ARCHIVES OF ACOUSTICS
Vol. **37**, No. 3, pp. 301–316 (2012)

DOI: 10.2478/v10168-012-0039-y

Beam Tracing with Refraction

Marjan SIKORA(1), Ivo MATELJAN(1), Nikola BOGUNOVIĆ(2)

(1) *Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, University of Split*
Ruđera Boškovića 32, Split HR-21000, Croatia; e-mail: sikora@fesb.hr

(2) *Faculty of Electrical Engineering and Computing, University of Zagreb*
Unska 3, Zagreb, HR-10000, Croatia

(*received September 30, 2011; accepted June 22, 2012*)

This paper presents the beam tracing with refraction method, developed to examine the possibility of creating the beam tracing simulation of sound propagation in environments with piecewise non-homogenous media. The beam tracing with refraction method (BTR) is developed as an adaptive beam tracing method that simulates not only the reflection but also the refraction of sound. The scattering and the diffraction of sound are not simulated. The BTR employs 2D and 3D topology in order to efficiently simulate scenes containing non-convex media. After the beam tracing is done all beams are stored in a beam tree and kept in the computer memory. The level of sound intensity at the beginning of each beam is also memorized. This beam data structure enables fast recalculation of results for stationary source and geometry. The BTR was compared with two commercial ray tracing simulations, to check the speed of BTR algorithms. This comparison demonstrated that the BTR has a performance similar to state-of-the-art room-acoustics simulations. To check the ability to simulate refraction, the BTR was compared with a commercial Finite Elements Method (FEM) simulation. In this comparison the BTR simulated the focusing of the ultrasound with an acoustic lens, with good accuracy and excellent performance.

Keywords: simulation, beam tracing method, refraction.

1. Introduction

In acoustics, the beam tracing method has primarily been used to simulate architectural environments in which waves propagate in a single homogenous medium. Other geometric methods, such as the virtual image source method and the ray tracing method, have also been designed to analyze the propagation of sound in homogenous medium. Simulating the propagation of sound in a non-homogenous environment with more than one medium has typically been conducted with method like Finite Elements Method (FEM), which calculates the numerical wave equation solution. Although these methods are well developed and widely used, they have significant limitations (wavelength/dimension ratio, simulation in full 3D). That is why, in many applications numerical simulations are not a good or practical choice. The method presented in this paper – the beam tracing with refraction (BTR) method – was developed with the intent to fill the gap between geometrical and numerical wave equation solution simulations. The BTR aims to simulate environments containing more than one media (which until present time has not been simulated with the beam tracing method) that have large dimensions compared to the sound wavelength and that require simulation in 3D (which is a problem for numerical method of wave equation solution).

In the second section of the paper, a brief review of existing acoustic simulation methods and their characteristics is presented. In the third section, the BTR is fully explained. The fourth section presents results from simulations of several scenes with the BTR and compares them with results from two geometrical simulations and one FEM simulation. The fifth section presents the conclusions and ideas for future work.

2. Previous work

Simulation methods in acoustics can be divided into two groups: geometrical and numerical wave equation solutions. Geometrical methods are primarily used for room acoustic simulations.

The first geometrical methods that were developed were the virtual source method and the ray tracing method (Kleiner *et al.*, 1993). These methods are often combined to overcome their individual limitations. In these combinations, the virtual source method is used to simulate early reflections (because of its accuracy), and the ray tracing method is used to simulate later reflections (because of its efficiency). Along with specular reflections, which are the most important phenomena in room acoustics, the ray tracing method is used to simulate diffuse reflections and diffraction. There exist several modified versions of the ray-tracing method such as the ray tracing in time domain (Alpkocak, 2010) or the ray tracing with refraction (Fink, 1994). The ray tracing with refraction method was developed by Kevin Fink for the simulation of acoustic lenses. This ray tracing simulation calculates focusing of the sonar beam by tracing the propagation of rays through an acoustic lens made of rubber and through seawater, a medium in which sonar is used.

The beam tracing method was first reported by Heckbert and Hanrahan (1984) and by Walsh, Dadoun and Kirkpatrick (1985) as a visualization technique. This method was developed to overcome limitations of the ray tracing method while retaining its efficiency. Because it uses beams instead of rays and a point detector instead of a spherical detector, it ensures spatial coherence and avoids the problem of aliasing inherent to the ray tracing method.

Siltanen *et al.* (2007) presented the acoustic radiance transfer method based on the room acoustic rendering equation (RARE). The RARE is an integral equation which generalizes above mentioned geometrical room acoustics modeling algorithms. Its formulation is adopted from computer graphics. Similar to the BTR, it calculates intensities of sound instead of the pressure, and to get the intensity at any point it sums intensities transmitted through the direct path and the reflections reaching that point. Like the BTR the memory consumption of the RARE algorithm is high, in typical cases hundreds of megabytes, but this allows the recalculation of the intensity of sound for different receiver without repeating complete algorithm. As it is a room acoustics modeling algorithm, the RARE doesn't include effects of the refraction of the sound.

Since the BTR is based on the beam tracing method, its development would now be examined in more detail. The beam tracing method was first used for acoustic simulation by Maercke, Maercke and Martin (1993), who used cone tracing instead of ray tracing. They overlapped cones slightly to achieve spatial coherence, which became a potential source of aliasing. This problem was solved later by Lewers (1993), who used beams with triangular sections instead of cones. This enabled him to achieve full spatial coherence without overlapping the beams. He also introduced a method of radiant exchange for calculating diffuse reflections. The next step in the development of the beam tracing method was taken by Farina (1994; 2000) and by Drumm (2000). They introduced an adaptive beam division algorithm that divided a beam into several smaller beams when it encountered more than one reflective plane. Both authors incorporated simulation of diffuse reflections into their models, and Farina simulated edge scattering as well.

The development of the beam tracing method continued toward interactive applications with the introduction of advanced spatial data structures. Funkhouser *et al.* (1998) presented an implementation of the beam tracing method for interactive auralization of architectural environments. Using spatial data structures, they achieved the performance required for interactive simulation of a moving listener and a stationary source. Later, they added diffraction to their model (Tsingos, Funkhouser, 2001) to improve the quality of the auralization. Finally, Laine *et al.* (2009) presented their implementation of the interactive beam tracing method, which was (in scenes of moderate complexity) able to produce auralizations at interactive rates, not only with a moving listener but also with a moving source.

Until now, implementations of the beam tracing method that simulate refraction existed only in the field of visualization, in which the refraction of light is simulated to visualize the interaction of light with water or with curved objects made of glass (Shah, Pattanaik, 2007). As far as the authors are aware, there is no beam tracing simulation in acoustics that is able to calculate the refraction of sound. The FEM is currently the method of choice for simulating the refraction of sound. The FEM is used for simulation in fields such as medical ultrasound, sonar and geology (Shah, Pattanaik, 2007; Wojcik *et al.*, 1994; 1998), both to design ultrasound transducers and to analyze the propagation of sound. The FEM is suitable for simulation of environments with many different media. In the FEM, a scene is composed of a number of finite elements, each of which can have different acoustical parameters. This structure allows the FEM to be used not only to simulate scenes containing several media but also to simulate diffuse environments in which there are no clear boundaries between different media. The FEM simulates all relevant wave phenomena, including reflection, refraction and diffraction, as it is based on numerical integration of the wave equation.

Besides the FEM, several other numerical wave equation solutions methods have been developed such as the acoustics diffusion equation (ADE) modeling and the finite difference time domain (FDTD) method. The ADE modeling uses the analogy of the movement of a single particle in a gas to model the sound field (Picaut *et al.*, 1997). In the FDTD method sound pressures and particle velocities are estimated at discrete grid locations for successive discrete times (Botteldoren, 1994).

The complexity of a FEM simulation is determined by the number of finite elements in the model. If the dimensions of the model are large compared with the wavelength of the sound, the model must be divided into a large number of finite elements. The complexity of simulating such a model can be a problem for a desktop computer. If the FEM simulation needs to be performed in full 3D, the computational complexity becomes even more problematic, and these simulations can often only be performed on computer clusters and grids.

Areas of application for beam tracing and the FEM in acoustics are shown in Fig. 1. Existing implementations of the beam tracing method in acoustics can efficiently simulate scenes where dimensions of model are larger than the wavelength of the simulated sound. The other limitation of the traditional beam tracing method is that the model contains only one medium. This makes the beam tracing method suitable for room acoustics simulations and virtual reality applications. FEM simulations can simulate models with any number of media, ranging from single-medium environments to diffuse environments, but when the size-to-wavelength ratio of the model increases beyond a certain limit, the computational complexity limits the utility of the FEM. The FEM is thus used in applications such as the medical ultrasound and the detection of faults in materials.

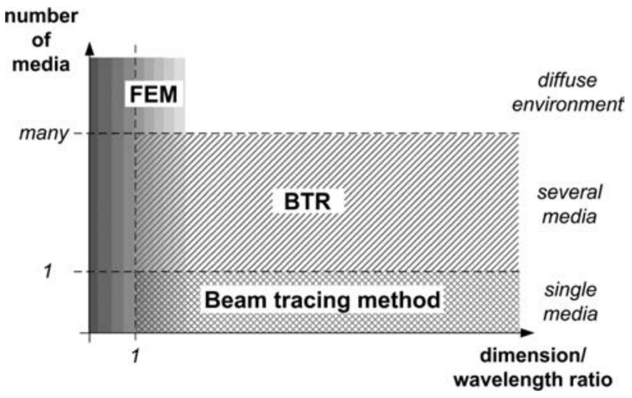


Fig. 1. Areas of application of simulation methods.

There is an area of simulation application that cannot be efficiently simulated with the beam tracing method or with the FEM. In these applications, the propagation space has dimensions that are large compared with the wavelength of the simulated sound, the space is composed of more than one media (or one medium with variable physical characteristics), and the simulation has to be done in full 3D. The BTR was developed with the intent to efficiently predict intensity of ultrasound in marine environments such as farmed fisheries and oyster farms, and for the simulation of the ultrasound propagation in the human head, where in medical applications the ultrasound propagates through several layers of soft tissue and bone.

3. Models and methods

The BTR is designed to simulate the propagation of sound in non-convex spaces composed of different media. The BTR simulates two wave phenomena: specular reflection and the refraction of sound. Diffuse reflection and diffraction of sound are not currently incorporated in the BTR. These phenomena are already solved with the beam tracing method, and published by several authors (Farina, 2000; Drumm, 2000; Funkhouser *et al.*, 1998; Tsingos, Funkhouser, 2001). The BTR has been specialized to include the refraction.

This section first explains the physical model of the BTR, then data structures and algorithms of the simulation, and finally the implementation of the simulation.

3.1. Physical model of simulation

The BTR traces beams with triangular cross-section. When beams hit the boundary surface between two media, they are divided into smaller beams, to adjust to individual triangles that make the boundary surface. After the division, each divided beam is reflected or refracted, and such beams are further traced in the same manner. The tracing stops when the level of the intensity on the end of the beam is smaller than predefined threshold, or when the volume of the beam is smaller than predefined volume threshold.

The result of BTR simulation is the level of sound intensity at a single point or at points in a rectangular raster. First, the physical model used for calculation of the intensity level will be explained, and after that the method for constructing the geometry of the reflected and the refracted beam.

3.1.1. The intensity of sound

Following the law of power averaging, the total sound intensity at the receiver is calculated by summing the individual intensities of each beam that contains the receiver. The reason the BTR employs this method rather than summing the pressures of each beam is because the phase of the reflected sound pressure wave can be treated as a random variable in the case of a highly reverberant environment. If this method of calculation was used in a scene that is not highly reverberant, then the results would not include sound interference. The individual beam that contains the receiver can be either the direct beam (coming directly from the source), or indirect beam (that has been already reflected or refracted on the boundary surface between two media). The sound intensity at a point inside a direct beam is calculated using the following equation:

$$I = \frac{P_A}{4 \cdot \pi \cdot r^2} \cdot e^{-\gamma \cdot r}, \tag{1}$$

where P_A is the acoustical power of the source of the sound, r is the distance from the source and γ is the attenuation coefficient of the media within which the beam propagates. The first term in the equation describes the attenuation of the sound caused by the propagation of the spherical wave. The second term in the equation describes the attenuation caused by viscosity and other dissipative processes in the media (Pierce, 1981).

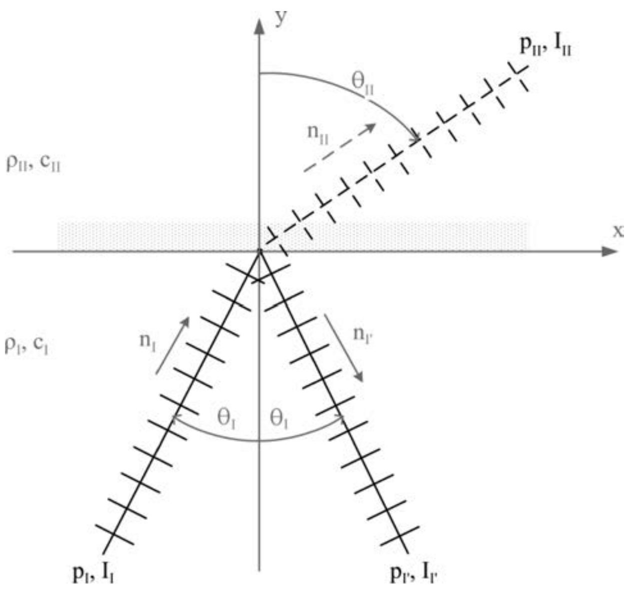


Fig. 2. The reflection and refraction of sound on a boundary surface.

When sound encounters a boundary surface that divides two media, the sound beam is reflected and refracted (Fig. 2). The sound intensity of such indirect beam is determined using the following equations:

$$I_{I'} = R^2 \cdot I_I, \tag{2}$$

$$I_{II} = \left(1 - R^2\right) \cdot I_I, \tag{3}$$

where I_I is the intensity of the incoming sound beam, $I_{I'}$ is the intensity of the reflected sound beam, I_{II} is the intensity of the refracted sound beam and R is the amplitude of sound reflection coefficient of the boundary (Pierce, 1981; Vorlander, 2008). The reflection coefficient $R$is calculated using the following expression:

$$R = \left| \frac{Z_{II} - Z_I}{Z_{II} + Z_I} \right|, \tag{4}$$

where Z_I and Z_{II} are the acoustic impedances of the two media. The acoustic impedance is function of the angle of incidence θ_I, as in (Pierce, 1981).

The traditional way to calculate the sound intensity of a point inside an indirect (reflected) beam is to generate path of the sound from the receiver to the source, in order to get the exact distance that the sound travelled. This path generation creates the exact path that sound traverses from the source to the receiver. It is computationally complex because one has to calculate all of the intersections of the sound path with the boundary surfaces using virtual sound sources.

In the BTR, to decrease the computational complexity of the path calculation, the intensity of the sound at a point inside an indirect beam is calculated in a different way (Fig. 3). The intensity is calculated relative to the sound intensity at the barycenter of the starting triangle of the beam (I_0) using the following equation:

$$I_{\text{BTR}} = I_0 \frac{r_1^2}{r_2^2} \cdot e^{-\gamma \cdot (r_2 - r_1)}, \tag{5}$$

where I_0 is the intensity of the sound at the barycenter of the starting triangle of the indirect beam, r_1 is the distance from the virtual source of the beam to the barycenter of the starting triangle of the beam, r_2 is the distance from the virtual source of the beam to the receiver and γ is the attenuation coefficient determined by the entity in which the beam propagates.

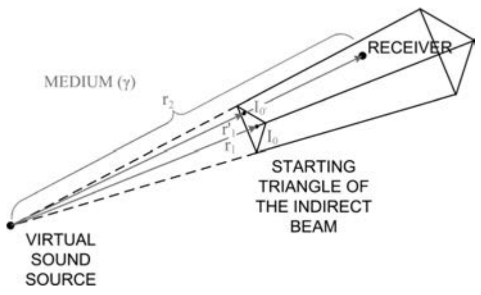


Fig. 3. Calculating the intensity of indirect sound in the BTR.

I_0 is calculated as the intensity of sound of original beam using Eq. (1), and then transformed with Eq. (2) for reflected, and with Eq. (3) for refracted beam. It is stored in the data structure of the indirect beam, so all data required to calculate the intensity of sound is stored with the beam.

The drawback of this method is that an error is introduced because the sound intensity should be calculated relative to the real intersection of the sound ray with the starting triangle of the indirect beam rather than the barycenter of the starting triangle. The equation for the exact sound intensity is:

$$I = I_0' \frac{{r'}_1^2}{r_2^2} \cdot e^{-\gamma \cdot (r_2 - r_1')}. \tag{6}$$

Let us examine the magnitude of this error. The error is largest for the first reflection/refraction of the beam because the beam is at its widest. The angular width of such a beam is 26.5° and it is determined by the shape of the icosahedron. For the square room, the worst error is 2.54 dB. The detailed description of how the worst error is calculated is given in Appendix A. For subsequent reflections and refractions, the error gets smaller because beams get divided, so they have smaller angular widths.

BTR has the similar complexity of the first part of algorithm – the tracing of beams – as other splitting beam tracing algorithms. But in the second part – the calculation of sound intensity – BTR shows the advantage over the traditional beam tracing method. In the BTR, there is no path generation, which can be time-consuming in the case of multiple reflections/refractions. All information needed for the calculation of the sound intensity is already recorded within the beam data structure. Let us consider the gain in the computational complexity. If n is the number of beams that hit the receiver, and m is the order of reflection/refraction, the time complexity of re-computing all m^{th} reflections in the BTR is linear: $O(n)$. In the traditional method with path generation, the time complexity is exponential $O(n^m)$ because there have to be m intersection points for n beams. The complexity is clearly considerably lower in the BTR. This feature of BTR has the capability of acceleration of the beam tracing algorithm, similar to already published efficient methods for the reduction of computation time (Stephenson, 1996; Siltanen *et al.*, 2009).

3.1.2. The geometry of the beam

When a beam hits a boundary surface, the geometry of the beam is changed. The BTR generates one reflected and one refracted beam for each incoming beam. Edge rays of reflected beam have the angle opposite to the angle of edge rays of the incoming beam – θ_I (Fig. 2). Edge rays of the refracted beam are generated according to Snell's law:

$$\frac{\sin(\theta_I)}{\sin(\theta_{II})} = \frac{c_I}{c_{II}}. \tag{7}$$

If the incoming angle θ_I is greater than the critical angle $\theta_{I-\text{crit}}$:

$$\theta_{I-\text{crit}} = \arcsin\left(\frac{c_I}{c_{II}}\right) \tag{8}$$

then only the reflected beam is generated.

Because the Snell law (Eq. (7)) is not linear, the beam loses focus after refraction. This problem was detected and described in one of the first papers on beam tracing for visualization (Heckbert, Hanrahan, 1984). In the BTR, this problem is solved by refocusing the refracted beam. To do this, three pairs of beam edge rays are intersected, resulting in three intersection points. The new focus of the beam is calculated as the arithmetic average of these intersections. Figure 4 illustrates the problem of losing focus of the refracted beam (the reflected beam is not shown). Edge rays of incoming beam are displayed with solid black line. Edge rays of refracted beam are displayed with dotted black line. Intersections of edge rays of the refracted beam are marked with circles. One can see that edge rays lose focus – they do not intersect in single point (Fig. 4 – detail – right).

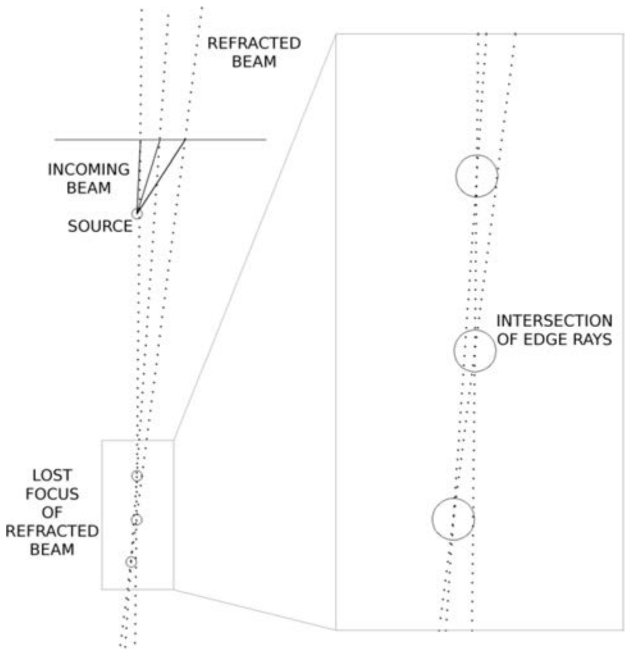


Fig. 4. The reflection and refraction of sound on a boundary surface.

This method of refocusing and its influence on the accuracy of the results have already been analyzed (Sikora *et al.*, 2010). It was shown that refocusing the initial beam causes the edge rays to have an average angular error of 0.13°. All subsequent beams are already divided and are thinner than the original beam. Thus, their edge rays are more parallel, which decreases this angular error. The influences of angular and intensity errors are tested using the example of refraction in an acoustic lens that is presented in Subsec. 4.2.

3.2. Data structures

3.2.1. Scene composition

The simulated propagation space in the BTR is called a scene, and its topology has three levels of hierarchy: entity, shell and boundary surface. An example scene is presented in Fig. 5.

An entity represents a volume containing a single medium and is defined by one or more shells. A simple entity is defined by a single shell. It can be convex (E1) or non-convex (E3). Entities that contain one or

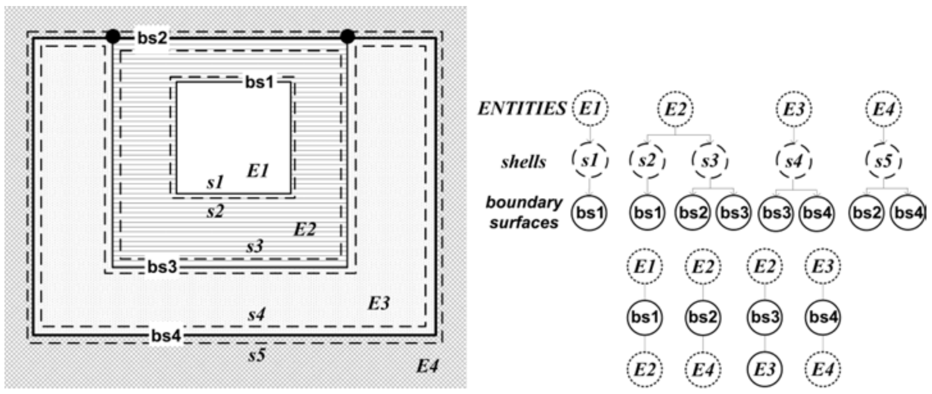


Fig. 5. The topology of a scene.

more islands are non-convex and have several shells (E2). A shell is a closed triangular mesh that defines the outer boundary of an entity (s2, s3, s4) or the inner boundary of an enclosing entity (s1, s5). A shell is two-dimensional object that is used to define the extent of an entity, which is a three-dimensional object. An entity doesn't have any geometry information, but only the list of shells that surround it. The orientation of triangles in the mesh of the shell defines the space that is inside of the entity.

A shell is composed of one or more boundary surfaces. A boundary surface is a triangular mesh that divides exactly two entities. Thus, a boundary surface is not always a closed mesh, but a set of boundary surfaces that compose one shell must be a closed mesh. A boundary surface has no orientation.

The scene topology is used during beam tracing to accelerate the geometric operations involving traversing beams between neighbor entities. When a beam hits a boundary surface, the topology determines in which entity it will continue to propagate, knowing that one boundary surface divides exactly two entities. In addition, as every beam "knows" inside which entity it is currently propagating, only local geometrical complexity affects the performance of the algorithm. This algorithm is explained in greater details in Subsec. 3.3.

Triangular meshes that compose boundary surfaces are topologically structured. They are composed of vertices, edges and triangles, and their spatial relationships are defined by a winged-edge structure. The mesh topology is used during beam division to speed up the process of finding which neighboring triangles are hit by the beam.

Boundary surfaces are based on triangular meshes instead of the polygons normally used in the beam tracing method. This choice was made because classical beam tracing is used in architectural environments, while the BTR is designed for man-made environments that are not necessary regular.

A binary space partitioning (BSP) tree is calculated for each entity in the scene. The BSP tree is used by the beam division algorithm to determine the correct order of visibility of illuminated triangles. The BSP tree is used to speed up the visualization process and to solve cases of cyclic hiding.

The mesh topology and the BSP tree are calculated automatically during the preprocessing phase of the simulation. The topology of the scene is not calculated by the simulation and has to be defined by the user when setting up the scene.

3.2.2. Beams

Beams in the BTR have triangular cross sections and are defined by three edge rays that intersect in the focus of the beam. Initial beams have the form of a triangular pyramid, with the focus of the beam at the position of the sound source. Transformed beams, which are created after reflection and refraction of the initial beams, take the form of clipped pyramids, with beam foci in the positions of the virtual sound sources (Fig. 6). The starting triangle of the beam is the place where the beam enters the entity through which it propagates. The ending triangle of the beam is the place where the beam leaves the entity, if it is refracted, or the place where the beam is returned back to the entity, if it is reflected.

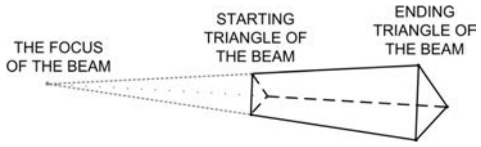


Fig. 6. A transformed beam in BTR.

In classical beam tracing, such as the method developed by Drumm (2000) and Funkhouser *et al.* (1998), beams have polygonal cross sections. The advantage of such an approach is that fewer beams are needed to trace a scene than are necessary with triangular beams. However, the performance gains that result from a lower number of beams are diminished by the complexity of polygon-polygon clipping operations that occur when such polygonal beams intersect with a geometry made of polygons. In the BTR, as all of the beams have triangular cross sections and because shells are defined by triangle meshes, all geometric operations operate on two triangles. Special attention was paid to the design of the triangle-triangle clipping algorithm that is used in the BTR with the goal of maximizing performance. The BTR clipping algorithm differs from the classical Sutherland-Hodgman algorithm (Sutherland, Hodgman, 1974). Rather than using a unified algorithm for all combinations of two triangles, the BTR first detects the combination of triangles with a few simple geometric tests and then creates the resulting clipped triangles for that particular combination. As a result, the BTR clipping algorithm performs better than other approaches.

3.3. Algorithm of BTR

After the scene geometry is defined and the calculation of the topology is done, the algorithm of BTR begins. Its flow diagram is shown in Fig. 7.

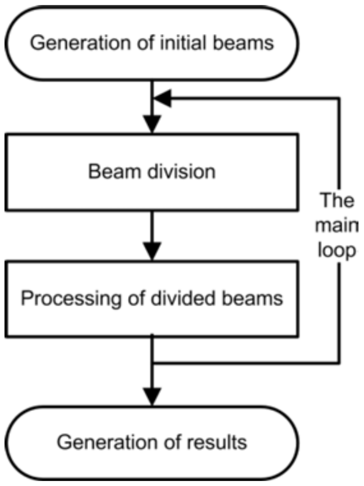


Fig. 7. The algorithm of BTR.

After the initial beams have been generated, the main loop of algorithm starts. In this loop, for every beam the beam division is performed, after which the divided beams are processed. After the main loop is finished, the results are generated in the form of raster with the distribution of sound intensity.

3.3.1. Initial beams

The process of beam tracing starts with the generation of initial beams. The foci of initial beams are at the position of the sound source. Beams are generated using the icosahedron in such a way that three edge rays of one beam pass through vertices of one triangle of an icosahedron. This process results in 20 initial beams. The cross section of each beam is an equilateral triangle. Initial beams propagate through the entity that contains the source of the sound. The information about the entity where the beam propagates is recorded with each beam and is updated whenever refraction occurs.

3.3.2. The main loop of the algorithm

After the initial beams are generated, they are pushed onto the raw beam stack, and the main loop of algorithm starts (Algorithm 1).

Algorithm 1. The main loop.

```
while the raw beam stack (RBS) is not empty
  pop one raw beam (rb) from RBS
  do the beam division of rb
  push beams produced by division of rb on
       the divided beams stack (DBS)
  process divided beams from DBS
repeat
```

In the main loop, one beam is popped from the raw beams stack and intersected with the shell(s) of the entity in which it propagates. This intersection triggers the beam division process. During the beam division process, the original raw beam is divided into several raw beams. The divided beams are then processed, resulting in several finished beams that are pushed onto the stack of finished beams and several new raw beams (from reflection and refraction) that are pushed onto the stack of raw beams. The loop repeats until the stack of raw beams is empty, which means that beam tracing is done.

3.3.3. Beam division

Each beam propagates inside of an entity and eventually encounters one of the entity shells that combine to represent the boundary of an entity. In the simplest case, the entire beam intersects only one triangle of the shell's mesh. In this case, it is not necessary to perform beam division. Generally, however, the beam intersects several triangles of the shell's mesh. In that case, the beam has to be divided into several smaller beams, one for each intersected triangle of the mesh. In addition, as entities in the BTR can be non-convex, some of the intersected triangles may obscure each other. In this case, the precise region that is not obscured has to be determined to generate an accurate beam division. The correct beam division is of the utmost importance because it ensures the spatial coherence of the beam tracing method and preserves the sound energy of the beam.

The beam division process implemented in the BTR is composed of these phases:

- finding intersected triangles and their transformation to the beam space (Subsec. 3.3.3.1)
- projecting and clipping triangles (Subsec. 3.3.3.2)
- hiding and dividing triangles (Subsec. 3.3.3.3)
- creating divided beams (Subsec. 3.3.3.4).

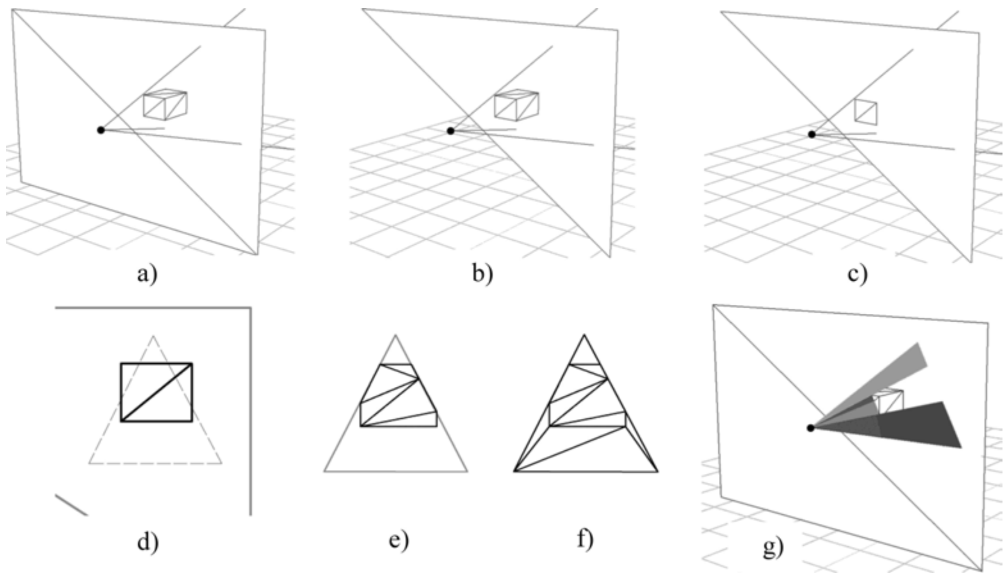


Fig. 8. Example of the process of beam division: a) the starting geometry, b) illuminated triangles, c) backface culling, d) perspective projection, e) clipping with the beam section, f) hiding and dividing triangles, g) divided beams.

The beam division process is illustrated in Fig. 8. This simple example shows the division of a beam, displayed with three edge rays, that propagates from the source displayed with small black circle. The beam first encounters a shell in the form of a box, positioned in front of another shell displayed with two large triangles (Fig. 8a).

3.3.3.1. Finding intersected triangles and their transformation to the beam space. The beam division process begins by determining the order of visibility of the triangles. This is done by traversing the previously calculated BSP tree from the position of the sound source. The next step is the phase of finding triangles that are completely or partially within the beam. These triangles are candidates for beam division. The backface culling process is performed to remove those triangles whose normal vector points in the direction opposite to the beam propagation. The triangles that remain are transformed from the world space to the beam space. The world space is the coordinate space of the scene as displayed in Fig. 8a. The beam space is the coordinate space that has z axis as the direction of beam propagation. After all triangles are transformed to the beam space, they are clipped with the plane $z = 0$. This is done to avoid geometry distortion due to parts of the triangles that are beyond the focal point.

This stage in the beam division process is illustrated in Figs. 8a–8c. Figure 8a displays the geometry on the start of the process. Figure 8b displays those triangles that are within the beam. In this stage the lower left triangle of the second shell is eliminated from further processing. Figure 8c displays the result of the backface culling process. After that stage only two front triangles of the first shell (box) and one triangle from the second shell remain for subsequent processing.

3.3.3.2. Projecting clipping triangles. Transformed triangles are then projected to 2D as preparation for the hidden surface algorithm. This algorithm is performed on the projected triangles in 2D rather than in 3D to increase performance. Triangles are projected using a perspective projection onto the projection plane orthogonal to the z-axis (the direction of beam propagation). Then, all projected triangles are clipped with the projected section of the beam to consider only the parts of the triangles that are within the beam volume. After this, the triangles are added to a binary sorted tree. The order in which the tree is sorted is determined by the order of visibility of the triangles.

This stage in the beam division process is illustrated in Figs. 8d and 8e. Figure 8d shows the perspective projection of triangles from Fig. 8c. Two triangles from the first shell (box) are displayed in black. They are positioned in front of the triangle from the second shell which is displayed in gray, and which is only partially visible. The projected section of the beam is displayed with thin dashed line. Figure 8e shows the triangles from Fig. 8d, clipped with the projected section of the beam. Two black triangles are divided into several smaller ones. The clipping of the gray triangle resulted with the single triangle. After clipping, black triangles still overlap the gray one. The reason is because the hiding of triangles is not jet performed.

3.3.3.3. Hiding and dividing triangles. The next phase is the application of the hidden surface algorithm. Algorithm traverses projected triangles, in the order of visibility. Projected triangles are stored in a binary sorted tree named PT. Hidden triangles, after they are processed, are saved in a list named HT. In the first step all foremost triangles from PT with the same order of visibility are transferred to HT. This is done because they cannot possibly occlude each other, and further checking is not necessary. After that, every triangle t from PT is checked for the occlusion with triangles (already processed) from HT. If the triangle t is not occluded by any triangle hT from HT, it is moved from PT to HT. Otherwise, if the triangle hT occludes t, they are subtracted. Triangles resulting from this subtraction are added to PT and the loops are restarted.

In the case of occlusion, the loop is restarted several times. In order not to check triangle pairs twice, the information of already checked triangle pairs is recorded, and in further occlusion checks, they are skipped. To avoid overhead, two more tests are performed: if t and hT have the same order of visibility they cannot occlude each other, and the occlusion test is skipped. Also if they are neighbors, since we have already done the backface-culling, they also cannot occlude each other, and the occlusion test is skipped.

The results of the hidden surface algorithm are displayed in Fig. 8f. In this figure, triangles from the previous phase (Fig. 8e), are processed according to their order of visibility. In this case black triangles, which are closer to the source of the beam, are subtracted from the gray triangle. The process gives in 10 triangles, which do not overlap each other, while covering the whole area of the beam section.

3.3.3.4. Creating divided beams. The process of beam division in the BTR terminates with the creation of divided beams. The initial beam that has been divided is removed from the raw beams stack. Divided beams are created and pushed onto the divided beams stack. These beams are created from the 2D triangles that result from the hiding and dividing phase. These hidden 2D triangles are first projected back from the beam space to the world space. Divided beams are then constructed so that their three corner rays originate at the source of the original raw beam and pass through the three corners of hidden triangles.

Figure 8g displays divided beams generated from the triangles in Fig. 8f. Beams are displayed in different shades of gray. One can see only four beams, which occlude the other six beams.

3.3.4. Processing divided beams

The beam division results in several divided beams that reside in the divided beams stack. These beams are now adapted to the geometry of the entity. The next step is to process the interaction of these beams with the shell of the entity (Algorithm 2).

Algorithm 2. The algorithm for processing divided beams.

```
while the divided beams stack (DBS) is not empty
  pop one divided beam (db) from DBS
  close db with the plane of the
       intersecting triangle
  push closed beam on the finished beams
       stack (FBS)
  if termination criteria are not fulfilled
      create one reflected beam from db
      push reflected beam on RBS
      if incoming angle of db is less than
        critical
          create one refracted beam from db
          push reflected beam on RBS
      endif
  endif
repeat
```

The algorithm repeats until all divided beams are processed. In each pass, one divided beam is popped from the divided beams stack. The popped beam is then closed using the plane of the intersecting triangle. The sound intensity at the end of the beam is calculated, and the beam is then pushed on the finished beams stack. The finished beams stack contains beams that are completely processed and ready to be used to generate the results.

The finished beam is then checked to determine whether it meets the termination criteria. Further tracing of a finished beam is not performed if the sound intensity at the end of the finished beam is below a predefined level. The tracing also stops if the beam volume is lower than a predefined volume threshold.

If these criteria are not fulfilled, then tracing continues, and new reflected and refracted raw beams are generated. For each divided beam, one reflected and one refracted raw beam is created. If the incoming angle of the divided beam is greater than the critical angle (Eq. 8), only a reflected beam will be created.

Newly created reflected and refracted beams are put onto the raw beams stack and are subsequently processed. The beam tracing is finished when the raw beams stack is empty and all beams in it have been processed and transferred to the finished beams stack.

3.3.5. Generation of results

After the beam tracing is finished, the generation of results starts. The result of the BTR is the rectangular raster of points with the distribution of level of sound intensity. To calculate the level of sound intensity for each point, simulation has to determine which beams contain the point. The finished beams are organized into an octree to speed up the spatial search algorithm. By using the octree, only the leaf that contains the receiver has to be searched instead of the whole collection of finished beams. Because the number of beams in one leaf of the octree is approximately two orders of magnitude smaller than the total number of beams, the computational complexity of this part of the algorithm is considerably decreased by using the octree structure.

The search of the leaf of the octree results with the list of all beams that contain the receiver. The level of the sound intensity of each beam is then calculated as presented in Subsec. 3.1.1 and summed to get the total level of sound intensity for the desired point in space.

3.4. Implementation

The simulation was coded in Microsoft Visual Studio C++ using Microsoft Foundation Classes for user interfaces and DirectX for displaying 3D graphics. The simulation code had 23 000 lines, including proprietary BSP and octree classes.

Special attention was paid to the numerical robustness of the code. All relational operators were replaced with relational functions that compare values using both absolute and relative margins (Goldberg, 1991). In addition, all geometrical relational functions in the BTR are designed to check boundary situations. For example, the function that checks whether a point is inside a triangle returns not only true or false, but also information about whether the point is collinear with the edge of the triangle and whether the point is collocated with a vertex of the triangle. In addition, these geometric relational functions use the above-mentioned relational functions instead of the usual relational operators of C++.

All arithmetic in the BTR is done in double precision. DirectX forces floating point math by default, but this property is overridden to enforce double precision arithmetic.

4. Results

In this section the performance and the accuracy of the BTR was tested. The BTR was first tested to check if it detects all reflection paths in a rectangular room. Then a comparison with room acoustics simulations was done to check the speed of the BTR. The BTR was compared to two commercial ray tracing simulations and one interactive beam tracing simulation.

Finally, the comparison with the FEM simulation of ultrasound propagation was done to check the ability of the BTR to simulate refraction. The BTR was compared to the FEM simulation, because there are no commercial beam tracing or ray tracing simulation that can simulate refraction.

4.1. Reflection detection test

First test of the BTR was check if it detects all reflection paths in a room. The test room was a simple rectangular room. The Eq. (9) gives the number of reflections for such a room:

$$N_r = \sum_{n=1}^{r} 4\cdot n + 2\sum_{m=r-1}^{0}\left(1+\sum_{n=1}^{m} 4\cdot n\right), \qquad (9)$$

where N_r is number of reflections, and r is the order of reflections. The tests were performed for reflections up to the fifth order. Table 1 and Fig. 9 show the number of reflections detected by the BTR compared to the exact number of reflections calculated with Eq. (9).

Table 1.

r	Nr (exact)	Nr (BTR)	
0	0	0	–
1	6	6	100%
2	24	24	100%
3	62	63	98.39%
4	128	126	98.44%
5	231	237	97.40%

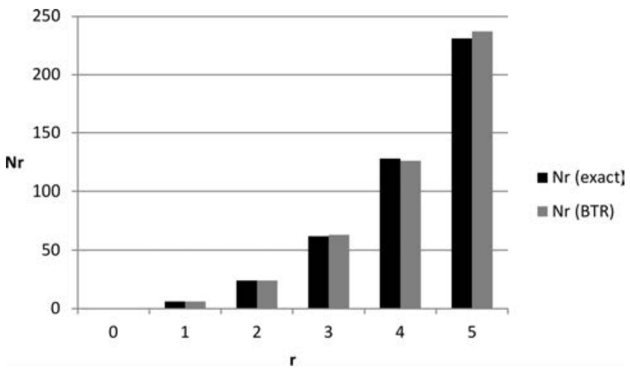


Fig. 9. Results of the reflection detection test.

The reflection detection ratio for the first two orders of reflection is 100%, for the third order is 98.39%, for the fourth order is 98.44%, and for the fifth order is 97.40%.

4.2. Comparison with two ray-tracing room acoustics simulations

BTR's performance and accuracy were compared with two commercial room acoustics simulation programs based on the ray tracing method. These two simulation programs were part of the 3rd round robin on room acoustical simulations (Bork, 2005). The tests were performed on a computer equipped with an Intel Core2 T5600 processor (1.83 GHz), 2 GB of RAM and Microsoft Windows XP operating system.

The tests were based on the three scenes shown in Fig. 10. In this figure, the room geometries are displayed with solid lines and the position of the source is indicated with a small black sphere. The source was unidirectional emitting sound at 1 kHz with a power of

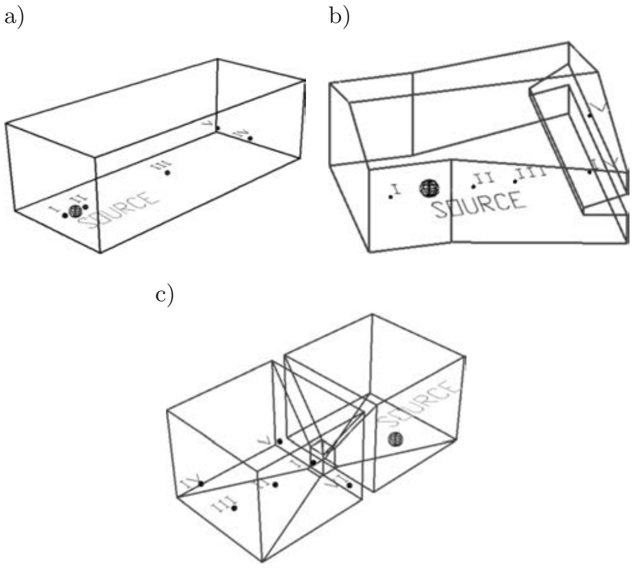


Fig. 10. Test scenes: a) Shoebox, b) Aura, c) Concave.

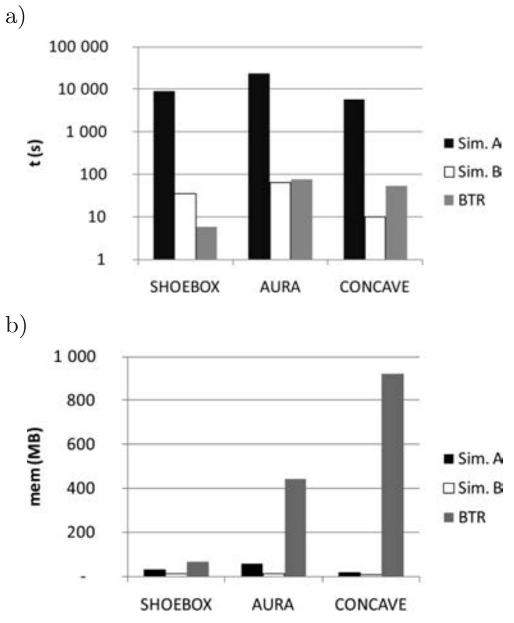


Fig. 11. Wall-clock time (a) and memory consumption (b) in three simulations of the tested scenes.

1 mW. The boundary surfaces had absorption coefficients of 10%. The scattering coefficient was set to zero.

Test scenes were traced with two commercial room acoustic simulations and the BTR. The first ray tracing simulation (Feistel *et al.*, 2007) was named *Simulation A*, and it required that the number of initial rays was set as a parameter before the simulation was executed. To determine the optimal number of initial rays, several simulations were performed with different number of initial rays. The number of initial rays was finally fixed to 10000 because for greater number of rays simulation gave practically the same results, and the calculation time significantly increased.

The second ray tracing simulation (James *et al.*, 2008) was named *Simulation B*, and the number of rays that this simulation traced was automatically determined by the simulation program. In the BTR, the beams were traced until the level of the sound intensity dropped below 40 dB. In all simulations, the tracing was performed for direct sound and three reflections.

The average difference of the level of the sound intensity in control points between the BTR and *Simulation A* was 3.87 dB, and between the BTR and *Simulation B* was 0.87 dB.

Simulation A was the slowest by two orders of magnitude for all three scenes (Fig. 11a). The speeds of the BTR and *Simulation B* were similar. The BTR was faster for the simplest scene (*Shoebox*), and *Simulation B* was faster for the other two scenes.

Simulation B had the lowest memory consumption for all three scenes (Fig. 11b). The BTR had by far the highest memory consumption because the BTR keeps all traced beams permanently in the memory. The other two simulations store the traced rays only while their contributions to the sound field are calculated. The advantage of the method that the BTR uses is that only part of the processing has to be redone when scanning a different part of the room or when scanning with a different resolution. In such situations, the BTR works significantly faster than the other two simulations.

The tests presented in this chapter show that although the BTR is not designed for room acoustics, it has good speed, compared to commercial ray-tracing simulations. The BTR was as fast as the fastest ray tracing simulation. The memory consumption of the BTR was higher than the ray tracing simulations but allowed faster recalculation of the results in cases with fixed scene geometries and fixed source positions. The drawback of the BTR is its exponential space complexity. The BTR has the space complexity $O(n^r)$, where n is the number of triangles in the model, and r is the order of reflections calculated in the simulation. Because of this the BTR cannot be used efficiently for the calculation of later reverberations.

4.3. Comparison with the beam tracing room acoustics simulation

The BTR was then compared to Evertims simulation (Noistering *et al.*, 2008). This is an interactive simulation, which is composed of three components: the graphical interface, the simulation engine and the auralization software. The simulation engine of Evertims simulation is based on the beam tracing method. It calculates the room impulse response, which is subsequently used for the auralization. For each reflection Evertims calculates the direction of arrival. Evertims calculates specular reflections and doesn't take into account the diffraction of sound.

The tests of the BTR and Evertims were performed on the computer equipped with an Intel Core2Duo E6550 processor (2 GHz), 4 GB of RAM and Microsoft Windows 7 operating system.

Evertims and the BTR were tested on the model of Sigyn hall in Turku, Finland (Fig. 12), which was composed of 561 triangles. Both simulations calculated the room impulse response, composed of direct sound and reflections up to the third order. Simulations were performed for the single source of sound. To get the fair comparison, Evertims simulation performed the calculation of the impulse response without the auralization of the sound. On the other hand the BTR didn't calculate the refraction.

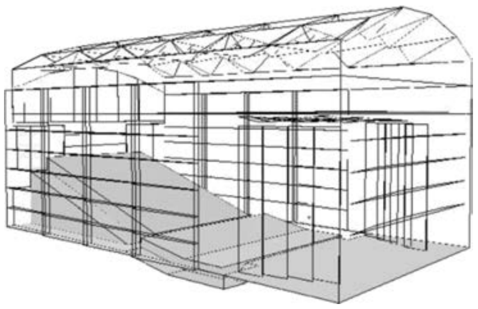


Fig. 12. The model of Sigyn hall in Turku, Finland.

The comparison of performance of BTR and Evertims is presented in Fig. 13. The speed of the BTR was similar to the speed of the Evertims simulation (Fig. 13a). The memory consumption of the BTR is much higher (Fig. 13b) than Evertims. The reason for this is probably the adaptive beam tracing used in the BTR, which results in greater number of beams, and consequently greater memory consumption.

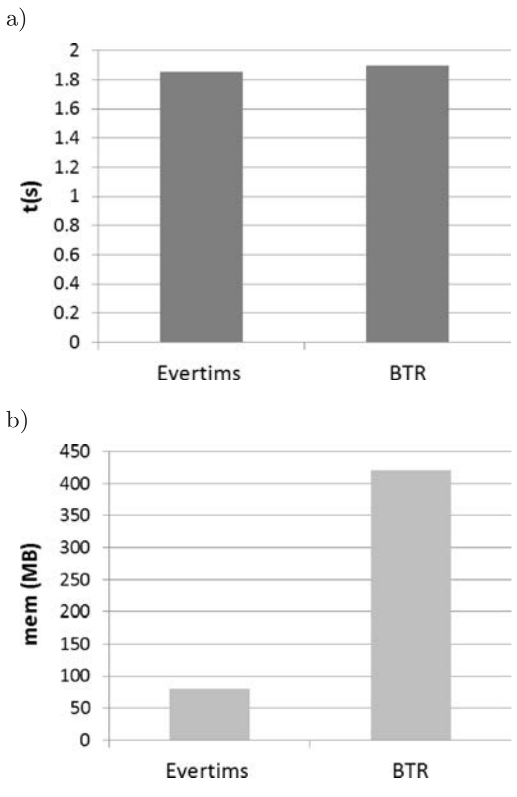


Fig. 13. The speed (a) and the memory consumption (b) of the simulation of Sigyn hall with Evertims and the BTR.

4.4. Comparison with the FEM simulation of ultrasound propagation

The tests presented in the previous chapter have checked the speed of the BTR algorithm. However, the main advantage of the BTR over other beam tracing methods is that it can trace not only reflections but refractions as well. To verify this property of the BTR, a simple scene with an acoustic lens was constructed (Fig. 14).

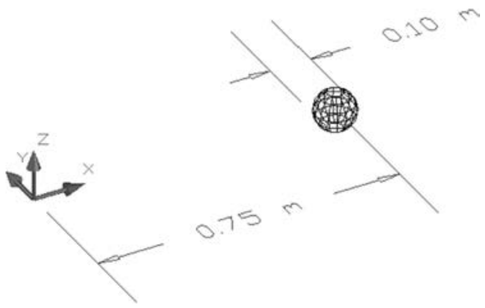


Fig. 14. The scene used to test refraction in the BTR.

The acoustic lens scene consisted of a unidirectional sound source (f = 100 kHz; P = 100 W) emitting ultrasound into a space filled with glycerol ($c = 1920$ m/s; $\rho = 1260$ kg/m^3; $\gamma = 3 \cdot 10^{-6}$ m^{-1}). In the glycerol, there was an entity made of rubber ($c = 900$ m/s; $\rho = 930$ kg/m^3; $\gamma = 43 \cdot 10^{-3}$ m^{-1}). The rubber entity was a sphere with a diameter of 0.1 m centered 0.75 m from the sound source (located at the origin of the coordinate system in Fig. 14 and Fig. 15).

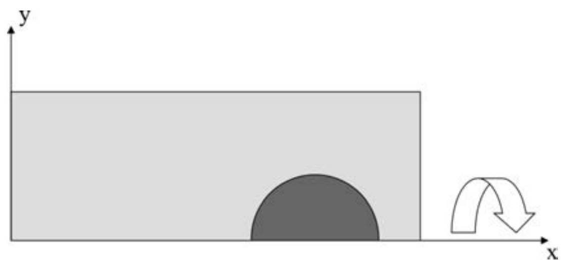


Fig. 15. 2.5D FEM simulation model-rotation around the x-axis.

The acoustic lens scene was first simulated with the BTR and compared with a ray tracing simulation (Sikora *et al.*, 2010). The results of these two methods agreed well. To gain further insight, the BTR was compared with a well-established FEM simulation

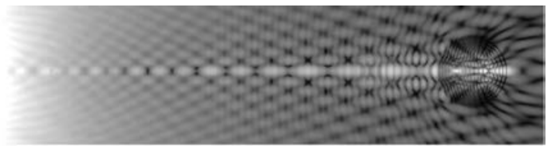
(Wojcik *et al.*, 1998). This simulation takes a finite-element and explicit time-domain approach to solve the propagation of ultrasound.

Because simulating this scene in full 3D with the FEM, would be too computationally complex to perform on a desktop computer, the simulation was performed in 2.5D by utilizing the rotational symmetry of the acoustic lens scene. The rotational axis is the line from the source of the sound (the origin of coordinate system) through the center of the rubber sphere (Fig. 15). In this 2.5D setup, one finite element had dimensions of 0.36×0.36 mm, for a total of $2\,222 \times 833 = 1\,850\,926$ elements.

The BTR scene structure had two entities: the outer, non-convex entity filled with glycerol, and the inner, convex entity filled with rubber. Both entities had only one shell, in the form of a sphere. The geometry of the sphere consisted of 224 equilateral triangles (Fig. 14). The BTR simulation traced direct sound and reflections/refractions up to 4th order.

The results of the FEM and the BTR simulations are shown in Fig. 16a and 16b, respectively. In the left part of figures one can see an area with the high intensity ultrasound surrounding the source. This area is shown in white. The sound intensity level decreases as the sound travels from the source towards the rubber sphere situated in the right part of the model. As sound decreases, the color in figures changes from white to gray. Inside the rubber sphere, the focusing of the ultrasound occurs, as shown in Fig. 16a for the BTR and 16b for the FEM. The areas where the sound focuses are shown as lighter gray areas inside the sphere. In both figures, one can see how the refraction on the boundary between the glycerol and the front of the rubber sphere bends the ultrasound toward the primary focus, which is situated in the back of the sphere. The position and maximum intensity of the primary focus is nearly the same in both simulations. The difference between the locations of the maximum is 0.7 mm, and the difference between the sound intensity levels is 1 dB.

a)



b)

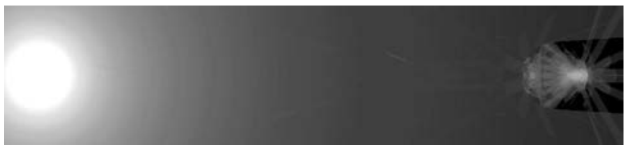


Fig. 16. The simulation of the acoustic lens scene with the FEM (a) and the BTR (b).

In the left part of the sphere, there is also a secondary focus, which occurs because the refracted ultrasound that has entered the sphere reflects two more times inside the sphere. It first reflects from the back of the sphere, and then from the upper of lower boundary of the sphere. Because of multiple reflections/refractions and because of the longer path that the sound traverses after the first refraction, the secondary focus has lower intensity level.

Figures 16 and 17 show two differences between the FEM and the BTR. Since the FEM calculates the pressure of the sound taking the phase into account, its results exhibit the effects of the sound wave interference, which can be seen as the line patterns. Since the BTR calculates the intensity of sound, it doesn't take the phase into account, so results of the BTR don't show the interference lines. Also, the BTR creates a clear shadow region behind the lens, while the FEM simulation does not, because the FEM calculates the diffraction of the sound.

a) b)

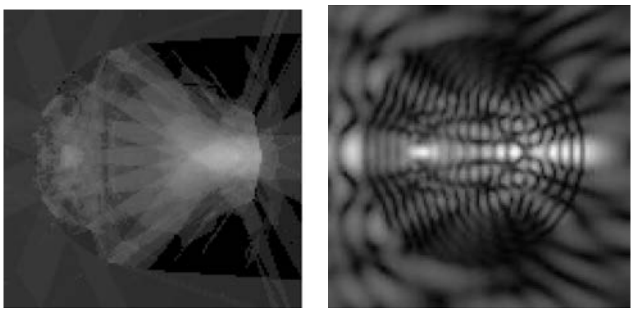


Fig. 17. The distribution of the level of sound intensity for the area of the rubber lens: the BTR (a) and the FEM (b).

Let us consider the performance of the FEM and the BTR simulations. Both simulations were executed on the same hardware platform, which consisted of an Intel Core2Duo processor with a frequency of 2.4 GHz and 4 GB RAM.

Figure 18a shows that the BTR simulation executed approximately 10 times faster than the FEM simulation. The BTR also used less than half of the memory that the FEM used (Fig. 18b). The price for this good performance from the BTR is that the FEM simulation modeled more wave phenomena than the BTR, and that the error in the beam refocusing in the BTR caused certain discrepancies. On the other hand, the FEM could not perform this simulation in full 3D, but only in 2.5D. In addition, given a higher sound frequency, the performance of the BTR would stay the same, while the performance of the FEM would decrease. Because of the required wavelength/finite element size ratio, the number of finite elements would have to be increased, and the performance of the FEM would decrease significantly.

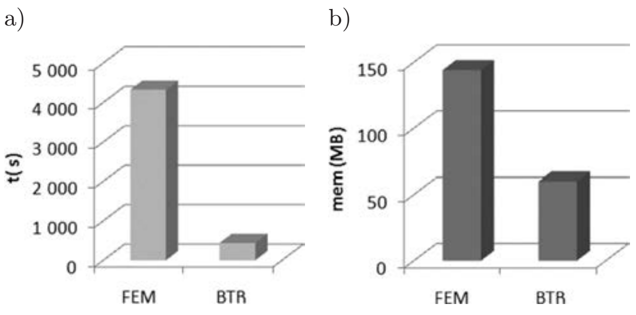


Fig. 18. The speed (a) and the memory consumption (b) of the simulation of acoustic lens with the FEM and the BTR.

This test showed that the BTR is a good choice for situations in which the dimensions of the scene are large compared with the wavelength of ultrasound and the interference and diffraction of sound do not have to be simulated. In such cases, the BTR provides excellent performance and good accuracy.

5. Conclusion and future work

This paper presents the beam tracing with refraction (BTR) method, which is designed for the simulation of sound propagation in environments with more than one media. The BTR method can simulate reflection and refraction of sound.

Although primarily designed for non-homogenous environments such as situations with temperature vertical gradients, the BTR can also be efficiently used for room acoustics. The BTR is suitable for the quick estimation of stationary sound field intensity distribution during the design of sound reinforcement systems. After the beam tracing is done all beams with the level of sound intensity at their beginning are stored as a beam tree structure in the computer memory. That enables the fast recalculation of results for the stationary source and geometry. The BTR is not suitable for the analysis of long delayed reflections of high order, but this limitation can be overcome by combining the BTR with other methods, which is a common practice today – most commercial simulation packages combine the virtual source method for early reflections, the ray tracing for higher order reflections and the statistical reverberation tail creation.

Three tests were performed to check the quality of BTR method: two to determine the speed of the BTR and one to check the ability to simulate the refraction of sound. The speed of BTR was checked against two commercial room acoustic ray-tracing simulations and against an interactive beam tracing simulation. The ability of the BTR to simulate the refraction of sound was checked by the comparison with the commercial FEM simulation. Tests showed that refraction can be simulated with the beam tracing method, and that it can be efficiently used for simple models made of few different media, bearing in mind that diffraction and interference are not simulated. On the other hand, the method of calculating the sound intensity used in the BTR shows to be useful in room acoustics, for quick estimation of stationary sound field distribution, where several recalculations have to be performed for stationary source.

In the future, it would be desirable to incorporate other wave phenomena such as diffraction, interference and diffuse reflections into the BTR to obtain more realistic results. In addition, both beam tracing and generation of the results should be parallelized to increase the performance on the multicore processors that are now common.

Appendix A.

The approximated sound intensity of the receiver I_{BTR} is calculated with Eq. (5), where I_0 is the intensity at the barycenter of the starting triangle of the beam, r_1 is the distance from the virtual source of the beam to the barycenter of the starting triangle, and r_2 is the distance from the virtual source to the receiver (Fig. 3). The exact sound intensity of the receiver I is calculated with Eq. (6), where I_0' is the intensity at the intersection of r_2 and the starting triangle of the beam, and r_1' is the distance from the virtual sound source to the intersection. The relative error of the BTR intensity ΔI is:

$$\Delta I = \frac{I_{\mathrm{BTR}}}{I}. \tag{10}$$

Using Eqs. (5) and (6) Eq. (I) is transformed to:

$$\Delta I = \frac{I_0}{I'} \cdot e^{-\gamma(r_1' - r_1)} \tag{11}$$

From Eqs. (1) and (3) one can get expressions for I_0 and I_0':

$$I_0 = (1 - R^2)\frac{P_A}{4 \cdot \pi \cdot r_1^2} \cdot e^{-\gamma \cdot r_1}, \tag{12}$$

$$I_0' = (1 - R^2)\frac{P_A}{4 \cdot \pi \cdot {r'}_1^2} \cdot e^{-\gamma \cdot r_1'}. \tag{13}$$

Using Eqs. (12) and (13) the ratio of I_0/I_0' is:

$$\frac{I_0}{I_0'} = \frac{{r'}_1^2}{r_1^2} e^{-\gamma(r_1 - r_1')}. \tag{14}$$

Entering Eq. (14) into Eq. (11) one gets the relative error of the BTR intensity ΔI:

$$\Delta I = \frac{{r'}_1^2}{r_1^2}. \tag{15}$$

The relative error expressed as the level of the intensity is:

$$\Delta L_I = 10 \cdot \log \frac{{r'}_1^2}{r_1^2}. \tag{16}$$

From Eq. (16) it is evident that error of the BTR sound intensity level depends only on the difference of the distance between the source and the barycenter and the distance between the source and the real intersection of the sound beam. In the worst case those two distances can be the distances between the source and the closest and the farthest point on the starting triangle of the beam.

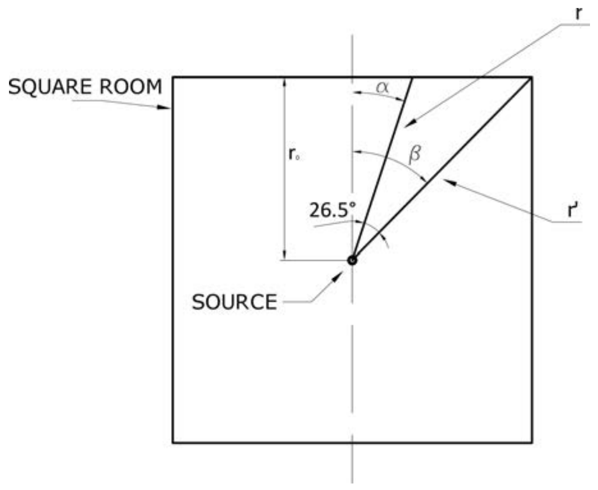


Fig. 19. A square room used to estimate the intensity error.

Let us examine a square room with an initial beam like one on the following Fig. 19. To calculate the maximum error it is enough to consider the 2D case.

The maximum error is for the beam that stretches to the corner of the room. For such beam r and r' can be expressed as:

$$r = \frac{r_0}{\cos(\alpha)}, \tag{17}$$

$$r' = \frac{r_0}{\cos(\beta)}, \tag{18}$$

$$\alpha = \beta - 26.5°, \tag{19}$$

$$\beta = 45°. \tag{20}$$

When Eqs. (17) and (20) are entered in Eq. (16) we get that the maximum error of the BTR sound intensity level is $\Delta L_I = 2.54$ dB.

However we have to stress here that this is the worst case scenario. This error is the maximum error for the initial beam, and the initial beam is the widest possible beam in the BTR. The subsequent beams, because of the beam division, have the smaller angular difference and consequently the error is smaller.

References

1. Alpkocak A., Sis M.K. (2010), *Computing impulse response of room acoustics using the ray-tracing method in time domain*, Archives of Acoustics, **35**, 4, 505–519.
2. Bork I. (2005), *Report on the 3rd Round Robin on Room Acoustical Computer Simulation – Part II: Calculations*, Acta Acustica united with Acustica, **91**, 753–763.
3. Botteldooren D. (1994), *Acoustical ?nite-difference time-domain simulations in a quasi-Cartesian grid*, Journal of Acoustical Society of America, **95**, 2313–2319.
4. James A., Dalenback B.I., Naqvi A. (2008), *Computer Modelling With CATT Acoustics – Theory and Practise of Diffuse Reflection and Array Modeling*, Proceedings of 24th Reproduced Sound Conference, 20–21.11, Brighton, UK.
5. Drumm I. (2000), *The adaptive beam tracing algorithm*, Journal of Acoustical Society of America, **107**, 3, 1405–1412.
6. Farina A. (1994), R*amsete un nuovo software per la previsione del campo sonoro in teatri, ambienti industriali ed ambiente esterno*, Atti del XXII Congresso Nazionale AIA, 13–15 April, Lecce.
7. Farina A. (2000), *Validation of the pyramid tracing algorithm for sound propagation outdoors: comparison with experimental measurements and with the ISO/DIS 9613 standards*, Advances in Engineering Software, **31**, 4, 241–250.
8. Feistel S., Ahnert W., Miron A., Schmitz O. (2007), *Improved methods for calculating room impulse response with EASE 4.2 AURA*, Proceedings of 19th International congress on Acoustics, 2–7 September, Madrid.
9. Fink K. (1994), *Computer Simulation of Pressure Fields Generated by Acoustic Lens Beamformers*, M.Sc. Thesis, University of Washington.
10. Funkhouser T., Carlbom I., Elko G., Pingali G., Sondhi M., West J.A. (1998), *Beam Tracing Approach to Acoustic Modeling for Interactive Virtual Environments*, Proceedings SIGGRAPH 98, pp. 21–32, Orlando.
11. Goldberg D. (1991), *What every computer scientist should know about floating-point arithmetic*, ACM Computing Surveys, **21**, 1, 5–48.
12. Heckbert P.S., Hanrahan P. (1984), *Beam Tracing Polygonal Objects*, Proceedings of the 11th annual conference on Computer graphics and interactive techniques, pp. 119–127, New York.
13. Kleiner M., Dalenback B.I., Svensson P. (1993), *Auralization – An Overview*, Journal of Audio Engineering Society, **41**, 11, 861–875.
14. Laine S., Siltanen S., Lokki T., Savioja L. (2009), *Accelerated beam tracing algorithm*, Applied Acoustics (Elsevier), **70**, 1, 172–181.
15. Lewers T. (1993), *A combined beam tracing and radiant exchange computer model of room acoustics*, Applied Acoustics (Elsevier), **38**, 2, 161–178.
16. Maercke D., Maercke D., Martin J. (1993), *The prediction of echograms and impulse responses within the Epidaure software*, Applied Acoustics (Elsevier), **38**, 2, 93–114.

17. Noisternig M., Katz B.F.G, Siltanen S., Savioja L. (2008), *Framework for Real-Time Auralization in Architectural Acoustics*, Acta Acustica united with Acustica, **94**, 1000–1015.

18. Picaut J., Polack J.-D., Simon L. (1997), *A Mathematical Model of Diffuse Sound Field Based on a Diffusion Equation*, Acta Acustica united with Acustica, **83**, 614–621.

19. Pierce A.D. (1981), *Acoustics – An Introduction to its Physical Principles and Applications*, McGraw-Hill.

20. Shah M., Pattanaik S. (2007), *Caustic Mapping: An Image-space Technique for Real-time Caustics*, IEEE Transactions on Visualization and Computer Graphics, **13**, 272–280.

21. Sikora M., Mateljan I., Bogunović N. (2010), *The effect of refraction as a non-linear transformation on beam tracing simulation of sound propagation*, Proceedings of 1st EAA EUROREGIO Congress on Sound and Vibration – Acta Acustica united with Acustica, **96**, Supp. 1, 62.

22. Siltanen S., Lokki T., Kiminki S., Savioja L. (2007), *The room acoustic rendering equation*, Journal of Acoustical Society of America, **95**, 2313–2319, **122**, 3, 1624–1635.

23. Siltanen S., Lokki T., Savioja L. (2009), *Frequency Domain Acoustic Radiance Transfer for Real-Time Auralization*, Acta Acustica united with Acustica, **95**, 106–117.

24. Sutherland I.E., Hodgman G.W. (1974), *Reentrant polygon clipping*, Communications of the ACM, **17**, 1, pp. 32–42.

25. Stephenson U. (1996), *Quantized Pyramidal Beam Tracing – a new algorithm for room acoustics and noise immission prognosis*, Acta Acustica united with Acustica, **82**, 517–525.

26. Tsingos N., Funkhouser T. (2001), *Modeling Acoustics in Virtual Environments Using 1the Uniform Theory of Diffraction*, Proceedings of ACM SIGGRAPH 2001, pp. 545–552, Los Angeles.

27. Vorlander M. (2008), *Auralization*, Springer.

28. Walsh J.P., Dadoun N., Kirkpatrick D.G. (1985), *The geometry of beam tracing*, Proceedings of the first annual symposium on Computational geometry, pp. 55–61, Toronto.

29. Wojcik G.L., Vaughan D.K., Murray V., Mould J. Jr. (1994), *Time-domain Modeling of Composite Arrays for Underground Imaging*, Proceedings of the IEEE Ultrasonics Symposium, pp. 1027–1032, Cannes.

30. Wojcik G.L., Vaughan D.K., Abboud N., Mould J. Jr. (1998), *Finite Element Modeling for Ultrasonic Transducers*, Proceedings of Ultrasonic transducer engineering Conference, pp. 19–42, San Diego.