THE ALGORITHM OF BEAM DIVISION IN BEAM TRACING METHOD FOR NON-HOMOGENOUS ENVIRONMENTS

Marjan Sikora

Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, Split (sikora@fesb.hr)

Abstract: The methods for the simulation of propagation of sound based on the principles of geometric acoustics have so far been used mostly in homogenous environments. These methods are based primarily on the reflection, and calculate propagation only in homogenous media. The author has previously published a proposed alternative method based on the beam-tracing method that can be used to calculate propagation of sound in non-homogenous environments. The proposed method calculates not only reflection, but also the refraction of the sound waves. Since environment in this method can be of complex shape and composition, it is represented by irregular triangle networks (TIN). This paper presents the algorithm of beam division, which is used when the beam representing sound wave encounters a discontinuum in the environment. The discontinuum generates results in a reflected and a refracted wave. Also because the discontinuums are represented by TIN, in some cases the division of beam into several smaller ones also happens. The process of beam division can be rather complex, and this paper presents the detailed algorithm for this process.

Key words: simulation, beam tracing

1. INTRODUCTION

The simulation of the wave propagation is an important tool used for different purposes such as engineering, design, multimedia applications, and instrument and tool design [3, 4]. The method that author is discussing in this paper is called comprehensive beam tracing method (CBTM). It is designed primarily for acoustic waves, but it can be successfully applied to other kinds of waves, especially optical. The method aims to extend the existing beam tracing method (BTM). BTM is focused primarily on simulating reflection in room acoustics. CBTM would be able to trace not only reflections, but also the refraction. In contrast to current methods, which calculate the propagation in simple, homogenous environments, CBTM would be able to simulate the complex, non-homogenous scene. The most important goal of CBTM is to develop the structure of the scene and adaptive algorithm of tracing beams that would enable the simulation of such complex environments [7].

The basic BTM was developed by Walsh et al. [5] as an enhancement to ray-tracing method. Later in 1993. Lewers used triangular beams (pyramids) instead of cones in order to eliminate cone overlapping, and achieve the spatial coherence. In 1995. Farrina developed Rammsete [2] – a BTM simulation that uses pyramid, triangular beams. By comparing the simulated and measured values he discovered that he cannot get simulation error smaller than certain margin.

Fig.1. a) Simple reflection – upper left corner; b) complex reflection from the several planes – upper right corner; c) complex reflection from the obtrusive surfaces - down
He concluded that the BTM model didn’t take into account the complex cases where beam reflects from several planes (Fig. 1.b and 1.c), and not just one (Fig. 1.a). In this case, in the real life, the sound is reflected from all planes, and BTM made an approximation by using only one reflection beam – the strongest one. This approximation caused an error, and so the adaptive beam tracing algorithm by Drumm [1]. In the case like in Fig. 1.b this algorithm divided single input beam into several reflected beams. This process was done using a descriptor ray – a ray that described the parts of surfaces that were lightened by the input beam. This algorithm was further enhanced by Funkhouser [3].

All these authors specialized their BTM methods to simulate the homogenous environment were sound is propagating in single medium (air) and reflects from flat surfaces (walls) geometrically represented by polygons. The CBTM can simulate the non-homogenous environment, where sound is propagating through several media. It calculates not only reflection but also refraction of sound. Also the surfaces where the reflection and refraction happens are not polygonal, but instead are irregular triangular networks (TIN). Because of all this the author has developed a different beam division algorithm than that used in simulations described above. This algorithm is discussed in further text.

2. CBTM SCENE STRUCTURE

The detailed explanation of CBTM scene structure and algorithm can be found in author’s previous papers [7]. For understanding of beam division algorithm here would be presented only few most important facts. The structure of the scene in CBTM is different from classic BTM, where the scene is composed of reflective surfaces and single medium (Fig. 2.a). In CBTM scene is composed of boundary surfaces and entities (Fig. 2.b). Entities represent the volumes filled with single, homogenous medium.

![Fig. 2. a) Scene structure in BTM](image1)

![Fig. 2. b) scene structure in CBTM](image2)

Two entities are divided by a boundary surface. So, one boundary surface and two entities form a winged edge (WE) structure. This defines the topological structure of the scene. The example of scene is shown in Fig. 2.b. The scene is composed of three boundary surfaces – three winged-edge structures:
- D1 (E3, E2)
- D2 (E3, E1)
- D3 (E1, E2)

It can also be presented with lists representing three entities and the boundary surfaces that surround them:
- E1 [D2, D3]
- E2 [D1, D3]
- E3 [D1, D2]

Thus the scene space is completely topologically defined. The boundary surfaces are represented by TINs. They are composed of triangles, divided by common edges. One edge and two triangles also form a WE structure, which defines the topological structure of boundary surface. This topological structure would be of great use during beam division algorithm – the triangle adjacency information that is saved into the WE structure can greatly speed up the geometrical calculations.

Let’s trace one beam for the scene shown in Fig. 2.b. The beam originates in the source. The source is situated in entity E1, so the original beam also starts spreading in the entity E1. From the entities list we see that this entity is surrounded by two boundary surfaces – D2 and D3. This means that we have to check for hit of only these two boundaries and not all three – a 33% save of processing time. This is the first benefit of our topological links in the scene. In more realistic, complex cases, this would result in much bigger optimization.

In Fig. 2.b we see that one beam hits the D3 boundary surface. From out WE structure we see that D3 boundary surface divides entities E1 and E2. The result is that one part of the beam sound energy would reflect and stay into E1 entity, and the other would be refracted and go to the E2 entity. The angle of reflected beam is the same as the angle of the incident beam. The angle of the refracted beam would be calculated from the Snell’s law.
The whole process is then repeated for reflected and refracted beams respectively.
When the sound beam encounters the boundary surface it is reflected as well as refracted. In simplest case the incoming beam would hit only one triangle of boundary surface. Thus two beams would result: one reflected and one refracted. But most of the time things would not be so simple: the incoming beam would hit more than one triangle, and in complex situations those triangles would sometimes not be only adjacent ones – some triangles would be in front of the others. The way that incoming beam reflects and refracts in general is calculated according to the algorithm of adaptive beam division. This algorithm is the core of the CBTM and will be described in the next chapter.

3. BEAM DIVISION ALGORITHM

The simplest case of beam reflection/refraction is shown in Fig.3. In this case the incoming beam B1 (with triangular cross section – ABC), falls onto a single triangle T1 (points DEF). This triangle is part of the boundary surface D1, which divides entities E1 and E2.

![Fig.3. Simple case of beam reflection/refraction.](image)

Fig.3. Simple case of beam reflection/refraction.

Since the intersection of B1 and D1 is completely contained in the triangle T1, no beam division happens. In such case incoming triangular beam B1 results in two triangular beams: one reflected – B2, and one refracted – B3 (Fig.4).

![Fig.4. One input and two resulting beams.](image)

Second case is more complicated than the first one. It is shown in the Fig.5. In this case the incoming beam B1 (with triangular cross section – ABC), falls on two triangles: triangle T1 (points DEF), and triangle T2 (points DGE). Those two triangles are part of the boundary surface D1, which divides entities E1 and E2.

![Fig.5. Incoming beam falls on two triangles.](image)

The intersection of beam B1 and boundary surface D1 is not in this case completely contained in one triangle, but in two triangles – T1 and T2. So incoming beam B1 has to be divided into several beams. The first triangle ABC will be split using the line segment DE. This would create one triangle - D’BE’, and one trapezoid AD’E’C.

![Fig.6. Splitting the incoming beam.](image)

Since the CBTM uses beams that have triangular cross-section, we have to split the trapezoid in two triangles. This could be done using either diagonal AE’ or CD’. In order to get more uniform division, the criteria for choice is that the area of two divided triangles differs as little as possible. So we would choose the diagonal AE’. Finally, the division gives us three triangular beams:

- \( B_{11} = D'BE' \)
- \( B_{12} = AE'C \)
- \( B_{13} = AD'C \)

For each of these divided beams, two resulting beams will be created: one for reflection, and one for refraction (Fig.7.).

![Fig.7.](image)
Let us consider two more similar examples of beam division.

In this case, shown in Fig. 8, the incoming beam falls on six triangles. When divided by their edges, it is consisted of three triangles and three trapezoids. Trapezoids are then split into triangles, which gives us total of 9 divided beams.

In Fig. 9, input beam falls on 10 triangles. The difference from previous cases is that here incoming beam not only intersects triangles, but also completely contains one triangle (which results in beam B15). In this case incoming beam is divided into 10 polygons: 6 trapezoids and 4 triangles (3 intersected and 1 contained). This results in total of 16 divided beams.

From these examples the first algorithm for beam division can be derived:

1. For each triangle that incoming beam intersects and/or contains
   - If triangle is contained
     - Put complete triangle on the divided polygon stack
   - Else
     - Calculate intersection of incoming beam with triangle
     - Put resulting polygon on the divided polygon stack

2. For each divided polygon on the divided polygon stack
   - If polygon is triangle
     - Create a divided beam from triangle
     - Put it the divided beam stack
   - Else (polygon is a trapezoid)
     - Divide polygon into two triangles using the diagonal (from two possible diagonals, chose one which will give the smaller difference of areas of resulting triangles)
     - Create two divided beams for two resulting triangles
     - Put them on the divided beam stack

3. For each beam from the divided beam stack
   - Create one reflected beam
   - Create one refracted beam

In examples shown in the previous text, all triangles are coplanar, but in the real case it would not be so. Nevertheless the algorithm for beam division stays the same for all cases where normals for all intersected triangles are in the opposite direction of beam direction vector (Fig. 10.). This means that angle between beam direction vector and triangle normal is greater than 90° and smaller than 270°.
**Fig. 10.** Incoming beam is opposite to all triangle normals.

But in general case this is not always true. There are some boundary surface configurations where triangles obscure each other (**Fig. 11.**). In such cases, when triangles that intersect beam are projected to beam coordinate system, they obscure each other. So we have to consider it and make necessary changes to the algorithm.

**Fig. 11.** Some triangle normals are not opposite to incoming beam.

The hiding happens when some normals (from triangles that intersect incoming beam) are not opposite to beam direction vector (normal \( n_{T2} \) in **Fig. 11.**). If this happens than previous algorithm would not give correct results. Such a situation is presented in detail on **Fig. 12.**

**Fig. 12.a.** Left and top view of situation with obscured triangles.

In this example there are three strips of triangles:
- back strip: T1-T5 (triangle normal \( n_{T1} \))
- middle strip: T6-T13 (triangle normal \( n_{T6} \))
- front strip: T14-T18 (triangle normal \( n_{T14} \))

Note that middle and front strip obscure the back strip. This geometry is hit by incoming beam B1 with direction vector \( n_{B1} \). The incoming beam B1 intersects following triangles: T2, T3, T4, T8, T9, T10, T11, T15, T16, and T17. Then the angle between incoming beam direction vector and normal of intersected triangles is checked. Since normal \( n_{T6} \) is not in the opposite direction of incoming beam, four obscuring triangles (from middle strip): T8, T9, T10 and T11 are not valid for beam division. These triangles are thus removed from the list containing triangles for beam division. The list now contains T2, T3, T4, T15, T16 and T17.

Let’s now divide these triangles into polygons using three edges of the incoming beam (**Fig. 13**). This will result in six new points: D’, E’, F’, H’, L’, and M’.

**Fig. 12.b** Perspective view.

**Fig. 13.** Division of triangles with edges of incoming beam.

Using new points we get following division polygons:
- back row:
  - AD’IJ’
  - D’E’J’
  - E’BM’J
- front row:
  o F’GL’
  o GM’CL’
  o GH’M’

**Fig. 14.** Dividing polygons: left – without correction, right – with correction.

In **Fig. 14.** left dividing polygons are shown in perspective. It is clear that front row polygons are obscuring back row polygons. Now we have to use painter’s algorithm [8] to remove hidden parts of back row polygons. What is left is shown on **Fig. 14 right** and **Fig. 15.**

**Fig. 15.** By dividing trapezoids after hiding, we get triangles for beam division.

There are two more points created by the process of surface hiding (painter’s algorithm): D’’ and E’’. The front row is unchanged, but in the back row, because of hiding there are now three trapezoids:
- back row:
  o AD’D’’F’
  o D’E’E’’D’’
  o E’BH’E’’

Now we have total of four trapezoids and two triangles. After splitting trapezoids, there are 10 dividing triangles left. Those triangles are now used to make divided beams. Finally, it is possible to define the corrected algorithm for beam division:
- for each triangle that incoming beam intersects and/or contains
  - if angle between direction vector of incoming beam and triangle normal is greater than 90° and smaller than 270° than
    • if triangle is contained
      • put complete triangle on the divided polygon stack
    • else
      • calculate intersection of incoming beam with triangle
      • put resulting polygon on the divided polygon stack
- for all divided polygons on the stack do
  - painter’s algorithm (surface hiding)
    • if divided polygon is completely hidden
      • remove it from the divided polygon stack
    • else if divided polygon is partially obscured
      • remove the part of the polygon that is hidden
      • according to this correct the divided polygon geometry on the divided polygon stack
- for each divided polygon on the divided polygon stack
  • if polygon is triangle
    • create a divided beam from this triangle
    • put it on the divided beam stack
  • else (polygon is trapezoid)
    • divide polygon into two triangles using the diagonal (from two possible diagonals, chose one which will give the smaller difference of areas of resulting triangles)
    • create two divided beams for two resulting triangles
    • put them on the divided beam stack
- for each beam from the divided beam stack
  • create one reflected beam
  • create one refracted beam

**Fig. 16.** shows the perspective view of the results of this algorithm. This figure shows the scene from **Fig. 13** (with difference that incoming beam is not orthogonal to boundary surface).
Fig.16. Perspective view of results of beam division algorithm.

3. CONCLUSION

This paper presents the beam division algorithm used in the comprehensive beam tracing method (CBTM). This algorithm is specially designed for the structure of the scene used in CBTM. It covers simple and complex situations that can occur during beam tracing. The algorithm and the scene structure in CBTM, are the core advantages of this method. Because of them, CBTM can be used to simulate non-homogenous environments, which classical BTM cannot.

However, algorithm still has to be thoroughly checked for degenerate cases such as infinitely thin beams, boundary angles etc. Also the numerical robustness of algorithm is an issue which has to be tested. Regarding the performance, in future the simple painter’s algorithm for hidden surface removal can be replaced with more efficient one.

REFERENCES