Accelerated Beam Tracing Algorithm

Samuli Laine [a] Samuel Siltanen [a] Tapio Lokki [a] Lauri Savioja [a]

[a] *Telecommunications Software and Multimedia Laboratory, Helsinki University of Technology, Espoo, Finland*

Abstract

Determining early specular reflection paths is essential for room acoustics modeling. Beam tracing algorithms have been used to calculate these paths efficiently, thus allowing modeling of acoustics in real-time with a moving listener in simple, or complex but densely occluded, environments with a stationary sound source. In this paper it is shown that beam tracing algorithms can still be optimized by utilizing the spatial coherence in path validation with a moving listener. Since the precalculations required for the presented technique are relatively fast, the acoustic reflection paths can be calculated even for a moving source in simple cases. Simulations were performed to show how the accelerated algorithm compares with the basic algorithm with varying scene complexity and occlusion. Up to two orders of magnitude speed-up was achieved.

Key words: beam tracing
PACS: 43.55.Ka, 43.58.Ta

1 Introduction

Modeling room acoustics at interactive update rates is a great challenge in virtual reality applications. The most essential task is to find all perceptually relevant reflection paths, which contribute to the impulse response of the room. Usually, the early part of the response is considered to be significant, and it is typically characterized by strong specular reflections. Thus it is necessary that their reflection paths are computed correctly and efficiently.

Email addresses: `Samuli.Laine@tml.hut.fi` (Samuli Laine), `Samuel.Siltanen@tml.hut.fi` (Samuel Siltanen), `Tapio.Lokki@tkk.fi` (Tapio Lokki), `Lauri.Savioja@tkk.fi` (Lauri Savioja).

The image source method [1,2] has been used to accurately calculate the specular reflections in a polyhedral environment. Unfortunately, as the order of reflections increases, the number of modeled image sources grows exponentially, although a large portion of them is invisible. This approach is relatively simple to implement, but it involves futile work especially in densely occluded environments. Thus, the image source method is applicable only with simple geometry or with very low order of reflections, if the goal is to achieve interactive modeling of room acoustics.

Although the ray tracing methods used in room acoustics modeling are efficient [3], they produce systematic errors due to the finite number of rays and the size of the receiver [6]. Thus, some valid early specular reflections might be omitted while some invalid reflections would be detected, if no precaution were taken to prevent that. This basic approach, enhanced with some advanced techniques, is commonly used in commercial room acoustics prediction software [4,5]. However, the calculation times are too long for interactive modeling.

The beam tracing technique is an optimal solution to the problem of finding the early specular reflections, because it produces exactly the same results as the image source method, but optimizes the visibility calculation so that only image sources which are part of a valid reflection path are considered [19].

It is still possible to accelerate the beam tracing algorithm for a moving listener without affecting its accuracy. In this paper two optimizations are presented for this purpose. Additionally, unlike in the previous beam tracing algorithms, the precalculation phase is relatively fast, which allows acoustics modeling with a moving sound source or even dynamic geometry in simple cases. When attached to an efficient sound renderer, interactive walkthroughs in complex acoustic environments are made possible [7]. The optimization could also be used to accelerate the calculation of the reflection paths for visualization in acoustics modeling software [4,5].

2 Related Work

The beam tracing technique was first introduced in computer graphics to utilize the spatial coherence in generating realistic images [8]. Since then this technique has not been widely utilized in image synthesis due to its limitations. However, in visibility and occlusion calculations beam tracing algorithms have been proved to be useful [11,12,14,15]. Yet another area in computer graphics where beam tracing has been used is calculating light–water interaction [10].

On the other hand, beam tracing is a more generic algorithm in computational geometry [9,20] and it has been successfully applied in calculating very different phenomena, such as radio wave propagation [16,18] and acoustic reflection paths [13,17,19,21–23,25,29].

The most interesting application area from our point of view is real-time acoustics modeling of virtual environments [21]. Beam tracing is currently the fastest of the commonly used geometric room acoustics modeling techniques [30,27] which still produce accurate results [28].

3 Algorithm

In this paper an efficient beam tracing algorithm, solving the specular reflection paths from a source to a receiver, is presented. As in the previously presented beam tracing techniques, the algorithm consists of a precalculation phase and a run-time algorithm. In the precalculation phase the beam tree is constructed, and this structure is utilized in the run-time phase to find the reflection paths from the source to a moving listener in real-time. The precalculation has to be repeated whenever the geometry changes or the sound source moves. In complex cases this can take a long time, but with the presented technique the calculation can be repeated often enough to model a moving source with simple room models. The precalculation phase does not do accurate visibility calculations and is thus more efficient compared to other acoustic beam tracing algorithms [21]. This is justified by the presented optimizations which allow the run-time algorithm to eventually skip most of the visibility calculations.

3.1 Precalculation

In the precalculation phase the polygonal geometry is first inserted into a binary space partitioning (BSP) tree [31]. The polygons are assumed to be convex, and concave polygons can be clipped into convex parts. In the presented implementation the split planes in the tree are chosen to be axially aligned and the structure resembles a kd–tree [32]. However, the split plane orientations do not rotate regularly, but the orientations are determined by the surface area heuristic to create a balanced tree [34]. This is the acceleration structure used in calculations involving the room geometry, and it allows performing most of the geometric operations in sublinear time instead of linear time. Similar tree structures are commonly used in computer graphics when handling complex geometry [33].

The main goal of the precalculation phase is to create a beam tree. This implementation is similar to the other beam tracing algorithms [19,21,22,25,29], but there are some differences which allow the presented optimization techniques to be utilized. Thus, the implementation is briefly covered here.

The beam tree is a tree structure where the root node corresponds to the sound source. Each child of the root node is a beam defined by the sound source and a polygon. Consequently, there are as many children at the first level as there are polygons in the scene. Occlusion is not taken into account in this structure since the benefits of accurate visibility calculations would be outweighed by the complexity of the resulting structure in most cases. In addition, postponing the visibility calculations to the run-time algorithm speeds up the precalculation phase, and, eventually, the majority of the visibility checks are never performed since the presented optimizations prune the calculation earlier in the beam tree.

Each beam is further reflected with the plane of their defining polygon such that the new beam is defined by the image source of that plane and the polygon. The polygons intersecting the reflected beam are determined and new beams are defined by the current image source and the intersecting polygons. If a polygon is only partially within the beam, it is clipped against the beam, and the clipped polygon is used in the beam tree construction instead of the original one. These beams form the next level of the beam tree. Again, the occlusion testing is not necessary. Figure 1 illustrates the process of creating beams. The beams are reflected and the child nodes are created up to the desired depth which corresponds to the maximum order of reflections in the image source method.

Calculating this structure is a relatively efficient operation since the BSP allows fast intersection testing, and accurate cutting or pruning of the beams due to occlusion tests is not necessary in this phase. A conservative approximation is sufficient or even better than the accurate structure since the branching factor of the tree would increase if the polygons were split often. While the order of reflections increases the beams become narrower thus intersecting less polygons. This is why the branching factor of the beam tree decreases in relation to the depth, and the problem of exponential growth is not nearly as bad as in the image source method.

To save memory only the polygon IDs defining the beams are stored per node. Other information can be retrieved during the run-time calculation.

3.2 Run-time

The goal of the run-time part of the algorithm is to extract quickly the information of the potential reflection paths and to validate them. This process yields the accurate specular reflection paths up to the desired order of reflections. For a static listener the basic algorithm is sufficient, but for a moving listener the process must be accelerated. Thus, two novel optimization techniques are introduced, i.e. fail plane and skip sphere optimizations.

3.2.1 Basic algorithm

The first step is to find the potential reflection paths for the listener. Because only the polygon IDs are saved in the beam tree, there is not enough information to determine in which beams the listener is located. Thus, initially all leaf nodes in the beam tree correspond to the potential reflection paths. The tree must be traversed to validate them and thus collect the actual reflection paths.

To validate a reflection path, the image sources are reconstructed by reflecting the source with each polygon in the path. The reconstruction is done while traversing the beam tree to avoid calculating the same image sources several times for different paths. In the path validity test the tree is traversed upwards while testing each path segment for intersections. The first segment is the path from the listener to the image source of the current leaf node polygon. The next segment is from the intersection point of the previous segment and the current beam tree polygon to the image source of the parent beam tree node polygon. The beam tree is traversed upwards until the root is reached. Thus, the final segment is from the last intersection point to the source. This process corresponds to the one used to validate the paths in the image source method. Figure 2 illustrates the validation process.

The actual test is performed by an efficient ray tracer which utilizes the BSP acceleration structure. There are two criteria a valid segment of the path must fulfill. The first one is that the ray from the current point to the current image source must intersect the beam polygon. The second one is that there must not be occluders in the path from the current point to the intersection point on the polygon. The first one is a trivial geometrical test independent of the occluding geometry, while the second one requires a ray cast which is performed only if the first test is passed. As soon as an invalid segment is found the whole path is considered invalid and no further testing is required. If the path is intersecting all the beam tree polygons up to the root level, and there are no occluders in the path, then the reflection path is valid, and it can be added to the solution.

3.2.2 Fail-plane optimization

The basic algorithm described above can be accelerated significantly for a moving listener. Testing whether a segment of the path intersects a polygon can be seen as testing whether the starting point of the segment is inside the beam defined by the current image source and beam tree polygon. Another formulation for the convex beam is a collection of planes which border the beam volume. The planes are defined by the image source and edges of the beam tree polygon. The test can be done by comparing on which side of the planes the starting point of the segment is. This requires only one dot product per plane. When a test against such a plane fails, the whole path is considered invalid. In addition, the information acquired in this test can be utilized in the subsequent tests for this beam tree node and for the whole path down to the tree rooted from there. The plane against which the test failed, or the *fail plane*, is *propagated* down the path by reflecting it with the polygon planes and stored in the beam tree node. When the listener moves, it is enough to test the listener location against the stored fail plane to know if the same point vs. plane test in the same validation step would fail again. Figure 3 shows the fail plane of a second order reflection. This optimization tends to give the negative result early for invalid paths.

There are two kinds of fail planes. In addition to the beam planes, the planes of the beam tree polygons themselves can become fail planes. This happens when the segment is entirely on one side of that plane, as shown in Fig. 4. It is better to test this before testing against the beam planes, so that in the case of a negative result, creating the beam becomes unnecessary. The plane of the polygon is similarly propagated down the path in the beam tree as a fail plane.

Transforming the segment intersection tests into beam plane tests requires still an additional test phase to validate the path for occlusion. This can be done by an efficient ray tracer as in the basic algorithm. The number of ray tracer calls is far reduced by the plane tests.

3.2.3 Skip sphere optimization

Another optimization method is grouping the potential reflection path nodes in *buckets*. A small number of neighboring BSP-nodes is placed into a bucket. The first time, every node in the bucket goes through the validation test. In the case all nodes fail, it is easy to calculate the distances from the listener to the propagated fail planes. Both the smallest distance and the current listener position are stored in the bucket. These values define the radius and the center of a *skip sphere*. It is guaranteed that if the listener stays inside the skip sphere the paths in the bucket remain invalid. Thus, the test against a number of planes in several paths is replaced by a very quick test against a sphere. In Fig. 5 the skip sphere corresponding to a fail plane is shown. When the listener moves out of the skip sphere the situation is updated. At least one path in the bucket has become potentially valid, and all the nodes must be tested as in an unfailed bucket.

Choosing a proper size for the bucket is crucial. With a small bucket the gain is smaller, because there are less paths per bucket which can be skipped when the test against the skip sphere fails. On the other hand, if the bucket is larger it is more unlikely that all paths in the bucket are invalid. In the presented implementation bucket size 16 proved to be the most efficient with the test models introduced in the results section.

Both of these optimizations of the basic beam tracing algorithm utilize the concept of a fail plane. Propagating the fail planes down the beam tree cuts the tests for the invalid paths early. An efficient ray tracer is still needed to validate the paths for occlusion, but even then the number of tests is reduced by these optimization techniques.

4 Results

Both the precalculation phase and the run-time algorithm were tested and their update times were recorded with six different room models ranging from a cube to a complex concert hall and auditorium models. The models can be seen in Fig. 6. All the tests were run on a computer with an Intel P4 2.8 GHz processor and 1 Gb RAM.

The times taken by the precalculation phase are given in Table 1. It can be noticed that with simple models and low order of reflections the precalculation can be done at interactive rates, which allows also moving sound sources and changing geometry. By interactive rates, an update rate of about 30 Hz of reflection parameters is meant [35]. However, the precalculation time rises rapidly when the order of reflections grows. It can be seen that the precalculation time does not depend only on the number of polygons and the reflection order since there is a significant difference between the concert hall model and the auditorium model, although their polygon counts do not differ much. Closer examination of the models shows that there are some very densely occluded parts in the auditorium model which makes the beam tree more complicated than the one constructed using the concert hall model.

The run-time algorithm was tested with and without the optimizations using third order reflections. An exception is the most complex model in which case only second order reflections were used. The source was stationary while the listener was moving along a predefined path. The performance results can be seen in Table 2.

On average, the fail plane optimization makes the algorithm approximately 40 times faster than the unoptimized version. In addition, the skip spheres still cut the calculation time by a factor of 50 percent. The results show that a model consisting of 1190 polygons can be simulated for real-time auralization upto third order at an interactive rate [35].

The reflection paths produced by the optimized versions of the algorithm were compared to those computed by the unoptimized algorithm. The paths were identical and thus it is safe to conclude that the quality of the results is not affected by the optimizations.

5 Possible Extensions and Future Work

There are also other possibilities for optimization. Although the occlusion calculation is postponed in the run-time algorithm as long as possible, they could be utilized in the precalculation phase to conservatively prune the beam tree [19], since in densely occluded models most polygons are not visible to others at all. On the other hand, this would increase the precalculation time, and modeling with a moving source would not be possible anymore. The target application determines whether pruning the beam tree is tolerated or not.

Currently the algorithm supports only specular reflections, which are important in the early part of the response. It would be possible to extend the algorithm to handle diffraction [25]. A beam can be defined by a diffracting edge and a polygon. Such a beam is still convex and can be represented by a collection of planes similarly to a beam in the case of a specular reflection. This representation is sufficient for using the fail plane and skip sphere optimizations. Only in the validation phase more effort is required since the actual reflection points on the diffracting edges have to be calculated. An iterative approach using, e.g. the Newton method, is probably efficient enough.

6 Conclusions

An optimized beam tracing algorithm for finding and efficiently updating specular reflection paths for a moving listener is presented. This algorithm performs well even with complex, lightly occluded room models. It is shown that even a moving sound source can be modeled at interactive rates with moderate model complexity. The proposed optimizations are based on the concept of a propagated fail plane. Utilizing the optimization techniques allows returning the negative results for validated paths quickly. Because most of the results are negative, this speeds up the computation significantly. The bucketing optimization utilizes the local coherence in the validation results. Together these optimizations can give a two-orders-of-magnitude speed-up compared to the unoptimized algorithm.

Acknowledgments

T. Lokki's work was supported by the Academy of Finland (#119092).

References

[1] Allen JB, Berkley DA. Image method for efficiently simulating small-room acoustics. Journal of the Acoustical Society of America 1979;65(4):943-950.

[2] Borish J. Extension of the image model to arbitrary polyhedra. Journal of the Acoustical Society of Americca 1984;75(6):1827-1836.

[3] Krokstad A, Strom S, Sorsdal S. Calculating the acoustical room response by the use of a ray tracing technique. Journal of Sound and Vibration 1968;8(1):118-125.

[4] Naylor GM. ODEON - Another hybrid room acoustical model. Applied Acoustics 1993;38:131-143.

[5] Dalenbäck BI. Room acoustic prediction based on a unified treatment of diffuse and specular reflection. Journal of the Acoustical Society of America 1996;100(2):899-909.

[6] Lehnert H. Systematic errors of the ray-tracing algorithm. Applied Acoustics 1993;38(2-4):207-221.

[7] Savioja L, Huopaniemi J, Lokki T, Väänänen R. Creating interactive virtual acoustic environments. Journal of the Audio Engineering Society 1999;47:675-705.

[8] Heckbert PS, Hanrahan P. Beam tracing polygonal objects. ACM SIGGRAPH Computer Graphics 1984;18(3):119-127.

[9] Dadoun N, Kirkpatrick DG, Walsh JP. The geometry of beam tracing. Proceedings of the first annual symposium on Computational geometry 1985:55-61.

[10] Watt M. Light-water interaction using backward beam tracing. ACM SIGGRAPH Computer Graphics 1990;24(4):377-385.

[11] Teller SJ. Visibility computations in densely occluded polyhedral environments. Ph.D. thesis, University of California at Berkeley, 1992.

[12] Teller SJ. Computing antipenumbra of an area light source. ACM SIGGRAPH Computer Graphics 1992;26(2):139-148.

[13] Lewers T. A combined beam tracing and radiant exchange computer model of room acoustics. Applied Acoustics 1993;38(2-4):161-178.

[14] Teller SJ, Hanrahan P. Global visibility algorithms for illumination computations. Proceedings of the 20th annual conference on Computer graphics and interactive techniques 1993:239-246.

[15] Luebke D, Georges C. Portals and mirrors: simple, fast evaluation of potentially visible sets. Proceedings of the 1995 symposium on Interactive 3D graphics 1995:105-106.

[16] Fortune S. A beam-tracing algorithm for prediction of indoor radio propagation. Selected papers from the Workshop on Applied Computational Geometry, Towards Geometric Engineering 1996:157-166.

[17] Monks MC, Oh BM, Dorsey J. Acoustic simulation and visualization using a new unified beam tracing and image source apporach. 101th Convention of the Audio Engineering Society 1996, preprint 4335.

[18] Rajkumar A, Naylor BF, Feisullin F, Rogers L. Predicting RF coverage in large environments using ray-beam tracing and partitioning tree representated geometry. Wireless Networks 1996;2(2):143-154.

[19] Funkhouser TA, Carlbom I, Elko G, Pingali G, Sondhi M, West JE. A beam tracing approach to acoustic modeling for interactive virtual environments. Proceedings of the 25th annual conference on Computer graphics and interactive techniques 1998:21-32.

[20] Fortune S. Topological beam tracing. Proceedings of the fifteenth annual symposium on Computational geometry 1999:59-68.

[21] Funkhouser TA, Min P, Carlbom I. Real-time acoustic modeling for distributed virtual environments. Proceedings of the 26th annual conference on Computer graphics and interactive techniques 1999:365-374.

[22] Funkhouser TA. A visibility algorithm for hybrid geometry- and image-based modeling and rendering. Computers & Graphics 1999;23(5):719-728.

[23] Drumm IA, Lam YW. The adaptive beam tracing algorithm. Journal of the Acoustical Society of America 2000;107(3):1405-1412.

[24] Min P, Funkhouser TA. Priority-driven acoustic modeling for virtual environments. Proceedings of EUROGRAPHICS'2000 2000.

[25] Tsingos N, Funkhouser TA, Ngan A, Carlbom I. Modeling acoustics in virtual environments using the uniform theory of diffraction. Proceedings of the 28th annual conference on Computer graphics and interactive techniques 2001:545-552.

[26] Funkhouser TA, Tsingos N, Carlbom I, Elko G, Sondhi M, West JE. Modeling sound reflection and diffraction in architectural environments with beam tracing. Forum Acusticum 2002.

[27] Svensson UP, Kristianssen UR. Computational modeling and simulation of acoustic spaces. AES 22nd international conference on Virtual, synthetic and entertainment audio, Espoo, Finland, June 15-17 2002:11-30.

[28] Tsingos N, Carlbom I, Kubli R, Funkhouser TA. Validating acoustical simulations in the Bell Labs Box. Computers & Graphics 2002;22(4):28-37.

[29] Funkhouser TA, Tsingos N, Carlbom I, Elko G, Sondhi M, West JE, Pingali G, Min P, Ngan A. A beam tracing method for interactive architectural acoustics. Journal of the Acoustical Society of America 2004;115(2):739-756.

[30] Funkhouser TA, Tsingos N, Jot JM. Sounds good to me! Computational sooound for graphics, VR, and interactive systems. SIGGRAPH 2002 Course Notes.

[31] Fuchs H, Kedem ZM, Naylor BF. On visible surface generation by a priori tree structures. ACM SIGGRAPH Computer Graphics 1980;14(3):124-133.

[32] Bentley JL. Multidimensional binary search trees used for associative searching. Communications of the ACM 1975;18(9):507-517.

[33] Samet H. The design and analysis of spatial data structures. Addison-Wesley, 1990.

[34] Havran V. Heuristic ray shooting algorithms. Ph.D. thesis, Czech Technical University in Prague, 2000.

[35] Lokki T, Savioja L, Huopaniemi J, Väänänen R, Takala T. Creating interactive virtual auditory environments. IEEE Computer Graphics and Applications 2002;22(4):49-57.

Table 1
For each of the six test models, the precalculation was done for 1st–6th order reflections. The beam tree took too much memory with higher order reflections in some models, which is why the results are not given in such cases.

		Order					
Model	Polygons	1st	2nd	3rd	4th	5th	6th
		(s)	(s)	(s)	(s)	(s)	(s)
Cube	6	0.0001	0.0005	0.0022	0.010	0.030	0.0525
Simple Room	438	0.0033	0.130	1.57	12.41	85.99	449
Regular Room	1190	0.0094	0.950	20.93	294	-	-
Complex Room	5635	0.047	4.96	77.0	549	2824	-
Concert Hall	12115	0.130	12.5	325	-	-	-
Auditorium	14472	0.143	91.2	-	-	-	-

Table 2
For each test model the third order reflections were calculated, except for the auditorium model for which only second order refletions could be calculated, while the listener was moving along a predefined path around the source. The average number of reflections as well as the calculation time for the algorithm are given without optimizations (NO), with fail plane (FP) optimization only, and with both the fail plane optimization and skip spheres (FP + SS). The values given in parentheses are speed-up factors compared to the version without the optimization.

Model	Polygons	Reflections	NO	FP	FP + SS
			(s)	(s)	(s)
Cube	6	63.0	0.0014	0.0008 (1.75)	0.0008 (1.75)
Simple Room	438	45.5	0.358	0.009 (39.8)	0.004 (59.7)
Regular Room	1190	34.1	3.03	0.057 (53.2)	0.019 (160)
Complex Room	5635	28.5	5.29	0.114 (46.4)	0.048 (110)
Concert Hall	12115	35.8	38.2	0.943 (40.5)	0.489 (78.2)
Auditorium	14472	8.78	21.1	0.475 (44.4)	0.229 (91.9)

List of Figures

3 *The path segment validity test concerning polygon $P2$ has passed, and the next segment to be tested is the one from the intersection point X on polygon $P2$ towards image source $IS(1)$. The test is performed by testing point X against the planes defining beam B'. Since the point is on the wrong side of plane FP, the test fails, and FP becomes the fail plane. The fail plane is propagated down the beam tree by mirroring it against the plane of polygon $P2$ which yields plane FP'. When the listener is updated the next time, the testing begins by comparing the listener position against plane FP', and while it is on the wrong side, the rest of the tests can be skipped. The wrong sides of FP and FP' are indicated by the arrows normal to the fail planes.* 17

4 *If the segment to be tested is entirely on one side of the plane of the polygon defining the beam, there cannot be a reflection. The segment from intersection point X towards image source $IS(1)$ is entirely behind the plane of polygon $P1$. Thus, that plane FP becomes the fail plane and it is not necessary to construct the planes for beam B' for further tests. The propagated fail plane around the plane of polygon $P2$ is not shown since it is the same as the original fail plane in this particular case.* 18

5 *Assume that in this case all the paths (not shown in the figure) in the bucket have returned negative results and that the corresponding fail planes have been propagated (dashed lines). Hence the distances from the listener to all the fail planes are calculated and the shortest one is chosen. Here it is the distance to fail plane FP'. By using that distance as the radius and the current listener position L as the center, the skip sphere SS can be created for the bucket. While the listener stays inside SS the whole bucket of paths can be skipped.* 19

6 *The models used in performance tests: a) box, b) simple room, c) regular room, d) complex room, e) concert hall, and f) auditorium.* 20

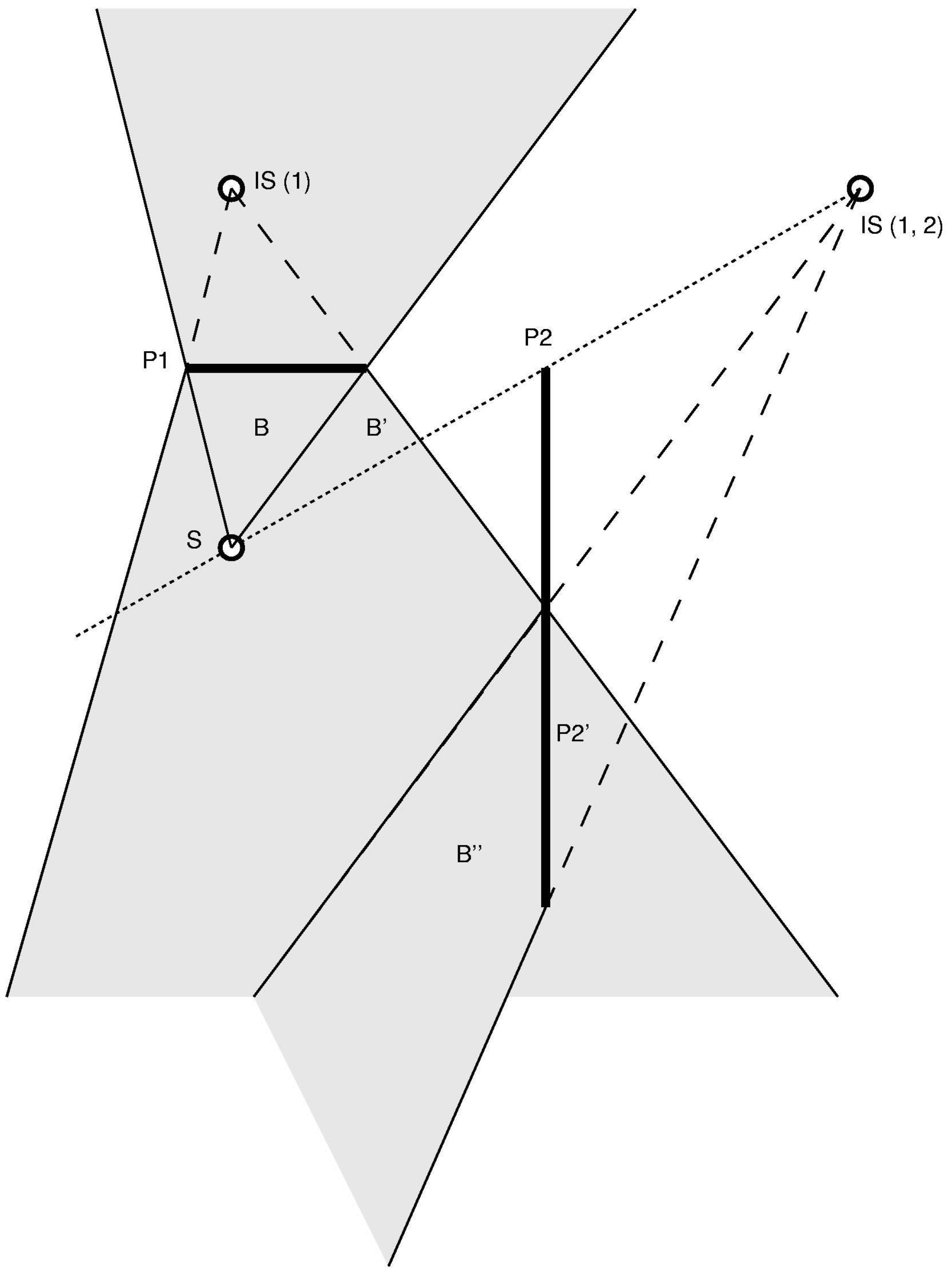

Fig. 1. *Source S and polygon $P1$ define the root level beam B which is not explicitly stored in the beam tree since it can be easily reconstructed whenever needed. Source S is mirrored at a plane of polygon $P1$ to create the first level image source $IS(1)$. $IS(1)$ and $P1$ define the first level beam B' which is stored as a child node of the root in the beam tree. Only the ID of $P1$ is stored since $IS(1)$ can be reconstructed at run-time. Beam B' intersects polygon $P2$, which leads to mirroring $IS(1)$ at the plane of polygon $P2$ yielding the second level image source $IS(1,2)$. In addition, polygon $P2$ is clipped by beam B'. $P2'$ is the clipped polygon inside the beam. $IS(1,2)$ and polygon $P2'$ define a second level beam B'' which becomes a child node of the node defining B'. Again, only the ID of the polygon $P2$ needs to be stored. Note that, since beam B' intersects polygon $P2$ only partially, beam B'' is clipped accordingly, the dotted line showing what would be the unclipped beam.*

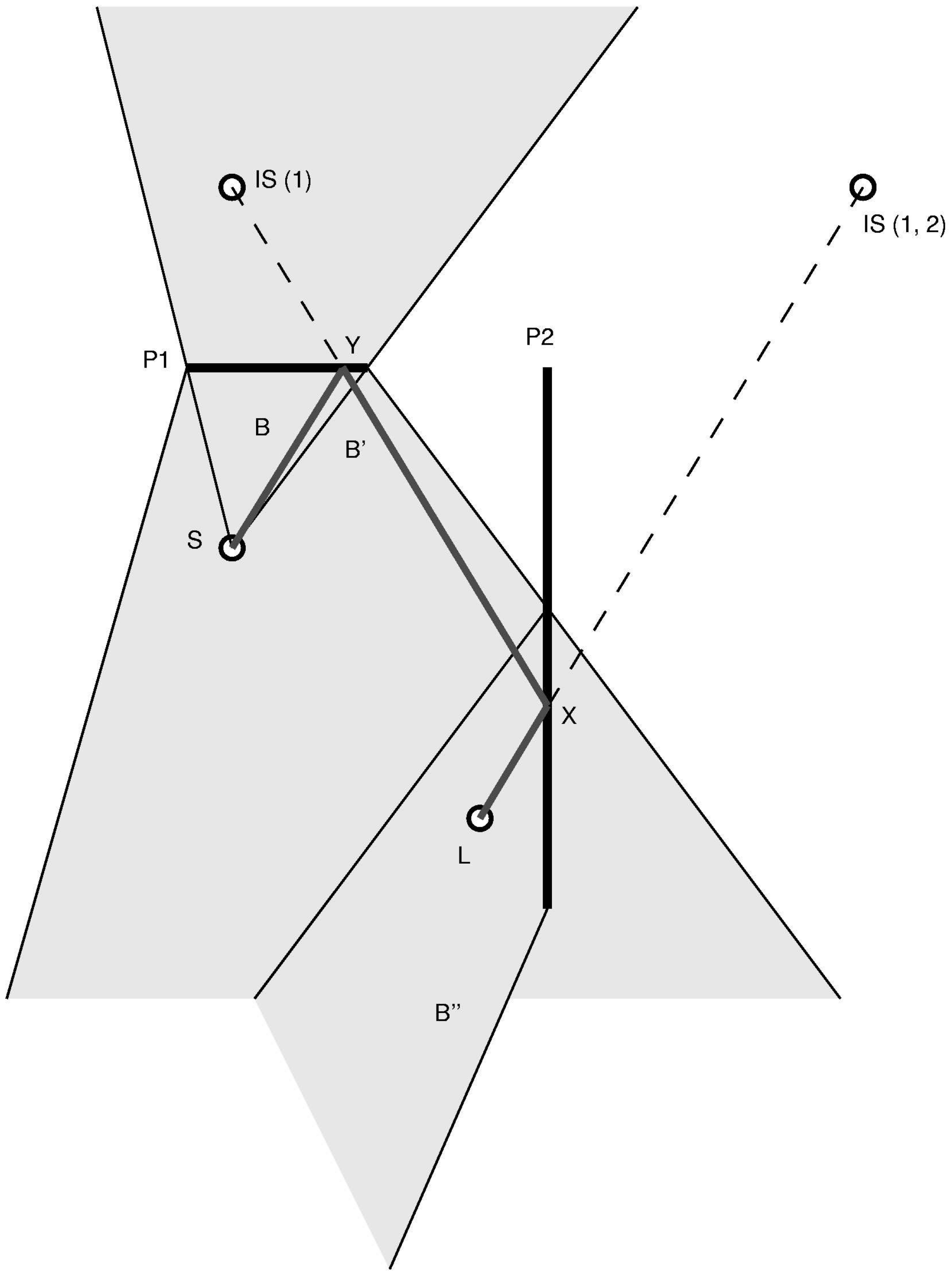

Fig. 2. *To validate the path from source S to listener L via polygons $P1$ and $P2$, the path is tested segment by segment, beginning from listener L. At first, a ray is shot from listener L towards image source $IS(1, 2)$. The intersection point of the ray with polygon $P2$ is X. If there are no intersections with other polygons in the segment $L \rightarrow X$, the first segment is valid. Then, the next segment is validated by shooting a ray from intersection point X towards image source $IS(1)$. The intersection point with polygon $P1$ is Y, and segment $X \rightarrow Y$ is tested for intersections with occluders. The last test is from the intersection point Y to source S. Since there are no occluders, path $S \rightarrow Y \rightarrow X \rightarrow L$ is a valid reflection path.*

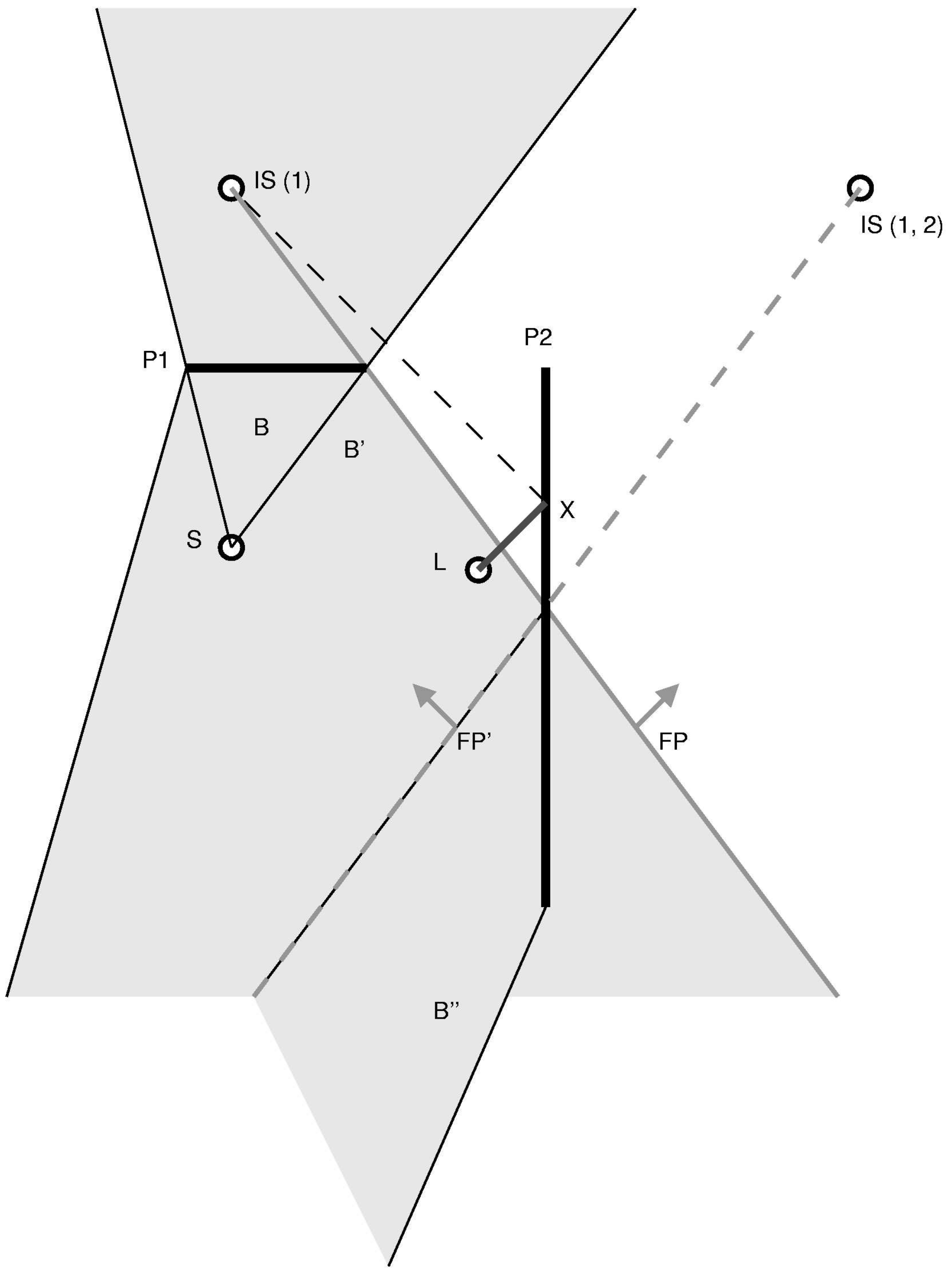

Fig. 3. *The path segment validity test concerning polygon $P2$ has passed, and the next segment to be tested is the one from the intersection point X on polygon $P2$ towards image source $IS(1)$. The test is performed by testing point X against the planes defining beam B'. Since the point is on the wrong side of plane FP, the test fails, and FP becomes the fail plane. The fail plane is propagated down the beam tree by mirroring it against the plane of polygon $P2$ which yields plane FP'. When the listener is updated the next time, the testing begins by comparing the listener position against plane FP', and while it is on the wrong side, the rest of the tests can be skipped. The wrong sides of FP and FP' are indicated by the arrows normal to the fail planes.*

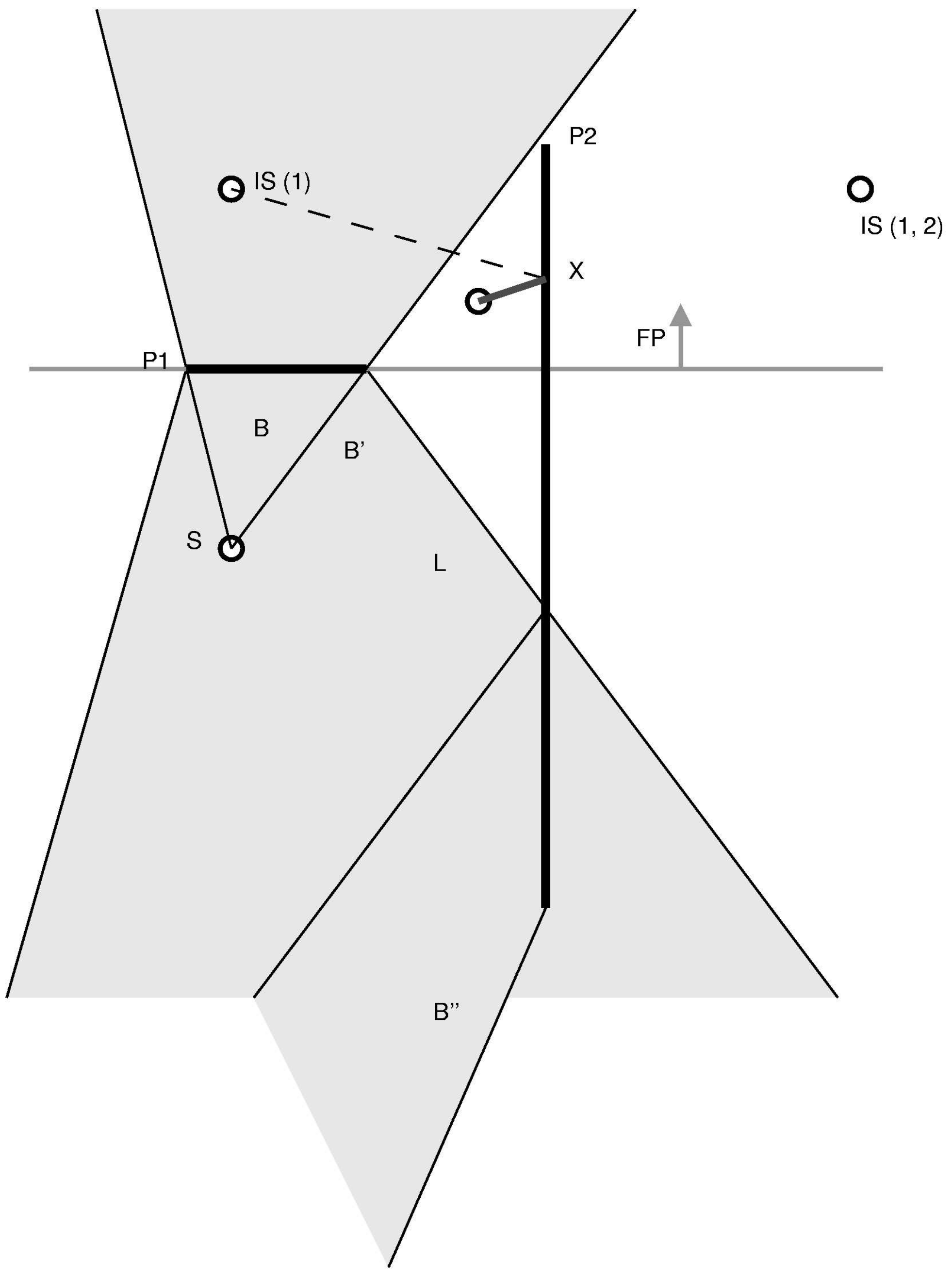

Fig. 4. *If the segment to be tested is entirely on one side of the plane of the polygon defining the beam, there cannot be a reflection. The segment from intersection point* X *towards image source* $IS(1)$ *is entirely behind the plane of polygon* $P1$*. Thus, that plane* FP *becomes the fail plane and it is not necessary to construct the planes for beam* B' *for further tests. The propagated fail plane around the plane of polygon* $P2$ *is not shown since it is the same as the original fail plane in this particular case.*

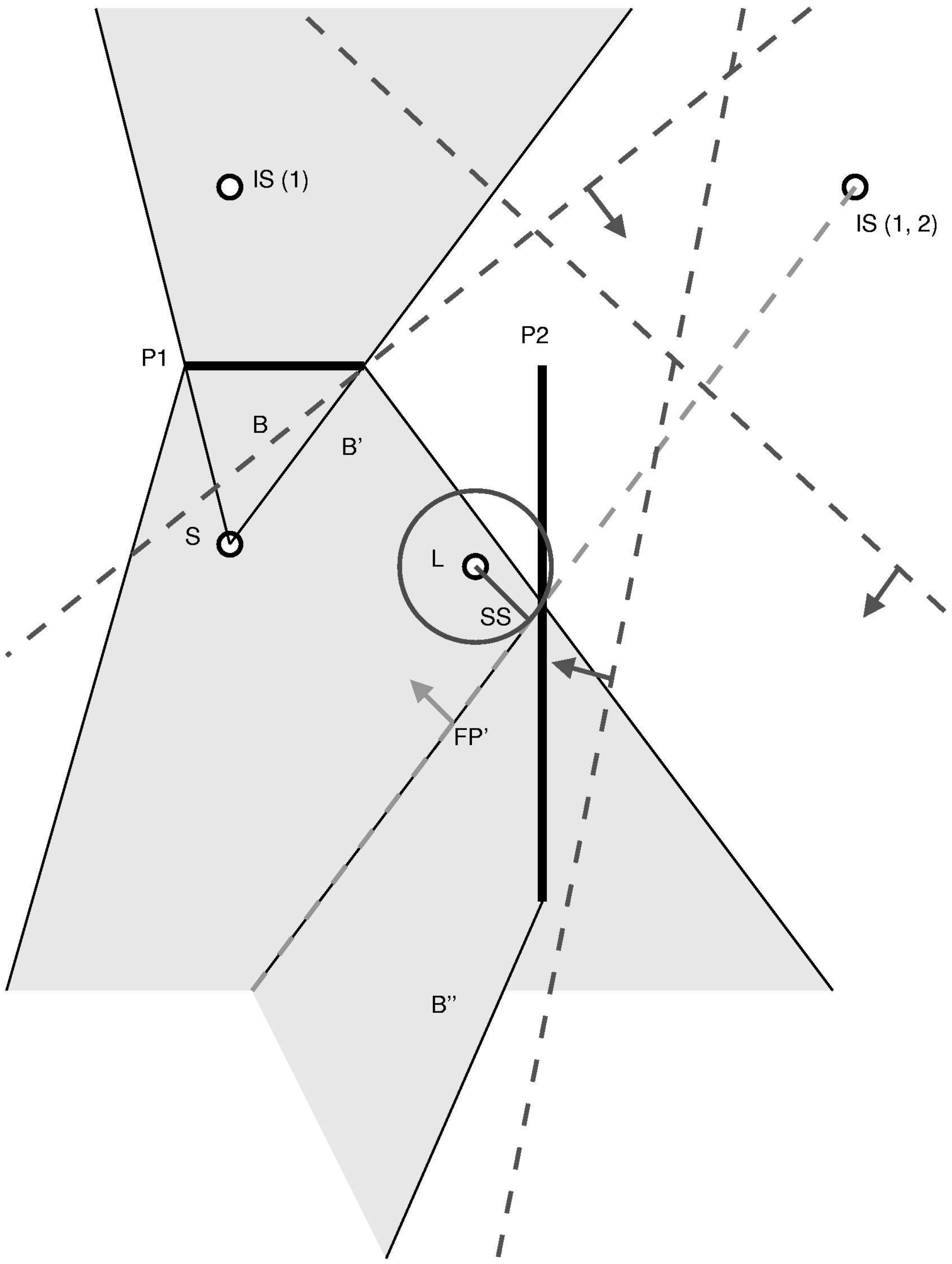

Fig. 5. *Assume that in this case all the paths (not shown in the figure) in the bucket have returned negative results and that the corresponding fail planes have been propagated (dashed lines). Hence the distances from the listener to all the fail planes are calculated and the shortest one is chosen. Here it is the distance to fail plane* FP'*. By using that distance as the radius and the current listener position* L *as the center, the skip sphere* SS *can be created for the bucket. While the listener stays inside* SS *the whole bucket of paths can be skipped.*

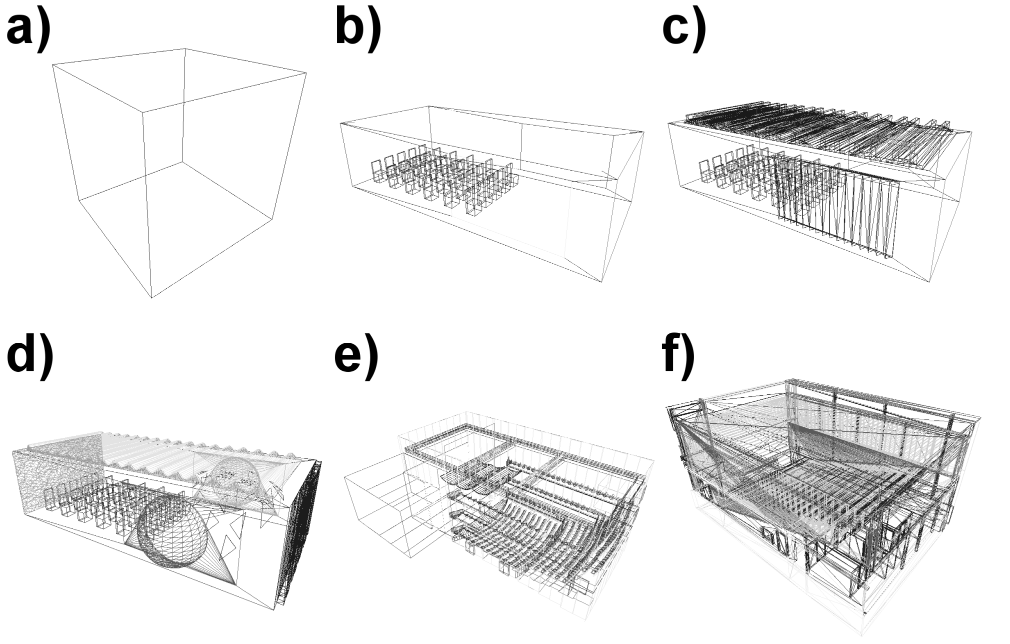

Fig. 6. *The models used in performance tests: a) box, b) simple room, c) regular room, d) complex room, e) concert hall, and f) auditorium.*