AUDIO POINTER V2 – JOYSTICK & CAMERA IN MATLAB

F. Rund

Department of Radioelectronics, Faculty of Electrical Engineering,
Czech Technical University in Prague, Czech Republic

Abstract

A virtual sound source can be positioned in a virtual auditory space with various methods, as shown in our other articles. In our last year paper, we shown an application of virtual sound source positioning for assistive technology: for helping blind people to orientate in real space. In the case, an operator can watch video simulating vision of sighted people and create a virtual sound source which navigates direction of user's walk. An appropriate positioning device (e.g. joystick) has to be used for determining the direction. The paper is devoted to improved version of last year simple implementation of the above mentioned system in Matlab. The main improvement is, that not only the azimuth, but also the elevation, can be adjusted by the operator. Also the operator can use more controls on joystick (volume, auxiliary signals) and more information is shown on his display. The Matlab-code from last year version was rewritten and optimized. This implementation is used for design of the system and testing the algorithms for virtual sound source positioning.

1 Introduction

Virtual sound source positioning methods can be used for various applications, e.g. in computer games. This paper describes a new version of a Matlab implementation of a system, positioning a virtual sound source to a position determined by using of joystick. The first implementation was presented last year [1], and intended for testing purposes of the PERSEUS (personal help for blind users) system [2]. The new version is used for the same purpose, photo of an experimental setup see on Fig. 1.

The PERSEUS system [2] consists of two parts - user's and operator's. The blind person (user) is equipped with stereoscopic camera, headphones and wearable PC which can communicate with the operator. In case of emergency, operator can see context of user's environment in 3D, so can estimate distance of obstacles. Thus he can navigate the user by means of acoustic cues, sent into user's headphones. An appropriate positioning device, e.g. joystick, is used for determining the direction.

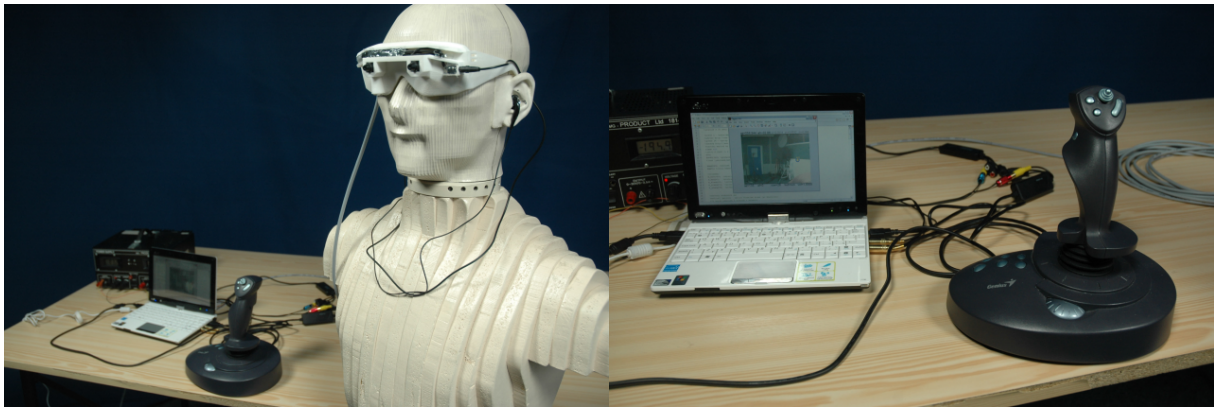

Figure 1: Experimental setup with (left) and detail of the computer with the script in use (right)

Matlab implementation, described here, is devoted to the problem of creating and presenting sound coming from the direction which corresponds to position of the joystick. An experimental setup (Fig. 1) was prepared, for testing of the main ideas. The setup and the script will be described in next sections. This implementation is aimed on the audio part of the project, so, in this version is implemented the 2D video (one camera) only.

2 The system

The main idea of the system is the same as in the first version [1]. The described system deals with two main problems. The first problem is to create virtual sound source in desired direction. The second problem comprises of reading of position of joystick and sending the appropriate sound to user's headphones.

Solving of the problems is divided into two scripts, described below. The actual version adds some improvements to the first version [1]. The operator's interface for the first version (left) and the actual version (right) is in Fig. 2.

2.1 Sound generating script

The main approach for the virtual sound source creating is the same as in the first version [1]. So the HRTF (Head Related Transfer Function) is convolved with an mono sound with good localization properties. In the actual version is the HRTF from databases such as [3] used, but, also the measured HRTF data set can be used.

For the second script (sound positioning, next subsection), only the format of the matrix of virtual sources is defined, so, the matrix can be created with another method (for example, only simple amplitude-panning).

The main difference from the first version of the script involves the possibility to 3D data generation. In the first version [1] was only one matrix generated, only for the elevation of zero degrees. In the actual version, the $\pm$ 45 degrees elevation matrices are generated in addition (in the case the corresponding parameter is set). In 3D case are in the output `.mat` file saved more set of matrices (3 in the actual version) – one set for each elevation stage.

In addition, the script was rewritten and some improvements were introduced, for example, the name of the output file (`.mat`) is generated automatically and comprises information about the stimulus type, the HRTF set, etc.

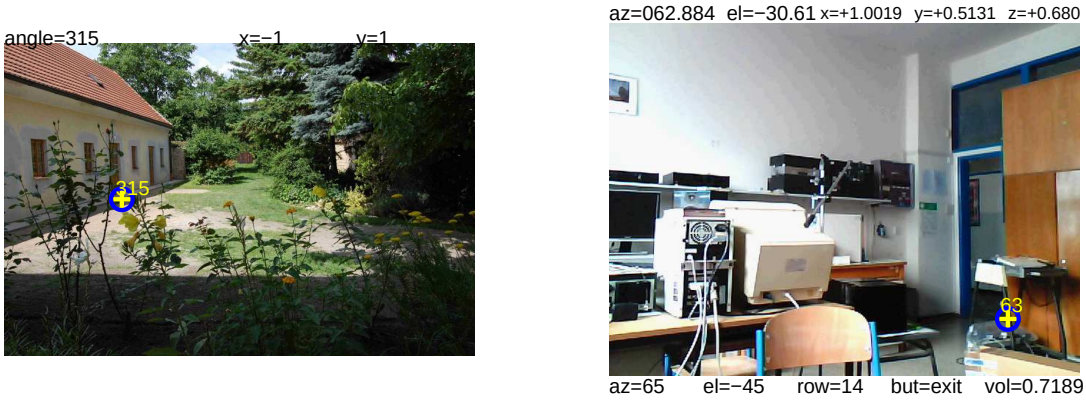

Figure 2: Examples of the operator's interface. Last year version (left) and the new version (right).

2.2 Sound positioning script

The main differences between the first [1] and the actual version are in the sound positioning script. Like in the first version [1], at first the sound data matrix is loaded, then the script reads position of the joystick, finds sound for corresponding direction and play it through user's headphones. The operator sees picture from user's camera and can point somewhere with joystick, as seen in Fig. 3. The sound data matrix (or matrices, in 3D case) can be generated by the script described in previous subsection, but, it can be created using another method, so, the sound positioning script is independent on the sound generating script, only the structure of the data is defined.

In the actual version of the script, more buttons on the joystick than in the first version [1] is enabled. The sound will be played to the user only if the appropriate button is pressed. The position (azimuth & elevation), and also level, of the virtual sound source presented to the user is fully controlled by the joystick. The operator can also send some *alarm signal* to the user's ears - for example, to stop the user, or warning him, etc. There are also many improvements in the operator's interface, described below, as seen from Fig. 2–4.

At beginning of the script the basic setup of the script is defined. The user can choose, if a static image (for testing purposes), or the image from a camera will be used. In the case of a camera usage, is also necessary to select which of cameras connected in computer have to be used (new in actual version). Then is defined the function of joystick buttons – for adopting the specific joystick to the an operator (there are differences between joysticks). Function of the following buttons can be defined:

- the *exit button* – for exit the script
- the *play button* – the sound is played only when the button is pressed – one of important changes from the first version – it enables near real-time function of the script.
- *alarm buttons* – 3 alarms are defined - when one of the buttons is pressed, appropriate auxiliary sound is played – it can be use as an alarm.

The second step is the initialization. Then the `.mat` file with the set of sounds (created with the script like this described in previous subsection) is loaded. The alarm sound are created (for this version was used simple pure tones, 990, 440, 1 555 Hz, but any of signal can be used).

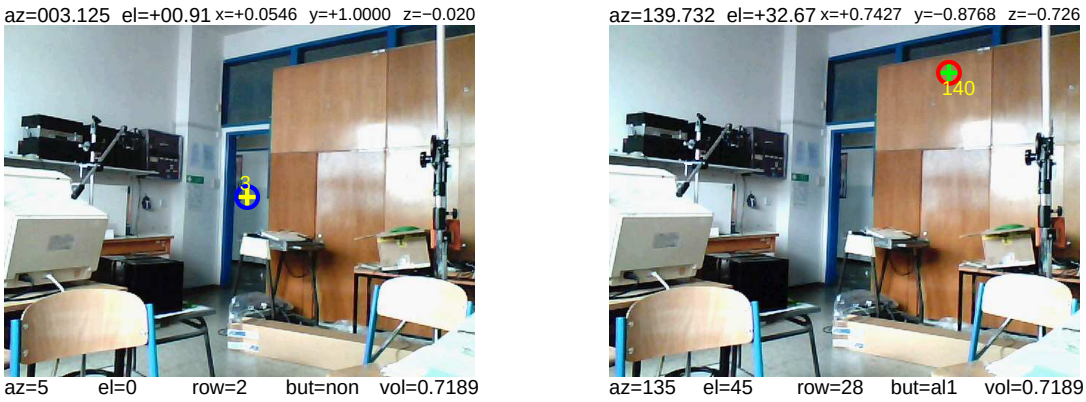

Figure 3: Examples of the operator's interface. Position of last played sound is similar to the actual position of the cursor, the level in the user's headphones is set to approx. 70 %. Left – the cursor is in front of the user, horizontal plane, no button is actually pressed. Right – the cursor is in the back of the user, upper plane, the alarm 1 sound is played.

In the case of camera usage is the proper camera initialized, otherwise the static image (from the disk file) is loaded.

Then begins the main part of the script. The main part of the script is enclosed in `while` cycle. The cycle is repeated until the *exit button* on joystick is pressed. During at first the information from the joystick is read and the azimuth and elevation is decoded. If any of *alarm buttons* is pressed, the corresponding alarm sound is played. If the *play button* is pressed, the positioned sound is played. Before the sound playing the presence of 3D data is tested and the angle of elevation is evaluated. If 3D sound data exists, and the elevation angle is not zero (with some tolerance), the upper or bottom matrix is used. Otherwise the zero elevation data are played.

Then the picture from the camera is displayed, the cursor with appropriate color (blue means in front of the user, red means in the back) and the figure of azimuth. Outside the picture are displayed (see e.g. the Fig. 4):

- above the picture
  - *az* – azimuth of the cursor
  - *el* – elevation of the cursor
  - *x, y, z* – the coordinate from the joystick
- bellow the picture
  - *az* – azimuth of the data played (button pressed)
  - *el* – elevation of the data played
  - *row* – row of the matrix played
  - *but* – button pressed, one of this
    - non – none of the buttons is pressed
    - *play* – the sound is played
    - *exit* – the exit button was pressed
    - *al1, al2, al3* – the alarm is played (one of three)
  - *vol* – volume for the users headphones (can be controlled from the joystick control)

The above listed information are vital for the operator, because he don't hear the signal for the user.

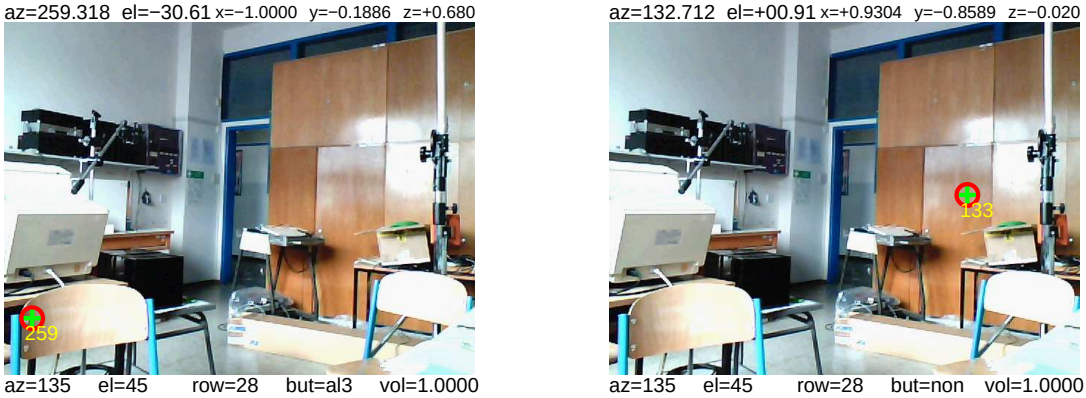

Figure 4: Examples of the operator's interface. Position of last played sound is different from the actual position of the cursor, the level in the user's headphones is set to 100 %. Left – the cursor is in the back of the user, bottom plane, the alarm 3 sound is played. Right – the cursor is in the back of the user, horizontal plane, no button is actualy pressed.

2.3 Implementation details

For the script presented here, reading of the information from the joystick is done using the solution presented as [4]. The image from the camera (cameras) is captured using the [5]. For creating the sound data were used the HRTF data from [3]. More details can be found in description of the first version [1].

3 Conclusion

This Matlab implementation is used for design of the system and testing the algorithms for virtual sound source positioning. The actual version extends function of the first version [1] to 3D space. Presented implementation is very simple, but is working near the real-time and can be (and already is) used for design of the system and testing the algorithms for virtual sound source positioning.

The function of the implementation was tested with the experimental device from the PERSEUS project [2], as seen in Fig. 1. The purpose of the actual version is testing of the orientation in virtual auditory space (and testing of the algorithms of the creation of the virtual audiory space), so, only one of the PERSEUS cameras is used (2D picture only). The results from experiments on the testing setup show, that the system described in the paper can be used for above defined purpose.

Acknowledgement

This work was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS11/159/OHK3/3T/13 and the research program MSM 6840770014 of Ministry of Education of the Czech Republic. Presented algorithms use data from CIPIC HRTF database [3].

References

[1] Rund, F.: Audio Pointer - Joystick & Camera in Matlab. In Technical Computing Bratislava 2010 [CD-ROM]. Bratislava: RT systems, s.r.o, 2010, p. 1-3. ISBN 978-80-970519-0-7.

[2] Vítek, S., Klíma, M., Husník, L., Špirk, D.: New Possibilities for Blind People Navigation. In 2011 International Conference on Applied Electronics. Plzeň: ZČU v Plzni, 2011, p. 405–408. ISBN 978-80-7043-987-6.

[3] Algazi, V. R., Duda, R. O., Thompson, D. M., Avendano, C., The CIPIC HRTF Database, Proc. 2001 IEEE Workshop on Applications of Signal Processing to Audio and Electroacoustics, pp. 99-102, Mohonk Mountain House, New Paltz, NY, Oct. 21-24, 2001.

[4] Bodson, M. Fun Control Experiments with Matlab and a Joystick. Last updated 6 Jan. 2010. <http://www.ece.utah.edu/~bodson/fun/index.html>

[5] Kazuyuki Kobayashi. VCAPG2 Matlab Central. Last updated 12 Feb. 2003. <http://www.mathworks.com/matlabcentral/fileexchange/2939>

František Rund
Katedra radioelektroniky, FEL ČVUT v Praze, Technická 2, 166 27, Praha 6
tel. 22435 2108, e-mail: xrund@fel.cvut.cz