developing the external.

When MAX loads an external, it calls its *main()* function which provides initialization features. Once loaded, the object needs to be instanced by placing it inside a patch. Then the external allocates memory, defines inlets and outlets and configures the webcam. Finally, faceAPI engine starts sending data about the position of the head. In our implementation the external reacts only to *bang* messages: [2] as soon as a message is generated, a function of faceAPI is invoked to return the position of the head through *float* variables.

Each MAX object has to be defined in terms of a C structure, i.e. a structured type which aggregates a fixed set of labelled objects, possibly of different types, into a single object. Our implementation presents only pointers to the object outlets in order to directly pass variables to the tracking engine.

```
typedef struct _head {
   t_object c_box;
   void *tx_outlet, *ty_outlet, *tz_outlet;
   void *rx_outlet, *ry_outlet, *rz_outlet;
   void *c_outlet;
} t_head;
```

Such values represent the translation along 3 axes (tx, ty, tz), the orientation of the head in radians (rx, ry, rz) and a confidence value. After their detection, values are sent to their corresponding *outlets* and they are available to the MAX environment. In brief, the headtracker external presents only one inlet that receives bang messages and seven outlets that represent the values computed by the tracking engine.

5. THE "HEAD IN SPACE" APPLICATION

This section aims at introducing the Head in Space (HiS) application for MAX. As discussed in Section 4, we assume that our head-tracking external acts as a black box that returns a set of parameters regarding the position of the head.

In Figure 1 a workflow diagram of the system is shown.

In input, two sets of parameters are available to the system, in order to define: 1. the position of the listener, and 2. the position of the audio source. Given this information, and taking into account also the position of the camera, it is possible to calculate the relative position of the listener with respect to the source in terms of azimuth, elevation and distance. This is what the system needs to choose which impulse response to use for spatialization. Once the correct HRIR is obtained from the database, it is possible to perform convolution between a mono audio signal in input and the stereo impulse response. Since the position both of the listener and of the source can change over time, an interpolation mechanism to switch between two different HRIRs has been implemented.

[2] A bang is a MAX special message that causes other objects to trigger their output.

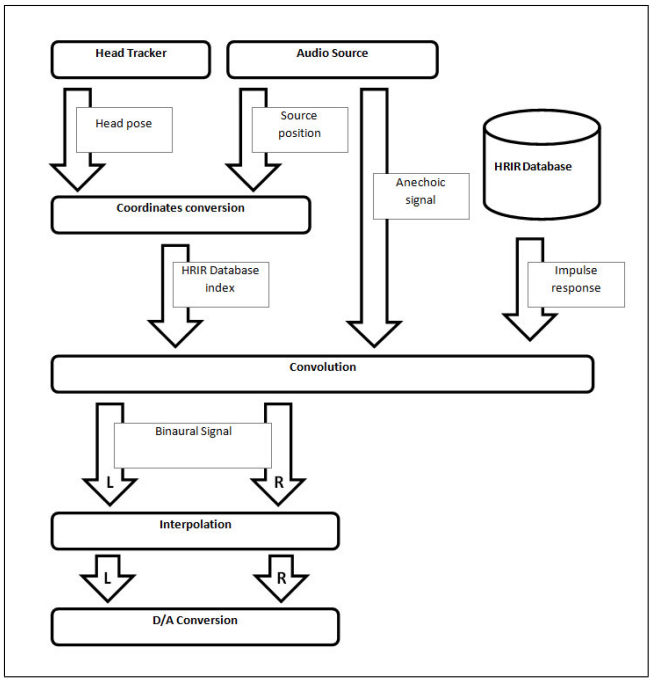

Figure 1. The workflow diagram of the system.

5.1 Coordinates Extraction

The spatializer uses a spherical-coordinates system that has its origin in the center of the listener's head. Source is identified by a distance measure and two angles, namely azimuth on horizontal plane and elevation on median plane. Angular distances are expressed in degrees and stored in the patch through integer variables, whereas the distance is expressed in meters and its value is stored as a float number.

Please note that the head tracker presents coordinates in a cartesian form that has its origin in projection cone of the camera. Thus the representation of coordinates of the spatializer and the one of the head tracker are different and a conversion procedure is needed. The conversion process first performs a rototranslation of the system in order to provide the new coordinates of translation both of the source and of the head inside a rectangular reference system.

Referring to Figure 3, given the coordinates for a generic point P, representing the source in a system $(O_1; X_1, Y_1, Z_1)$, we can determine a set of coordinates in a new cartesian plane $(O_2; X_2, Y_2, Z_2)$ that refers to the position of the head through the relation:

$$V_2 = V_0 + (1 + k) \cdot R \cdot V_1 \qquad (1)$$

where:

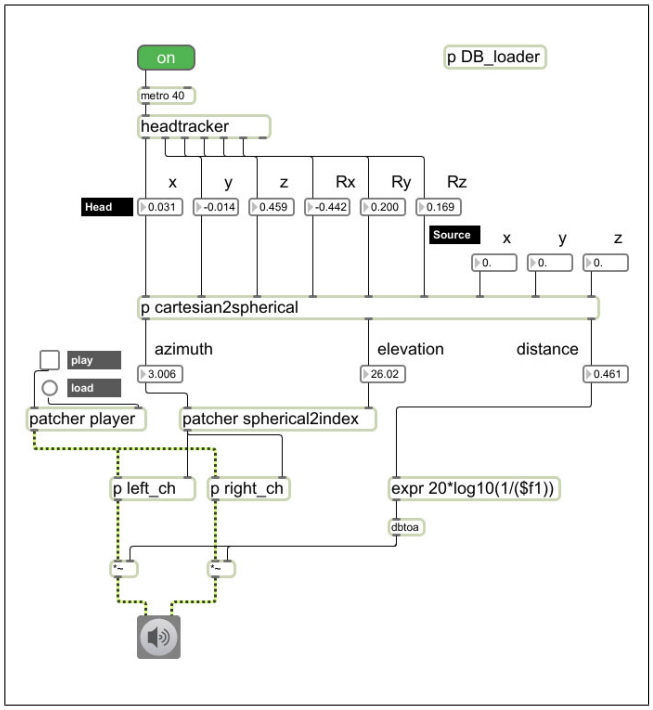

Figure 2. An overview of the patch.

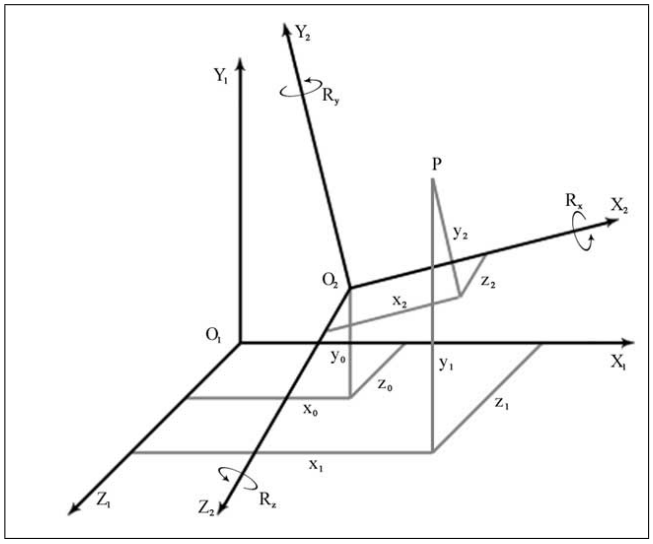

Figure 3. The translation system.

$$V_0 = \begin{Vmatrix} x_0 \\ y_0 \\ z_0 \end{Vmatrix} \quad \text{translation components}$$

$$V_1 = \begin{Vmatrix} x_1 \\ y_1 \\ z_1 \end{Vmatrix} \quad \text{known coordinates of } P \text{ in } O_1$$

$$V_2 = \begin{Vmatrix} x_2 \\ y_2 \\ z_2 \end{Vmatrix} \quad \text{unknown coordinates of } P \text{ in } O_2$$

$$k = 0 \quad \text{scale factor}$$

$$R = R_x \cdot R_y \cdot R_z \quad \text{rotation matrix} \tag{2}$$

R is the matrix obtained by rotating each cartesian triplet with subscript 1 along its axes X_1, Y_1, Z_1 with rotation of R_x, R_y, R_z to displace it parallel to X_2, Y_2, Z_2. Rotation matrixes are:

$$R_x = \begin{Vmatrix} 1 & 0 & 0 \\ 0 & \cos(R_x) & \sin(R_x) \\ 0 & -\sin(R_x) & \cos(R_x) \end{Vmatrix} \tag{3a}$$

$$R_y = \begin{Vmatrix} \cos(R_y) & 0 & -\sin(R_y) \\ 0 & 1 & 0 \\ \sin(R_y) & 0 & \cos(R_y) \end{Vmatrix} \tag{3b}$$

$$R_z = \begin{Vmatrix} \cos(R_z) & \sin(R_z) & 0 \\ -\sin(R_z) & \cos(R_z) & 0 \\ 0 & 0 & 1 \end{Vmatrix} \tag{3c}$$

the product $R_x \cdot R_y \cdot R_z$ is calculated with (4).

We can now derive formulas to calculate the position in the new system:

$$\begin{aligned} x_2 =& (x_0 + x_1)[\cos(R_y)\cos(R_z)] \\ &+ (y_0 + y_1)[\cos(R_x)\sin(R_z) \\ &+ \sin(R_x)\sin(R_z)\cos(R_z)] \\ &+ (z_0 + z_1)[\sin(R_x)\sin(R_z) \\ &- \cos(R_x)\sin(R_z)\sin(R_z)] \end{aligned} \tag{5}$$

$$\begin{aligned} y_2 =& (x_0 + x_1)[\cos(R_y)\sin(R_z)] \\ &+ \\ &(y_0 + y_1)[\cos(R_x)\cos(R_z) \\ &- \sin(R_x)\sin(R_z)\sin(R_z)] \\ &+ (z_0 + z_1)[\sin(R_x)\cos(R_z) \\ &+ \cos(R_x)\sin(R_z)\sin(R_z)] \end{aligned} \tag{6}$$

$$\begin{aligned} z_2 =& (x_0 + x_1)\sin(R_y) \\ &+ (y_0 + y_1)[\sin(R_x)\cos(R_y)] \\ &+ (z_0 + z_1)[\cos(R_x)\cos(R_y)] \end{aligned} \tag{7}$$

Now we can calculate spherical coordinates using the following formulas:

$$distance\ \rho = \sqrt{x^2 + y^2 + z^2} \tag{8}$$

$$azimuth\ \varphi = \arctan\left(\frac{z}{x}\right) \tag{9}$$

$$elevation\ \theta = \left(\frac{y}{\sqrt{x^2 + y^2 + z^2}}\right) \tag{10}$$

$$R = \begin{Vmatrix} \cos(R_y)\cos(R_z) & \cos(R_x)\sin(R_z) + \sin(R_x)\sin(R_y)\cos(R_z) & \sin(R_x)\sin(R_z) - \cos(R_x)\sin(R_y)\sin(R_z) \\ -\cos(R_y)\sin(R_z) & \cos(R_x)\cos(R_z) - \sin(R_x)\sin(R_y)\sin(R_z) & \sin(R_x)\cos(R_z) + \cos(R_x)\sin(R_y)\sin(R_z) \\ \sin(R_y) & -\sin(R_x)\cos(R_y) & \cos(R_x)\cos(R_y) \end{Vmatrix} \quad (4)$$

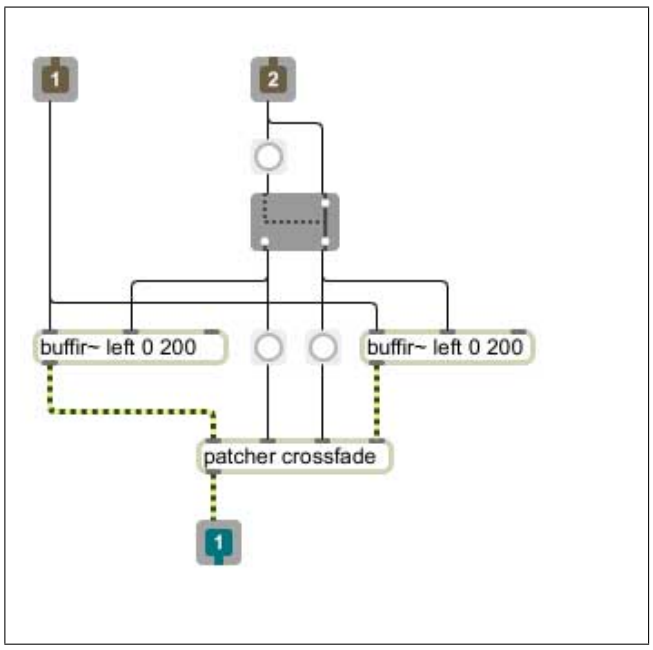

Figure 4. The detail of the MAX subpatch for the convolution process.

The new set of coordinates can be employed to retrieve the right HRIR from the database. Since our database includes only HRIRs measured at a given distance, we only use azimuth and elevation. How to use the distance value to simulate the perception of distance will be explained in Section 5.4. Since not all the possible pairs of azimuth and elevation have a corresponding measured HRIR within the database, we choose the database candidate that minimizes the euclidean distance.

5.2 The Convolution Process

This section describes the convolution process between an anechoic signal and a binaural HRIR. We use the CIPIC database [11], consisting of a set of responses measured for 45 subjects at 25 different values for azimuth and 50 different values for elevation. Each impulse consists of 200 samples.

Figure 4 illustrates the detail of the subpatch for one channel. From its first inlet it receives the anechoic signal, while from the second it gets the index for HRIR within a *buffer*$\sim$ object. HRIRs are stored in a single file that concatenates all the impulses. Please note that the process is performed one time for left channel and another one for right channel. Inside the database, azimuth and elevation values are numbered through an ad hoc mapping. Given an azimuth position naz and an elevation position nel we can calculate the starting point within the buffer with the

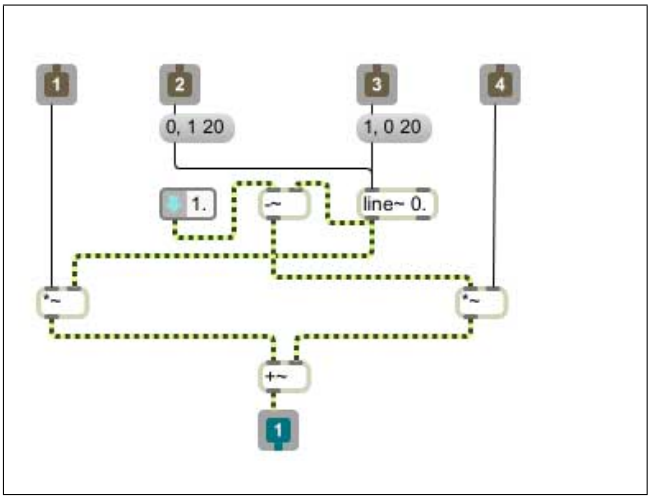

Figure 5. The detail of the MAX subpatch for the crossfade system.

formula:

$$[((naz - 1) \cdot 50) + (nel - 1)] \cdot ir_{length} \quad (11)$$

A *buffir*$\sim$ object is a finite impulse response (FIR) filter that loads both coefficients from the buffer and audio signal, and then performs convolution in time domain. Convolution is implemented through a FIR filter since the small number of samples of HRIRs makes computationally convenient to perform it in time domain instead of frequency domain. *buffir*$\sim$ object allows to store up to 256 coefficients.

5.3 Interpolation and Crossfade Among Samples

One of the known problems related to the use of HRIR for spatialization is the interpolation between two signals convolved with two different impulses. This is a very common case for this kind of real-time applications because when moving from one azimuth value to another impulses are very dissimilar. As a consequence, output signal can change abruptly, thus affecting negatively the perceived quality of the system. We have designed a simple yet performing interpolation procedure based on crossfade to limit the artifacts produced by the switch between impulses.

The approach is replicating the audio stream for each channel that lead to changes to the convolution subpatch. We add a second *buffir*$\sim$ object so now the first filter will produce signals convolved with the current impulse and the second filter will be loaded with the new HRIR provided by the new position. Then new signal will gradually overcome the signal from other filter with a crossfade function. Once done the role of the two filter will switch. This behaviour is achieved trough a *ggate*$\sim$ object.

As a performance issue it should be noted that in a real time environment every redundant operation should be avoided. In our implementation this means that a crossfade between samples is needed only if a switch has been detected by a change object that gives a value in output only if it is not equal to its previous value. This avoid unnecessary computation by the CPU that are useless if applied to the same impulse response and could lead to a degradation in terms of quality. Another improvement is given by the use of *speedlim*$\sim$ object that establish the frequency of messages in terms of minimum number of milliseconds between each consecutive message. It could happen that changing azimuth and elevation at the same time two different new messages could be generated in a rapid sequence. That could lead to a premature refresh in the filter coefficients leading to a loss of quality. With this component they are spaced by at least 40 msec. This value is chosen according with the typical refresh rate of a video stream (25 fps). This value is also used to define the crossfade duration between samples, and in our implementation the crossfade is linear. The user can define a value between 5 msec and 20 msec. By experiments, depending on the CPU power, it is possible to achieve a good quality even at 5 msec. So the overall delay between changes is $20\ msec + \frac{200_{samples}}{44100\frac{samples}{sec}}$.

5.4 Simulation of Distance

One of the limitations of the CIPIC database is presenting candidates only at one given distance. In order to simulate the distance effect, our patch contains a simple procedure based on the inverse square law. The function is implemented by an *expr*$\sim$ object [3] with the expression:

$$20\ log_{10}\left(\frac{1}{distance}\right)\ dB \qquad (12)$$

We limit the range of the distance value produced by the head-tracking system between 0.1 and 2. Conventionally 1 identifies the reference distance of the impulse response, and in this case no gain is applied. The mentioned distance value is employed to feed the gain of each channel. The process could be enhanced by adding a filter which simulates the air absorption or using a database where HRIRs are measured at various distances.

5.5 The Graphical User Interface

The software application that implements the algorithms described before is a standard patch for MAX/MSP. The patch uses an ad hoc external to implement the head-tracking function.

After launching it, the software presents a main window made of a number of panels and a floating window containing the image coming from the webcam after faceAPI processing. In the latter window, when a face is recognized, a wireframe contour is superimposed over the face image.

In Figure 6 we present the user interface of the application. As regards the main window, it is organized in several

[3] An *expr*$\sim$ object evaluates C-like expressions.

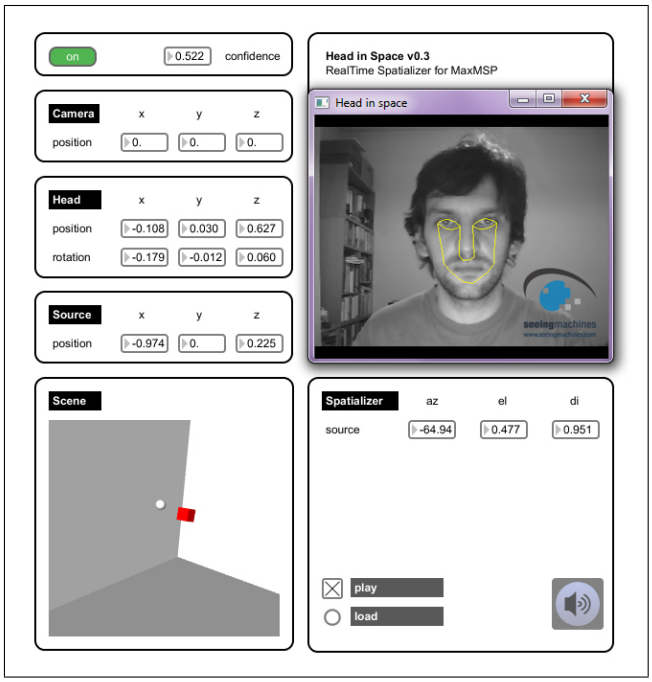

Figure 6. The graphical user interface of the program.

panels. First, it allows to switch on and off the processing engine. Besides, a number of text boxes and buttons are used to set the position of the camera and of the source. Other controls give feedback about the derived position of the listener and the corresponding translation into azimuth, elevation, and distance. A 3D representation (with the use of the OpenGL support of Jitter) of the system made of the listener (dark cube) and the source (white sphere) is also provided and updated in real time.

The bottom right panel contains the controls to choose the audio file to be played and to start the playback.

6. CONCLUSIONS & FUTURE WORKS

This paper has described a working application that performs real-time spatialization of an audio signal based on the position of the listener.

The system can be improved in several manners. The use of a single webcam corresponds to a limited resolution of azimuths and elevations ($\pm$ 90 azimuth, -30/+60 elevation, data coming from faceAPI specifications). It could be possible to combine more cameras in order to fully represent the space choosing the one with the highest confidence value.

Another improvement is adding support for more than one source in order to render a richer environment. It could also be interesting to take into account moving sound sources; this implies that a way to describe trajectories needs to be implemented.

The use of CIPIC database limits the number of possible measured distances and led us to implement a distance simulation mechanism, whereas it would be desirable to switch among HRIRs measured at various distances. Also the 200-samples HRIRs do not account for rooms ambience, so a reverberation tool is needed.

Since the application is structured in modules, it can be easily extended in order to support the future changes we have mentioned.

The source code and application are freely available from the authors at:
`http://www.lim.dico.unimi.it/HiS`.

7. REFERENCES

[1] J. Steuer, "Defining virtual reality: Dimensions determining telepresence," *Journal of Communication*, vol. 42, pp. 73–93, 1992.

[2] K. Osberg, *But what's behind door number 4? Ethics and virtual reality: A discussion*. Human Interface Technology Lab Technical Report R-97-16, 1997.

[3] S. P. Parker, G. Eberle, R. L. Martin, and K. I. McAnally, "Construction of 3-d audio systems: Background, research and general requirements.," tech. rep., Victoria: Defence Science and Technology Organisation, 2000.

[4] D. R. Begault, *3-D sound for virtual reality and multimedia*. Cambridge, MA: Academic press Professional, 1994.

[5] V. Choqueuse, "Binaural spatializer (for maxmsp)," *http://vincent.choqueuse.free.fr/*, 2007.

[6] J. Jot and O. Warusfel, "Spat˜: A spatial processor for musicians and sound engineers," in *CIARM: International Conference on Acoustics and Musical Research*, 1995.

[7] M. Noisternig, T. Musil, A. Sontacchi, and R. Hoeldrich, "3d binaural sound reproduction using a virtual ambisonics approach," in *VECIMS - International Symposion on Virtual Environments, Human-Computer Interfaces and Measurement Systems*, (Lugano, Switzerland), 2003.

[8] T. Tatli, "3d panner: A compositional tool for binaural sound synthesis," *International Computer Music Conference*, 2009.

[9] W. A. Yost, *Foundamentals of hearing: An introduction*. Academic press London, third ed., 1994.

[10] J. Blauert, *Spatial Hearing: The Psychophysics of Human Sound Localization*. Cambridge, MA: MIT Press, revised ed., 1996.

[11] V. R. Algazi, R. O. Duda, D. M. Thompson, and C. Avedano, "The cipic hrtf database," *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. W2001–1/W2001–4, October 2001.

[12] "Seeing machines face tracking api documentation," *http://www.seeingmachines.com/product/faceapi/*, 2009.

[13] A. Cipriani and M. Giri, *Musica Elettronica e Sound Design*, vol. 1. ConTempoNet, 2009.

[14] D. Zicarelli, J. Clayton, and R. Sussman, *Writing External Objects for Max and MSP 4.3*. 2001.