

High Level Simulator of Spatial to Auditory Mapping System for Blind and Visually Impaired

Miloš Petković and Goran S. Đorđević

Abstract – In this paper a high level simulator for conceptual testing of aid system for blind and visually impaired people is described. System converts spatial map of near by objects into corresponding auditory map for a user to hear.

Keywords - High Level Simulator, MATLAB, Blind Aid System.

I. INTRODUCTION

In order to help blind and visually impaired people to move more freely and independently a system for converting spatial map into corresponding auditory map was considered. System comprises of two functionally independent parts as shown in Fig. 1. The first part of the system creates spatial map of surrounding objects by using some of 2D or 3D ranging and positioning methods/systems. The second part of the system converts spatial map into corresponding audio signal that contains enough information about positions of nearby objects but in the same time doesn't confuse a user with too much of data for interpreting. Since finding a fine balance between these opposing objectives is not trivial the goal of this paper is to introduce a simulator for testing different ideas. MATLAB was used as designing platform for this purpose [1].

The paper is organized as follows. The subsequent section describes different types of spatial to auditorial conversion. The third section explains the functionality of the simulator, while the fourth section presents the implemented algorithm. The simulation results are given in the fifth section.

II. AUDITORY MAP

Finding a right way to convert spatial information of surroundings into easy to interpret audio signal which creates vague but full picture of nearby objects in user's mind is rather empirical. Nevertheless, the good guiding line is human ear characteristic [2, 3] and Head-Related Transfer Function (HRTF) [4, 5, 6, 7]. Normally, people can guess the position of a sound source by comparing

what they hear on the left and right ear. The difference can occur due to slight time delay because sound travels a little longer to one ear than another. This is called *interaural time difference* – ITD. For higher frequency sounds the human head acts as an obstacle which causes intensity difference between sounds heard on left and right ear. This is called *interaural intensity difference* – IID. IID differs significantly from person to person since it depends on shape of outer ear, head and torso. By ITD and IID people can guess the azimuth of the sound source. ITD and IID help to determine the distance to the source although the overall intensity of the heard sound also counts. The sound intensity is inversely proportional to the square of distance from the source. Therefore, one tends to guess how far the source is by overall decrease of sound intensity when known the intensity of sound at the source. However this is also guessed by listening at different radiuses from the source. Since humans have two ears in-line then it would seem that determination of the elevation angle of sound source is impossible. Fortunately, the shape of outer ear helps in different frequency attenuation depending on the elevation of the source. Finally, it is important to mention that people learn to guess sound source position or better to say learn their auditory map since birth. It is also assumed that children blind from their birth have trouble with doing this as they lack visual feedback.

Generally speaking people are best at determining azimuth of sound source. Accuracy at determining distance is a little bit worse. People are pretty limited at guessing elevation and not that accurate. Having all these in mind several methods for generating audio signal for detected object was made.

The main idea for system was to create a feeling in users mind that objects around are emitting sound. To be more precise, that some points on surface of objects are producing sound. Like that some virtual *audio markers* are placed at certain points on surface of objects. The sound they produce could be one frequency sine wave - *monotone*, various frequency sine wave - *multi-tone* or even music. Every of these three has it's own advantages and disadvantages. When only one frequency sine wave is used a user can adjust the frequency that fits him the best and thus still be able to hear the rest of naturally generated sound by surroundings. A loudness control of generated sound is implied in all three mentioned cases. Since only one frequency is in use then only one object-point (audio marker) could play sound at a time. This would mean that some order in witch audio markers will play themselves

Miloš Petković is PhD student of Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia, E-mail: misa5ko@yahoo.com.

Goran S. Đorđević is with the Department of Control Engineering at Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia, E-mail: goran.s.djordjevic@elfak.ni.ac.rs

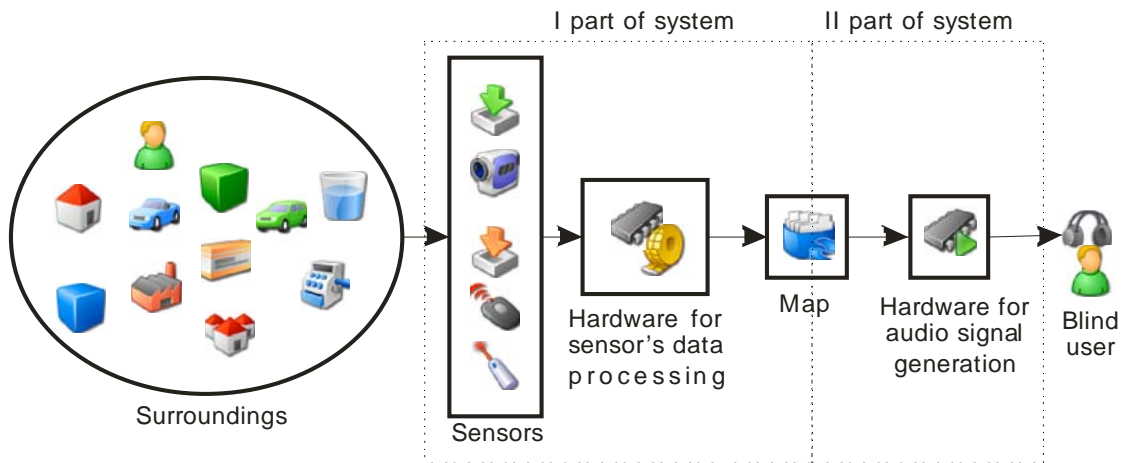


Fig. 1. Block diagram of system in use

has to exist. As said in introduction, position map of surroundings is created by processing data from 2D or 3D sensory system. 3D mapping is preferred so to eliminate possible misdetection of objects. However it is highly likely that 3D maps would have too much data for audio. What now may come as a question is which plane is going to be used in sound generation. That should be left for user to choose. Since only one plane is analyzed then the order in which audio markers are played could be defined as from right to left (anticlockwise) or from left to right (clockwise) from the users point of view. That would mean that for anticlockwise directions first would be heard objects on the most right of the user and then the ones at the left.

In order to create this azimuth position feeling principle something similar to IID can be used. By balancing sound intensity on left-right stereo audio signal user will have impression of horizontal displacement of an object. Intensity difference of left and right signal dependant on azimuth angle is given by Eqs. (1) and (2):

$$a_r = \cos(\frac{1}{2}\theta) \quad (1)$$

$$a_l = \cos(\frac{1}{2}\theta), \quad (2)$$

where θ is azimuth angle while a_r and a_l are coefficients with which are right and left audio channel signals multiplied. This is solid enough approximation of IID. As said before IID differs significantly from person to person and would need to be measured for each user. Finally by making sound intensity inversely proportional to square of distance from user to the source user could guess position of object. This eliminates completely measurements of IID and ITD.

The fact that human ear is more sensitive to frequency than intensity changes could be used in three different ways. Instead of sound intensity the frequency of sine wave can be distance dependant. For instance when an object is near the sound will be of high frequency and on the contrary when it is far frequency will be low. This could boost distance guessing accuracy. Similarly may be done with azimuth. When object is at the right corresponding

signal representation. Humans have very narrow elevation angle detection range. Therefore data only from one plane of 3D map is used for generating audio signal. It is left for system to detect collision and to warn a user that an object not visible in current plane of view is on the user's way. markers sound will be low, and vice versa. That could boost azimuth angle guessing accuracy. The second advantage of azimuth-frequency relation is that since for one azimuth angle can exist only one closest marker then different markers will have different frequencies. So all markers can be played at the same time thus increasing amount of information transmitted by sound.

Since mentioned sine wave sources can be irritating to listen music can be used as a substitute. As with mono tone sound source intensity would be proportional to the azimuth by Eqs. (1) and (2) and intensity would be inversely proportional to the square of distance.

III. SIMULATOR DESCRIPTION

The goal of this paper is to describe a simulator able to correlate spatial and auditory map.

Simultaneously it should help system designers to exam different ideas and to help potential users to adapt to the system. It has to be able to switch from one correlation method to other. Besides the simulator has to allow creating virtual environment with moving objects. Finally it must have ability to measure distances to objects so as replacement for real world environment and sensors readings.

Main functionalities of the simulator are:

- a) to be able to create, save and modify virtual maps,
- b) to simulate movements in virtual environment,
- c) to measure distances between virtual object and virtual user,
- d) to convert the position information into audio signal.

The simulator was made with only one plane or better to say with 2D map view of virtual world from above. This was done in order to speed up simulator so it could work in

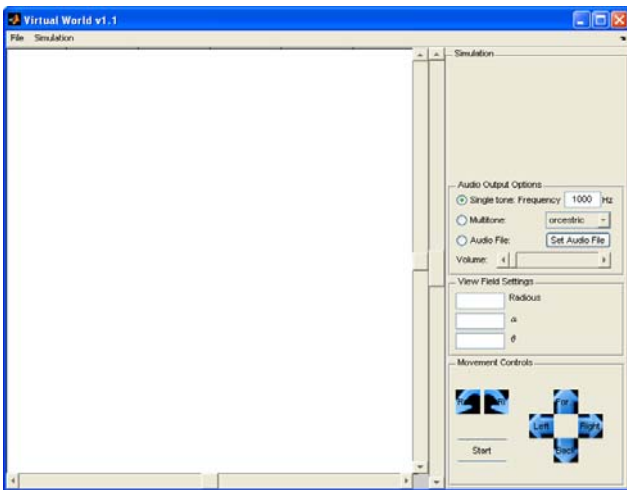


Fig. 2. Simulator's window after start up

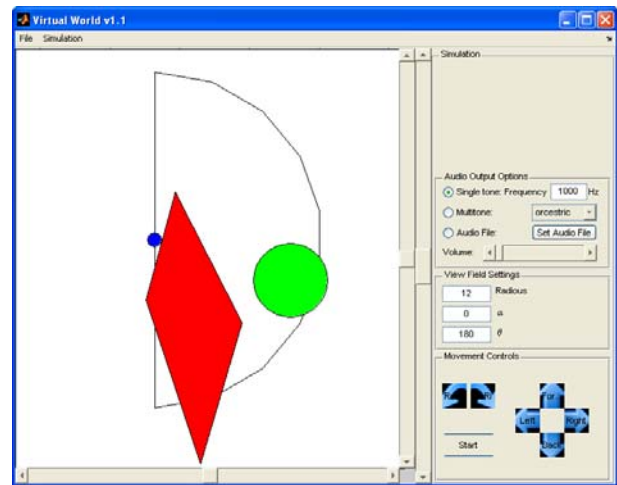


Fig. 3. Simulator's window with abstract map loaded

real time. The additional feature of possible collision detection with objects not seen in this plane is left out as currently irrelevant. In Fig 2. simulator window is shown after start up. The central plane is blank because no map has been loaded jet.

An saved map consists of array of objects with their properties, the user and its parameters and simulation time step.

There are two object types: polygon and circle. Polygon is defined with array of (x, y) pair of coordinates. There is no limit of number of pairs but the speed of simulation is inverse proportional to their number. That is why circle shape objects are included in simulator. They make creation of rounded objects easier than by using polygons. Also simulation speeds up when circle is used than multi line polygon replica of circle. These two types of objects are shown in Fig. 3 as red rectangle and green circle. Blue circle represents user and broken line circular sector is *audio filed of view* - AFV. AFV is area of space where object has to be in order for sound to be generated for it. This is very good way for a user to control amount of information converted to sound. The user can adjust AFV radius and the field of view. That means when a user is in open space radius can be bigger since there is no that much of objects around. To the contrary, when user finds himself at tight space with a lot of objects he can not only reduce the radius but also reduce the angle of view. Object detection in AFV is done with certain angular resolution. In other words there are certain number of angels at which an object presence is checked. This exist with two reasons. The first is to control amount of data converted to sound. The other one is to simulate impact of various sensors resolutions. Angular resolution is expressed in number of sub angles in AFV angle of view, not in degrees. While radius and field of view of AFV could be changed during simulation angular resolution can not. This is done because increasing a resolution increases time for completing right to left playing cycle that exist for cases when only one audio marker emits a sound. Therefore, normally real

angular resolution can be increased by narrowing angle of view.

When creating an object its significant coordinates can be inputted numerically or by clicking with a mouse pointer on the map. For a polygon significant coordinates are its endpoints' coordinates. Significant coordinates for circle are coordinates of centre and one point on its circumference.

To simulate realistic object movements segmented linear and rotational movements of objects can be defined. On every segment linear speed is constant. If segmented enough every type of movement can be achieved. For every linear segment a rotational speed can be defined as well as initial angle rotation at the beginning of each segment. Segment definition is relative, not absolute. This means that a segment path is defined by polar coordinates relative to position of object at the beginning of the segment. Polar coordinates can be inputted numerically or by clicking with mouse pointer at the map at the endpoints of the segment. When object comes to the end of its path it can be set to do one of four things: a) Stop – stop moving, its state is stop, b) Back – object continue to move backwards, reversing the linear segments sequence and their angles, but not the angles of rotational movements, c) Reverse – object continue to move backwards, reversing the linear segments sequence and their angles, as also as angles of rotational movements, d) Forward – object continue to move again as at the beginning of its path.

For moving the virtual user and changing AFV parameters an easy to detect under fingers "f", "j" and numpad's "5" keys were used as also as their neighbouring keys. This was done in order to make simulator friendly enough for blind people to use it too. That way direct users of the system in develop can give their opinion about it at an early stage of project thus reducing costs.

As it can be seen in Fig. 3 on the right side of simulator's window are audio output controls. There are three different options for sound source type as described in the previous section: single tone, multi tone and audio

file. When single tone is selected audio marker's sound source is sine wave of desired frequency which is defined by the input field adjacent to single tone radio button. Sound intensity is inverse proportional to square of distance from user to the source. Azimuth angle information is turned into intensity difference of left and right audio channels according to Eq. (1) and (2). Only one audio marker is played at a time. With multi tone audio option audio marker use various frequency sine wave as a source. There are three possible variants that are chosen from adjacent drop down list box: *orchestric*, *angular* and *radial*. When *radial* is chosen the frequency depends on distance of user to marker. So when an object is near the frequency will be high and vice versa. Azimuth angle information is turned into intensity difference of left and right audio channels according to Eq. (1) and (2). Only one audio marker is played at a time. In *angular* variant the frequency of marker depends on the azimuth angle. So for objects to the right the frequency is low and for the ones on the left it is high. Sound intensity is inverse proportional to square of distance to the source. Only one audio marker is played at a time.

Orchestric is similar to *angular*, frequency of marker depends on azimuth angle. Since for one particular azimuth angle there can be only one nearest audio marker than all audio markers will have different frequencies. This means that all audio markers can be played at the same time which is the case with this option. Sound intensity is inverse proportional to square of distance to the source.

If an audio file is to be used as a marker source it has to be firstly chosen by clicking on Set Audio File button. Only wav files are supported. Song is cut in appropriately long pieces that are successively played by markers. In order for the song to be heard constantly rather than in torrents a default song level is defined. So if for a particular azimuth angle marker doesn't exist a corresponding piece of song is played with a default volume level. As intensity of sound is inverse proportional to square of distance to the source, a default volume level was defined to be like the marker was on the outskirts of AFV. Azimuth angle information is again turned into intensity difference of left and right audio channels according to Eqs. (1) and (2). During simulation in the visual feedback screen a red line connects the virtual user and the obstacle that caused generation of audio sound. When there is no object in currently observed azimuth angle a green line stretches along from user to the border of AFV. This visual representation of currently observed azimuth was introduced to make it easier to track it.

IV. SIMULATION ALGORITHM

Simulation algorithm is given in Fig. 4 as Matlab pseudo code.

When the simulation is started copy of all objects data is made. This is done from stability reason imposed by Matlab. Next, proper graphical interface is initialized as

well as windows audio device. At the beginning of every simulation step it is being checked if user has issued a command for simulation to stop. Then in every cycle update of data is done. That implies movement commands for virtual user like forward, backward, left, right, rotate left/right, increase/decrease view angle, increase/decrease AVF radius. Now, a new position of user is calculated with assumption that it didn't collide with any static object. The following simulation loops through every static objects and checks if there were any collisions. Only static objects are considered because modelling collisions with different moving objects would consume much of simulation time. To every object type one should specify the elasticity parameter simulating the type of material. The purpose of this simulator is not to completely simulate real world interactions but to check the ability and usefulness of the system. Since only collisions with static objects are checked, then it is done by inspecting if there is any cross-sections between line which endpoints are current and new position of user and any line of object. There are two distinguished types of object so there are two slightly different methods for their collision detection. If collision is detected with a current object in loop the length of user movement is reduced so that user stops in front of the object. Next, simulator loops through every moving objects in order to update their positions. As said, the collision is not checked. First step in this loop is to calculate new position of object with assumption that it will still be on the same current segment of the movement path. Then the assumption is checked. If it is not true the rollover for the next segment must be calculated. If that was the last segment the at end command for the object is checked. As said before, one of four different actions is performed. After recalculation of new position with new segment, its validity is evaluated again. After new position is found, the object is rotated accordingly for every segment. Rotation of a circle objects is meaningless and not performed.

After all repositioning is done a loop for finding the nearest obstacles in AFV is executed. First it is checked if current object of loop cycle is visible from current view angle of user. If it is true a distance is calculated and if it is shorter than previously found one it is set as a new one. When looping is finished, the appropriate audio signal is generated according to the current settings and sent to PC audio card.

V. SIMULATION RESULTS

Several simulation with real life situation settings were performed. Two of them are shown in Figs. 5 and 6.

Fig. 5 represents every day situation of navigation through a typical living room. There is entry to the room in the bottom left corner. Pieces of furniture are tagged. Settings is completely static. Easy navigation was accomplished.

Fig. 6. presents a common situation of navigation on the street. This is rather dynamical and unknown to user

setting. Objects are tagged. An important conclusion was made after this simulation. It may happen that markers are played too slow for car moving in AFV to be detected by user. In another words since car moves too fast its audio markers never get their chance to be played. Nevertheless the potential usability of system was confirmed.

VI. CONCLUSION

In order to help blind and visually impaired people a system for converting position information of near by objects into stereo audio signal was conceptually tested by an original simulator. The simulator was built in MATLAB inheriting all its advantages and flaws.

Several conversion methods were tested and some showed more potential than others. The one that stood out was the single tone method. However this conclusion is strictly subjective. One can prefer one method more than another. Without further testing with more users nothing is final. Of course system should have flexibility for user to select the method that best works for him.

In order to make simulator more realistic for rotational movements of user, a rotational encoder can be place on rotatable chair. Readings from encoder would then be input for the simulator. This way the simulator will give a realistic figure in case when user rotates himself on the chair.

```

copy_data();
initialize_interface();
setup_audio_device();

while simulation_not_terminated
    update_data();
    %move user
    calculate_users_new_position();
    for i=1:n
        if object_is_static
            if line_defined_with_old_and_new_user's_position_and_object_line_have_cross-section
                reduce_users_move_so_there_is_no_cross-section();
            end
        end
    end
    %move objects
    for i=1:n
        if object_is_moving
            calculate_new_position_for_current_segment();
            while new_position_not_on_current_segment
                if there_is_no_next_segment
                    switch at_end_object_should
                        case 'Stop'
                            change_object_state_to_static();
                        case 'Back'
                            reverse_object's_segments_order_but_not_rotation_angles();
                        case 'Reverse'
                            reverse_object's_segments_order_and_change_rotation_directions();
                        case 'Forward'
                            go_back_to_the_first_segment_of_object's_movement_path();
                    end
                else
                    go_to_next_segment();
                end
                calculate_new_position_for_current_segment();
            end
            rotate_object_accordingly();
        end
    end
    %find distances of object in AFV
    initialize_solutions();
    for i=1:n
        calculate_distances_of_this_object_in_AFV();
    end
    %generate audio signal accordingly
    generate_sound_output_for_current_audio_output_settings();
    output_sound_on_audio_device();
end

```

Fig. 4. Simulator's algorithm written in pseudo MATLAB code

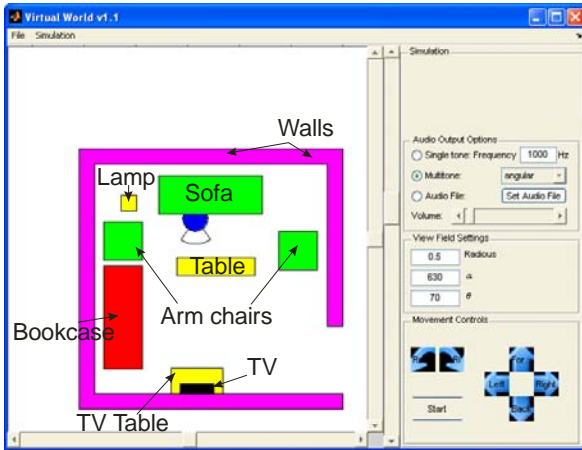


Fig. 5. Simulator's window with abstract map loaded

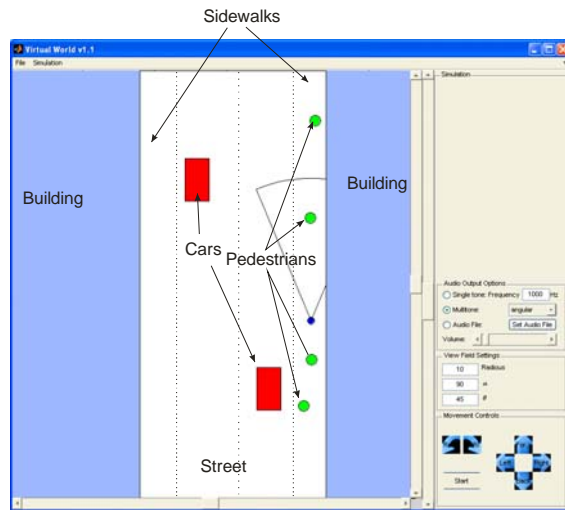


Fig. 6. Simulator's window with abstract map loaded

REFERENCES

- [1] Marchan P., Holland T., *Graphics and GUIs with MATLAB*, Chapman & Hall, 2003.
- [2] HyperPhysics, „Sensitivity of Human Ear“, *Sensitivity of Human Ear*, 2005. [Online] Available on: <http://hyperphysics.phy-astr.gsu.edu/Hbase/sound/earsens.html>. [Accessed: 11. Jul 2009.]
- [3] Wikipedia, „Equal loudness contour“, *Equal loudness contour*, 2009. [Online] Available on: http://en.wikipedia.org/wiki/Equal-loudness_contour. [Accessed: 11. Jul 2009.]
- [4] Christopher J. Plack, *The sense of hearing*, Lawrence Erlbaum Associates, 2005.
- [5] Wikipedia, „Sound localization“, *Sound localization*, 2009. [Online] Available on: http://en.wikipedia.org/wiki/Sound_localization. [Accessed: 11. Jul 2009.]
- [6] Wikipedia, „Head-related transfer function“, *Head-related transfer function*, 2009. [Online] Available on: http://en.wikipedia.org/wiki/Head-related_transfer_function. [Accessed: 11. Jul 2009.]